ИНФОРМАТИКА

За I година
на средното образование
(гимназиско, средно стручно и уметничко)

2020

Автори:
Данијела Ѓорѓевиќ

Рецензенти:
Доц. д-р Благој Ристевски
Анета Стојческа
Мимоза Мијоска

Лектор:
Јасмина Ѓорѓиева

Издавач:
Министерство за образование и наука на Република Северна Македонија

Печати:
Арс Ламина ДОО, Скопје

Тираж:
233

Со одлука бр.22-270/1 од 14.03.2013 на Националната комисија за учебници, се одобрува употреба на учебникот

CIP - Каталогизација во публикација
Национална и универзитетска библиотека "Св. Климент Охридски", Скопје
004(075.3)
ЃОРЃЕВИЌ, Данијела
Информатика за I година на средното образование : (гимназиско, средно стручно и уметничко) / Данијела Ѓорѓевиќ. - [2. изд.]. - Скопје : Министерство за образование и наука на Република Северна Македонија, 2020. - 302 стр. : илустр. ; 30 см

Библиографија: стр. 290. - Содржи и: Додаток А

ISBN 978-608-226-442-4

COBISS.MK-ID 111960330

ПРЕДГОВОР

Учебникот Информатика е пишуван како учебник по предметот информатика за прва година во средното образование (гимназиско, средното стручно и уметничко) и е приспособен кон Наставната програма дефинирана од Бирото за развој на образованието и одобрена од Министерството за образование и наука.

Следејќи ја Наставната програма, книгата е поделена во шест теми:

Преку првата тема – Хардвер – учениците ќе се запознаат со градбата на компјутерите, со компјутерска архитектура, функционалност на хардверските компоненти, со претставување и меморирање на податоците, со современи хардверски делови на персонален компјутер и со нивна функционалност и карактеристики, како и со современи и најнови технологии на пазарот.

Преку втората тема – Софтвер – учениците ќе се запознаат со поимите за системски и апликативен софтвер, оперативен систем: улога и структура, поим за датотека и систем на датотеки, со архивирање и компресирање на датотеки, со злонамерен софтвер и заштита од него и со поимите за слободен софтвер, пробна верзија, лиценциран софтвер...

Преку третата тема – Обработка на текст – учениците ќе се запознаат со програмите MS Word 2010 и Writer и ќе стекнат вештини за практична работа со стилови, содржина и индекси, шаблони и формулари и со заштита на документи.

Преку четвртата тема – Програмирање во C++ – учениците за првпат ќе се запознаат со поимите алгоритам, алгоритамска структура и програмски јазици. Преку низа примери и задачи ќе се стекнуваат со вештина на програмирање во програмскиот јазик C++.

Преку петтата тема – Табеларни пресметки – учениците ќе се запознаат со програмите MS Excel 2010 и Calc и ќе стекнат вештини за практична работа со напредно користење на формули и функции, напредна работа со графикони, табела како база на податоци, филтрирање на податоци, сортирање на податоци, креирање извештаи – пивот табели, заклучување на ќелии и заштита на работна книга и валидација на податоци.

Преку шестата тема – Компјутерски мрежи и интернет – учениците ќе се запознаат со поимот за компјутерска мрежа, карактеристики на компјутерските мрежи, периферни уреди и дополнителна опрема во мрежа, со мрежна дистрибуција и мрежни дистрибутери и со мрежни додатоци. Ќе се запознаат и со историја, развој и функционирање на Интернетот, како и со WWW – сегашни технологии и технологии кои доаѓаат: Веб 2.0 и семантички веб, со теоретски аспекти на социјалните мрежи, веб - социјални мрежи и прашањата за приватноста на личните податоци

Акцентот во учебникот е ставен на совладување на новите наставни единици преку вежби, решавање на задачи и со практична работа. Со помош на слики и други визуелни помагала ученикот ќе може полесно да ги совлада новите содржини и новите вештини.

На почетокот на секоја тема истакнати се клучни зборови со кои ученикот ќе се сретне во темата.

Во наставните единици има многу примери, вежби и задачи за практична работа. По секоја наставна единица е даден краток преглед во вид на резиме, прашања и задачи за самостојна работа. Исто така, по повеќето наставни единици е истакната рубрика „Вештини кои треба да ги совладаш“ со што на учениците им се укажува на важните елементи за практична работа и за совладување на вештини за работа со програмите.

Во самите наставни единици посебно се истакнати важните делови преку рубриките „Забелешка“, „Важно“ и „Внимавај“. За талентираните ученици се наменети рубриките „За љубопитните“ и „Истражи“ со кои се развива интерес и кај останатите ученици.

На крајот на учебникот се дадени решенија на задачите од програмскиот јазик C++.

Примерите, вежбите и задачите се приспособени на возраста и интересирања на учениците. Тие се осмислени така што преку нив особено внимание е посветено и на комјутерската етика, на заштита на Интернет, на заштита на податоците и на екологијата (електронски отпад).

Авторот

СОДРЖИНА

ХАРДВЕР

Клучни зборови

- Современи компјутери
- Компјутерска архитектура
- Хардвер
- Централна единица
- Процесор
- Матична плоча
- Магистрали
- Внатрешна меморија
- Надворешни мемории
- Влезни единици
- Излезни единици
- Влезно-излезни единици
- Бит
- Бајт
- Конфигурација
- Технологија заснована на допир
- Технологија без допир
- 3Д технологија на слика
- Холографија
- Хелиодисплеј технологија

1.1 Поделба на современите компјутери

Денес компјутерите претставуваат средство за секојдневна работа, за забава и за комуникација и се оставен дел на голем број машини, на пр. банкомати, автомобили, апарати за домаќинство и сл.

Потсети се!

Компјутер е електронски уред на кој можат да му се даваат инструкции за прием, за обработка, за чување, за прикажување на податоци и информации.

Постојат повеќе поделби на компјутерите, но наједноставната и најзначајната е поделбата според големината и можностите на компјутерите. Според оваа поделба, компјутерите се делат на:

- суперкомпјутери,
- големи (mainframe) компјутери,
- миникомпјутери и
- микрокомпјутери (персонални компјутери).

Суперкомпјутерите се најбрзите, најмоќните и најскапите компјутери. Тие се многу големи, имаат голем број на процесори и огромен капацитет на хард дисковите. Се применуваат за воените цели, за научноистражувачките цели (на пр. проучување на вселената, симулирање и моделирање на физичките и на хемиските процеси и др.), а денес најактуелната примена им е за симулирање на земјотреси, урагани, суши, поплави и за следење на глобалната клима. Произведувачите на авиони и автомобили дизајнираат нови модели и ги тестираат под различни услови со користење на симулации со помош на суперкомпјутерите.

Сл. 1. 1 Суперкомпјутер во лабораториите во Лос Аламос

Големите централни компјутери (mainframe) се исто така, големи, моќни и скапи машини кои се користат во големите организации, како што се државните институции, осигурителните компании, банките итн. Се користат за обработка на масовни податоци, на пр. податоци од гласање или попис, статистички податоци, финансиски трансакции и сл., а често се користат како *сервери*.

Основната разлика помеѓу суперкомпјутерите и големите компјутери е во тоа што суперкомпјутерите извршуваат мал број задачи што е можно побрзо, додека големите компјутери опслужуваат голем број корисници и извршуваат голем број задачи

истовремено. Корисниците до централниот компјутер пристапуваат преку нивните терминали кои се поврзани со компјутерот. Терминалите немаат сопствена меморија, имаат само единици за примање и за праќање податоци до главниот компјутер. Тие можат да бидат лоцирани во иста просторија со компјутерот, но и во други простории, згради или градови.

Сл. 1. 2 Голем централен компјутер

Забелешка:

Суперкомпјутерите, всушност, се посебен вид на големите компјутери.

Миникомпјутерите се многу помали и се многу поевтини од големите компјутери. Овој термин се користи за компјутерите кои можат да извршат многу задачи што не бараат користење на големи скапи компјутери, а не можат да бидат сработени на помалите евтини компјутери. Миникомпјутерите имаат слични карактеристики како и големите централни компјутери, но со многу поограничени можности. Работат со неколку процесори и можат да поддржат 4 - 200 корисници во исто време. Се користат за т. н. апликации во реално време, на пр. за контрола на воздушен сообраќај или за автоматизација на фабриките.

Сл. 1. 3 Миникомпјутер

Микрокомпјутерите или персоналните компјутери (PC) се мали, релативно евтини компјутери наменети за работа на еден корисник. Во споредба со досега разгледуваните поголеми компјутери, тие имаат многу ограничени можности затоа што имаат само еден процесор (микропроцесор по кој овие компјутери го добиле своето име), можат да опслужат само еден корисник во одредено време, многу се побавни и можат да чуваат и да обработуваат многу помало количество на податоци. Но, заради ниската цена и лесната употреба, одлични се за мали претпријатија, за училишта и за домашна употреба. Се користат за обработка на текстови, за гледање филмови, за слушање музика, за играње, за пресметување, за програмирање, за сметководство, за графички дизајн итн.

Со развојот на технологијата микрокомпјутерите добиваат моќни перформанси и денес имаат можност да работат во мрежа не само како клиенти туку и како сервери.

Персоналните компјутери можат да бидат *статични* (desktop) и *преносливи* (laptop, notebook, tablet, palm, PDA, IPod, iPhone итн.).

Статичен компјутер е дизајниран така што неговите основни делови (куќиштето, тастатурата, мониторот и глувчето) се одвоени и можат да се стават на масичка. Куќиштето може да биде поставено хоризонтално и вертикално и на него се приклучуваат останати делови преку порти. Најголемиот дел од портите се наоѓаат од задната страна на куќиштето. Во последното време се користат универзалните порти (USB) преку кои можат да се приклучат различни периферни уреди.

Сл. 1. 4 Статичен компјутер

Сите делови кај преносливите компјутери се интегрирани во една целина и најголемиот недостаток на овие компјутери е што тие не можат да се надоградуваат. Некои уреди можат да се приклучат преку порти. Преносливите компјутери содржат батерија за самостојна работа. Laptop е преносли PC кој може да се користи и при патување. Има помали димензии и е поскап од статичните компјутери со исти перформанси. Notebook е сличен на Laptop, но е помал. Тој е со големина на нотес.

Сл. 1. 5 Laptop и Notebook компјутер

Tablet се многу лесни компјутери кои денес се произведуваат во различни форми и големини. Можат да бидат безжични, со екран за цртање или со екран кој може да се држи во исправена или во легната положба.

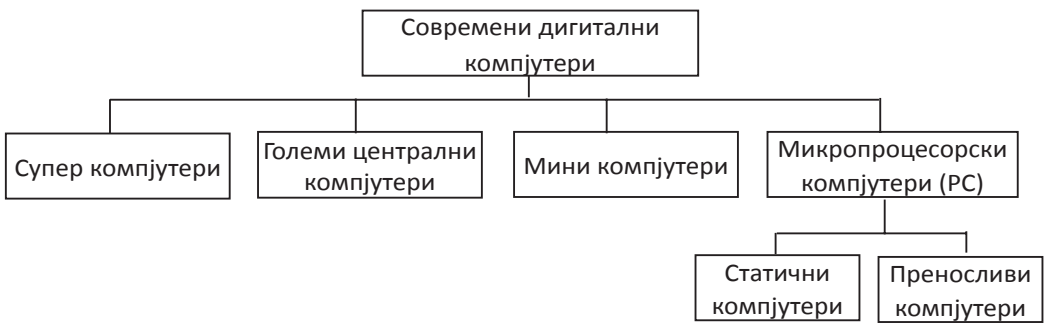
Сл. 1. 6 Tablet PC

Palm или џебни компјутери се многу мали и можат да се држат на дланка. Се користат за водење на деловни календари, како телефонски именици или адресари, за брзи пресметки, за праќање и за примање на податоци и на информации.

PDA (personal digital assistant) компјутерите се со мали димензии и со скромни можности. Се користат како потсетник или како адресар, за електронска пошта и слично. Имаат екран чувствителен на допир и молив со помош на кој се управува со наредби. Можат да се поврзат на Интернет и да имаат функција на мобилен телефон.

Сл. 1. 7 Palm и PDA

Поделбата на компјутерите според големината и можностите може да се претстави со следниов шематски приказ:



Сл. 1. 8 Поделба на современите компјутери

Освен наведените видови компјутери постојат и компјутери кои се вградуваат во други машини или во нивните делови, т.н. *вгнездени компјутери*. За разлика од другите компјутери, вгнездените компјутери немаат монитор и тастатура и не работат самостојно. Тие имаат програма која не може да се менува и е наменета да извршува само една специфична работа, на пр. контрола на температура и влажност, контрола на работа на срце, надзор за обезбедување на простории и зграда итн. Вгнездените компјутери се вградуваат во уреди како што се дигитални фотоапарати, мобилни телефони, музички плеери, микробранови печки и сл. Сè повеќе се вградуваат и во многу уреди кои претходно работеле без нив, како што се на пр. машини за перење, светилки, навигациски и сопирачки системи кај автомобилите итн.

Сл. 1. 9 Микропроцесор вграден во овие патики за трчање го пресметува притисокот помеѓу нозете на тркачот и подлогата пет милиони пати во секунди и постојано ја менува положбата на перничиња за да се добие поголема удобност.

Резиме

Според големината и можностите компјутерите се делат на: супер компјутери, големи компјутери, мини компјутери и микро компјутери (PC).

Суперкомпјутерите се најбрзите, најмоќните и најскапите компјутери.

Големите компјутери можат да чуваат и да обработуваат големо количество податоци, да извршуваат голем број задачи и да опслужуваат стотици корисници во исто време.

Миникомпјутерите имаат слични карактеристики како и големите централни компјутери, но со поограничени можности.

Персоналните компјутери се предвидени за работа на едно лице. Тие можат да бидат статични (desktop) и преносливи (laptop, notebook, tablet, palm, PDA, IPod, iPhone итн.).

Вгнездените компјутери се компјутери кои се вградуваат во други машини или нивните делови. Тие немаат монитор и тастатура и имаат програма која не може да се менува и е наменета да извршува само една специфична работа.

Прашања:

1. Што е компјутер?
2. Како се делат современите компјутери според големината и можностите?
3. Накратко опиши ги суперкомпјутерите и наведи каде се користат!
4. Накратко опиши ги големите компјутери и наведи каде се користат!
5. Накратко опиши ги миникомпјутерите и наведи каде се користат!
6. Како се делат персоналните компјутери?
7. Направи споредба помеѓу статичните и преносливите персонални компјутери!
8. Наведи неколку видови на преносливите персонални компјутери!
9. Што се вгнездени компјутери и кои се нивните карактеристики?
10. Наведи неколку примери за примена на вгнездените компјутери!

1.2 Компјутерска архитектура

Потсети се!

Сите физички делови на компјутерот кои се изработени од тврда материја се нарекуваат хардвер (hardware).

Основната функција на секој компјутер е обработка на податоци. За компјутерот да ја изврши оваа функција, сите негови делови мора да бидат поврзани и да работат како целина.

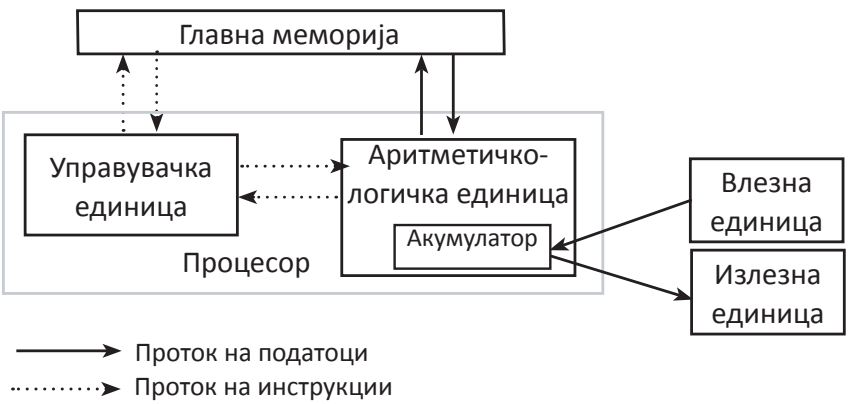
Компјутерска архитектура означува составни делови на компјутер и нивната меѓусебна поврзаност во функционална целина.

1.2.1 Фон-Најманов модел на компјутер

Теоретските основи на компјутерската архитектура ги поставил унгарско-американскиот математичар Џон фон Најман (John von Neumann) во 1945-та година врз основа на следниве принципи:

- структурата, односно градбата на компјутерот не зависи од задачата која се извршува на него;
- компјутерот мора да има способност да меморира инструкции; инструкциите и податоците се меморираат на ист начин во иста единица наречена меморија;
- инструкциите се извршуваат редоследно и во еден момент може да се изврши само една инструкција.

Фон Најман во 1946 година претставил нацрт за модел на првиот компјутер за општа намена:



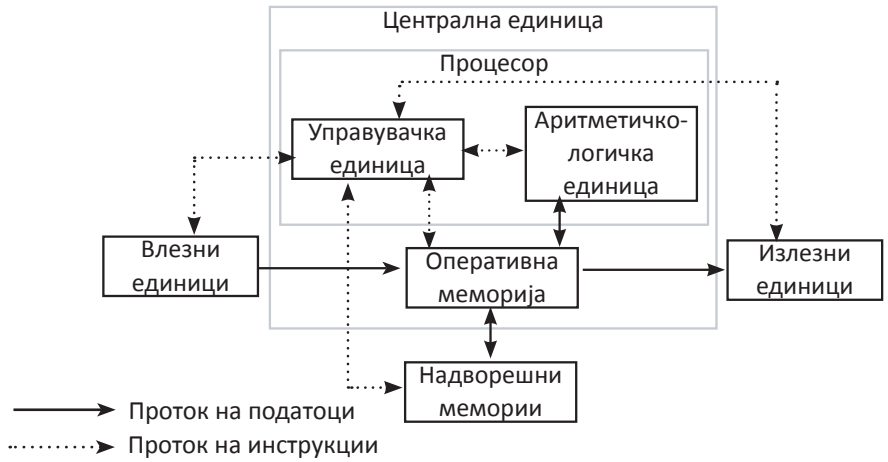
Сл. 1. 10 Поедноставен шематски приказ на фон Најмановата архитектура на компјутерите

Од сликата се гледа дека фон Најмановиот модел на компјутер се состои од:

- главна меморија
- процесор кој се состои од аритметичко-логичка единица и управувачка единица
- влезна единица
- излезна единица.

1.2.2 Современ модел на персонален компјутер

Иако идејата на фон Најман останала во фаза на нацрт, таа претставува основа за наредните генерации компјутери. Структурата на современите персонални компјутери, иако многу посложена и денес е заснована на архитектурата на фон Најмановиот модел.



Сл. 1. 11 Поедноставен шематски приказ на современиот компјутер

1.2.3 Основни функции на хардверските компоненти

Сите хардверски единици на компјутерот мора да бидат меѓусебно поврзани за да функционираат како целина. Целта на компјутерот е да прима податоци од влезните единици, да врши нивната обработка, да меморира податоци, а резултатите да ги прикажува на излезните единици.

Управувачка единица – УЕ, (Control Unit – CU) ги контролира сите делови на компјутерот и управува со операциите кои тие ги извршуваат. *Аритметичко-логичка единица* – АЛЕ (Arithmetic-Logic Unit – ALU) со посредство на управувачката единица управува со *процеси*, односно извршува основни аритметички и логички операции врз податоците зачувани во меморијата. Очигледно е дека АЛЕ и УЕ се тесно поврзани поради што тие не се раздвојуваат и заедно сочинуваат *централна процесорска единица* – ЦПЕ (Central Processing Unit – CPU) или скратено *процесор*.

Влезните единици обезбедуваат внесување на податоци (броеви, текст, слика, звук) и задавање инструкции на компјутерот. Податоците и инструкциите кои се внесуваат преку влезните единици се трансформираат во облик препознатлив за компјутер и се зачувуваат во делот од меморија наменет за таа цел.

Внатрешна или оперативна меморија служи за чување податоци и инструкции кои непосредно му се потребни на процесорот во процесот на обработка, а тоа се програми кои во моментот се извршуваат и податоци потребни на овие програми.

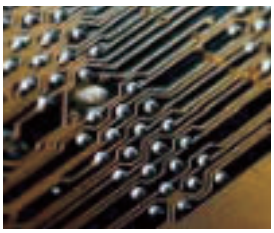
Резултатите на обработка се проследуваат до *излезните единици* каде тие повторно се трансформираат во форма препознатлива за човек.

Единиците на надворешните мемории се користат за трајно чување податоци и програми. Внатрешната меморија не е погодна за трајно чување податоци и програми, па се јавува потреба од надворешните мемории.

Начин на поврзување и на комуникација помеѓу основните делови на компјутерот

Врските помеѓу сите делови на компјутерот се остваруваат преку комуникациски линии кои се нарекуваат *магистрали*. Магистрали (bus) се најобични жици кои претставуваат пат или врска по која се пренесуваат електрични импулси. Постојат неколку видови магистрали:

- Податочна магистрала (data bus) претставува врска преку која се пренесуваат податоци од едно место на друго.
- Адресна магистрала (address bus) пренесува адреси кои одредуваат точно место на читање или на запишување на поединечни податоци.
- Контролна или управувачка магистрала (control bus) пренесува управувачки сигнали и неа најчесто ја користи управувачката единица.



Сл. 1. 12 Магистрали

1.2.4 Претставување и меморирање на податоци

Основниот елемент од кој се изградени електронските делови на компјутерите е транзисторот кој може да биде во две состојби – вклучен или исклучен. Овие две состојби соодветствуваат на бинарните цифри 1 и 0. Со 0 се означува дека не протекува струја, додека со 1 се означува дека има струја во елементот. Заради оваа особина на транзисторите, сите податоци и инструкции во компјутерот се претставуваат со записи составени од нули и единици (бинарни записи).

Бројниот систем кој за основа ги има овие две цифри се нарекува *бинарен броен систем,* а цифрите 0 и 1 се нарекуваат *бинарни цифри*.

За љубопитните:

Првиот компјутер, наречен ENIAC, користел декаден броен систем. Една цифра се претставувала со 10 електронски ламби од кои само една светела. За Џон фон Нојман било очигледно дека програмирање на компјутер кој работи на ваков начин е неефикасно. Тој сфатил дека податоци и компјутерски програми можат да се претстават преку бинарен броен систем во т.н. дигитална форма. Програмата која ја разбира компјутерот се состои од низа на бинарни записи.

Меморијата се состои од ќелии, во една ќелија може да се зачува само една бинарна цифра. Ова е најмало количество на информација кое може да се запомни во меморија и се нарекува ***бит*** (bit = binary digit). Битот може да има вредност 0 или 1.

Битовите се групираат во низа од 8 битови кои се нарекуваат ***бајт*** (byte). Со еден бајт може да се претстави една цифра, една буква или еден знак.

Бит се означува со b, а бајт со B.

1	1	0	0	1	0	0	0

Сл. 1. 13 1B = 8 b

Во внатрешната меморија постојат различни локации за чување на различни видови податоци и инструкции. Секој податок во меморијата има своја единствена адреса со што се овозможува користење на саканите податоци. На тој начин, процесорот може да

пристапи и да преземе потребен податок. Кога во некоја мемориска локација (адреса) ќе се запише некоја содржина, нејзината претходна содржина се губи. Капацитетот на меморијата се изразува со бајти, односно со поголеми единици – килобајт (1KB = 1024 B), мегабајт (1MB = 1024KB), гигабајт (1 GB = 1024MB), терабајт (1TB = 1024GB) итн.

За љубопитните:

Колку различни податоци можат да се претстават со еден бајт?

Ќе направиме аналогија помеѓу декаден и бинарен броен систем. На секое место во декаден броен систем може да стои една од 10 цифри (0, 1, 2 ... 9). Со две цифри можат да се претстават 100 броеви ($100 = 10^2$), со три цифри можат да се претстават 1000 броеви ($1000 = 10^3$) итн. На ист начин, со две бинарни цифри можат да се претстават $2^2 = 4$ бинарни записи, со 3 бинарни цифри можат да се претстават $2^3 = 8$ бинарни записи, а со 8 бинарни цифри можат да се претстават $2^8 = 256$ бинарни записи. Со овие записи се претставуваат 256 различни знаци. Побарај на Интернет ASCII табела!

Резиме

Компјутерска архитектура означува составни делови на компјутер и нивната меѓусебна поврзаност во функционална целина.

Фон Нојмановиот модел се состои од: меморија, процесор, влезна единица и излезна единица. Структурата на современите персонални компјутери е заснована на архитектурата на фон Нојмановиот модел.

Управувачката единица управува со сите делови на компјутерот. А*ритметичко-логичката единица* извршува основни аритметички и логички операции врз податоци. Овие две единици заедно го сочинуваат *процесор*. Влезните единици обезбедуваат внесување податоци и задавање на инструкции на компјутерот. *Внатрешната* или *оперативна меморија* служи за чување на податоци и на инструкции кои непосредно му се потребни на процесорот. Обработените податоци се проследуваат до излезните единици. Единиците на надворешните мемории се користат за трајно чување податоци и програми.

Врските помеѓу деловите на компјутерот се остваруваат преку комуникациски линии кои се нарекуваат *магистрали*.

Сите податоци и инструкции во компјутерот се претставуваат со помош на броеви и тоа само со две цифри 0 и 1. Најмалата количина на информација која може да се прикаже со една бинарна цифра се нарекува *бит* (b). Низа од 8 битови претставува *бајт* (B). Капацитетот на меморија се мери со бајти, односно со поголеми единици – килобајт, мегабајт, гигабајт, терабајт итн.

Прашања:

1. Што претставува компјутерска архитектура?
2. Кој е придонесот на Џон Фон Нојман за развој на компјутерите?
3. Врз основа на кои принципи се заснова фон Нојмановиот модел на компјутер?
4. Од кои делови се состои фон Нојмановиот модел на компјутер?
5. Дали и современите компјутери се засноваат на принципите на фон Нојмановиот модел на компјутер?
6. Кои делови го сочинуваат процесорот? Објасни ја нивната функција!
7. Која е функција на влезните и на излезните единици?
8. Која е функција на надворешните мемории?

9. Што се магистрали и каква улога тие имаат во компјутерот?
10. На кој начин се претставуваат сите податоци и инструкции во компјутерот? Зошто?
11. Што е бит, а што е бајт?
12. Со која единица се мери капацитет на меморија?
13. Знаеме дека 1 km = 1000 m. Дали можеш да објасниш зошто важи 1 KB = 1024 B, а не 1 KB = 1000 B?
14. Колку книги од 500 страници може да се зачуваат на едно CD со капацитет 850 MB? (во просек една страница има 1800 знаци).

Истражи!

Надворешните мемории не користат струја, но и тие податоците ги чуваат како бинарни записи кои се претставуваат со помош на две состојби. Истражи како податоците се зачувуваат на хард диск, а како на компакт диск!

1.3 Современи хардверски делови на персонален компјутер

Основната компјутерска конфигурација ја сочинуваат куќиштето, мониторот, тастатурата и глувчето. Постојат и многу други делови кои не се неопходни за работа со компјутерот но во многу нешта ни помагаат во работата. Овие делови се нарекуваат дополнителна опрема.

Потсети се!

Машинскиот дел на компјутер глобално може да се подели на централен дел и на периферни единици. Централниот дел се состои од процесор и внатрешна меморија. Периферните делови се влезни и излезни единици и единици на надворешни мемории.

Од појавата на првите електронски компјутери па до денес, евидентен е развој на хардверот. Ако се разгледаат перформанси на компјутерите кои се користеле само 15 години порано, може да се забележи голема разлика во бројот на извршени операции во секунда, како и во капацитетот на внатрешната и на надворешните мемории кои тогаш се користеле. Исто така, се развиваат и се појавуваат нови периферни единици.

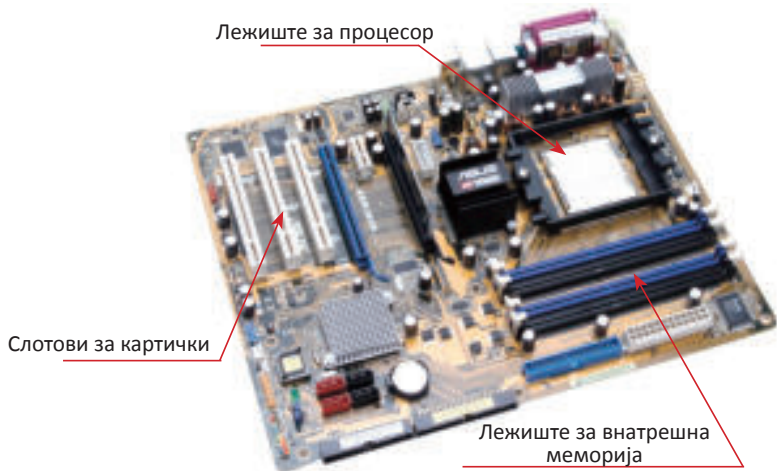
1.3.1 Централна единица

Централната единица (ЦЕ) или куќиште е метална кутија во која се сместени главните делови на компјутерот и има заштитна улога Во неа се наоѓаат главните елементи на компјутерот: матична плоча (на која се поставени процесор, внатрешна меморија и картички) и надворешни мемории. На предната страна од куќиштето се наоѓаат разни копчиња и индикаторски ламбички. Копчето за вклучување и за исклучување на компјутерот најчесто е означено со ON/OFF, 1/0 или POWER. На задната страна од кутијата се наоѓаат приклучоци за периферни единици, како и приклучок за електричната мрежа.

Сл. 1. 14 Централна единица

Матична плоча

Делот преку кој се поврзани сите делови на компјутерот е *матичната плоча* (motherboard). На неа, во посебни лежишта, се поставени процесор и внатрешна меморија. На матичната плоча постојат и места за поврзување на картички (графичка, звучна, мрежна, ТВ и др.) наречени *слотови*. На картичките се наоѓаат порти преку кои периферните единици се поврзуваат со матичната плоча. Кај денешните матични плочи звучната, а понекогаш и графичката картичка се интегрирани со неа.



Сл. 1. 15 Матична плоча

Процесор

Процесор или *централна процесорска единица* е основна компонента на компјутерот во кој се извршуваат инструкции и се врши обработка на податоци. Во персоналните компјутери се вградува микропроцесор кој е изработен од силициумски полупроводнички чип со повеќе десетици милиони (денес и милијарди) транзистори. Микропроцесорот врши обработката на податоци и координација на работата на сите уреди кои се поврзани со компјутерот.

Сл. 1. 16 Процесор

Важни делови на секој процесор се:

- аритметичко-логичка единица
- управувачка единица
- регистри.

Податоците кои доаѓаат од оперативната меморија се чуваат во *регистри* – мемориски ќелии со мал капацитет. Овие податоци ги обработува *аритметичко-логичката единица*. Основни операции кои таа може да ги изврши се аритметички операции (+, -, *, /) и логички операции (И, ИЛИ, НЕ). Управувачката единица определува кои податоци ќе се префрлат од оперативната меморија и кои операции врз нив ќе се извршат. По обработката податоците повторно се враќаат во оперативната меморија.

Процесорот се става во соодветно подножје на матичната плоча. Тој бргу се загрева па во негова близина се вградуваат системи за ладење (coolers). Современите микропроцесори имаат и *ултрабрза кеш (cache) меморија* која најчесто е вградена

во самиот процесор. Во неа се чуваат податоци кои често се користат и на тој начин процесорот има побрз пристап до нив.

Основната карактеристика на процесорот е *брзина* со која тој ги извршува инструкциите, а се изразува со милиони операции во една секунда MIPS (Milion Instruction Per Second).

Брзината на процесорот зависи од *работниот такт* на часовникот (clock) кој обично е поставен на самиот процесор. Кога компјутерот ќе се вклучи, часовникот со голема брзина произведува рамномерни импулси со помош на кои компјутерот врши синхронизација на извршување на инструкции. Брзината на создавањето на импулсите се нарекува фреквенција. Фреквенцијата се изразува во единицата херц (Hertz) со ознака Hz. На денешните часовници фреквенција се мери во гигахерци (GHz), при што 1 GHz означува милијарда тактови во една секунда.

Најпознатите производувачи на процесори се Intel, AMD, IBM итн.

Пр. 1. 1. Современите процесори и нивните технички карактеристики:

Опис	Брзина	Тип
AMD Phenom X4 Quad-Core 9750 Agena 2400MHz	2,4 GHz	Phenom X4 Quad-Core
Intel Core 2 Quad Q8300 2.55 GHz	2.55 GHz	Core 2 Quad
AMD Phenom X4 9850 2.5 GHz	2.5 GHz	Phenom X4

Совет:

Како да избереш процесор?

- За пишување текст, за гледање слики и филмови, за сурфање на Интернет и за играње поедноставни игри доволни се AMD Sempron i Intel Celeron.

- За обработка на слики во програмите како што е Photoshop, проектирање во AutoCAD, 3Д анимација и играње на посложени игри потребни се AMD Athlon X2, AMD Phenom X4, Intel Core 2 Duo, Intel Core 2 Quad.

- За професионална работа со слики, со музика, со филмови и со анимации и за играње на high-end игри со полни детали се потребни Intel i7, Intel Core 2 Extreme, AMD Phenom II X4.

Внатрешна или оперативна меморија

И внатрешната меморија се гради на полупроводнички силициумски чипови составени од милијарди транзистори. Постојат два вида на внатрешна меморија:

- RAM меморија и
- ROM меморија.

RAM меморија

RAM (Random Access Memory) е меморија до која може директно да се пристапи за читање и за запишување податоци и инструкции. RAM меморијата податоците ги чува само додека во неа постои електричен напон. Со исклучување на компјутерот од струја содржината на RAM меморијата се брише и до следното вклучување таа е потполно празна. Со секое ново запишување на податоци во некоја локација, нејзината претходна содржина се брише.

Сл. 1. 17 RAM меморија

Податоците и програмите кои му се потребни на процесорот, во RAM меморијата се вчитуваат од надворешните меморији (обично од хард дискот). Понатаму процесорот овие податоци и програми ги зема директно од RAM меморијата, ги обработува податоците според инструкциите дадени во програмите и резултатите на обработката пак се враќаат во RAM меморијата. Затоа RAM меморија уште се нарекува и работна или оперативна меморија. Од тие причини и брзината на компјутерот е поврзана со големината на RAM меморијата.

Основните карактеристики на RAM меморијата се:

- *време на пристап* – време за кое може да се најде бараниот податок, се мери со пикосекунди (еден трилионити дел од секундата, 1 ps = 10^{-12} s);
- *брзина на пренос* – количина на податоци која може да се прочита или запише во/од меморија за единица време, се мери со MHz;
- *капацитет* – максималната количина податоци која меморија може да содржи, а се мери со мегабајти (MB) и со гигабајти (GB).

За љубопитните:

RAM може да биде статичка (SRAM) и динамичка (DRAM). Податоците запишани во статичката меморија се стабилни и не мора да се освежуваат. Статичката меморија технички е посложена од динамичката, па е поскапа и со помали капацитети. Затоа таа се користи за брзи кеш мемории.

Динамичката меморија мора да се освежува (refresh) за податоците во неа да не се загубат. Освежувањето се врши така што динамичката меморија од време на време ги чита податоците и повторно ги запишува на истото место. Затоа динамичката меморија е побавна од статичката. Динамичката RAM меморија може да биде синхрона и асинхрона. Почесто се користи синхрона динамичка меморија SDRAM и нејзините поттипови (DDR SDRAM, DDR 2, DDR 3, и др.).

Пр. 1. 2. Современите меморији и нивните технички карактеристики:

Опис	Капацитет	Брзина	Тип
Kingston KVR 512MB SDRAM 133 CL3 DIMM	512 MB	133 MHz	SDRAM
PQI DDR 2GB PC400 CL2.5 Turbo Kit	2GB	400 MHz	DDR
Kingston KVR 1GB DDR2 800Mhz kit of two	1GB	800Mhz	DDR II

ROM меморија

ROM (Read Only Memory) се користи за трајно чување на податоци и на инструкции кои произведувачот ги запишува во неа. Обично тоа се програми кои го проверуваат хардверот секогаш кога компјутерот ќе се вклучи во струја (BIOS – Basic Input Output System). Содржината на ROM меморијата не може да се избрише ниту да се менува.

Сл. 1. 18 ROM меморија

За љубопитните:

Постојат и програмибилни ROM мемории – PROM (Programmable ROM). PROM е слична со ROM меморијата, освен што таа на почеток е празна и набавувачот на компјутерот со помош на специјални уреди ја пополнува со специјални инструкции. Од тој момент таа се однесува како ROM. Подобрените варијанти на оваа меморија се EPROM (Erasable PROM) и EEPROM (Electrically Erasable PROM).

1.3.2 Влезни единици

Влезните единици се користат за внесување на податоци и на инструкции во компјутерот. Тастатурата и глувчето се најчесто користени влезни единици. Други влезни единици се скенер, микрофон, електронски молив, оптички читач, сензори и многу други.

Тастатура

Тастатурата е неопходна влезна единица со помош на која се внесуваат букви, специјални знаци и броеви во компјутерот. Со комбинации на некои копчиња можат да се внесат и одредени инструкции. Кај современите тастатури копчињата воглавно се поделени во пет групи: текстуалниот дел (во кој се наоѓаат копчиња за внесување на знаци), функциискиот дел (во кој се наоѓаат копчињата F1-F12), делот за навигација (во кој се наоѓаат копчињата со стрелки и копчињата Home, End, Page UP, Page Down, Insert и Delete), нумеричкиот дел (во кој се наоѓаат копчињата со цифри и знаци за аритметички операции) и дел за мултимедија.

Сл. 1. 19 Тастатура

Глувче

Со помош на *глувчето* се означуваат објекти на екранот и се праќаат наредби до компјутерот. Глувчето обично има 2 или 3 контролни копчиња. Денес најмногу се користат оптички глувчиња, а постојат и механички глувчиња. Глувче се приклучува преку PS/2 или преку USB порта. Врската со компјутерот се остварува преку кабел или по безжичен пат.

Сл. 1. 20 Оптички глувчиња

Скенер

Скенер е уред за внесување на слика или текст во компјутерот во облик на графички приказ. Квалитетот на приказот зависи од резолуција на скенерот. Во поново време скенерите поддржуваат софтвер за препознавање на текст. Технологија со која текстот се претвара во дигитална форма е од голем значење за користење на пишувани извори и креирање на дигитални библиотеки лесно достапни на сите корисници на Интернет.

Сл. 1. 21 Скенер

Оптички читачи

Оптички читач е уред кој препознава рачно пишувани или печатени знаци на точно определени места и ги претвора во податоци разбирливи за компјутерот. Постојат три вида на оптички читачи:

- Уред за читање на означени полиња (Optical Mark Readers – OMR) препознава присуство на соодветна ознака во одредено поле (пример ЛОТО ливче, тестови итн.) со помош на инфрацрвена светлина.

Сл. 1. 22 OMR

- Уред за читање на печатени знаци (Optical Character Recognition – OCR) скенира текст како графички приказ кој потоа со соодветни програми се препознава и повторно се претвора во текст.

Сл. 1. 23 OCR

- Уред за читање на линиски код или бар-код читач (Optical Barcode Reader – OBR). Бар-код е шифра на артикли претставена со низа од тенки и дебели линии со определен простор меѓу нив.

Сл. 1. 24 OBR

Магнетен читач

Магнетен читач е уред кој чита картички на кои освен текстот е нанесена и магнетна лента. Пример се картички за евидентирање на работно време и банковни картички.

Сл. 1. 25 Магнетен читач

Аудио-визуелни уреди

Аудио-визуелни уреди се уреди кои овозможуваат внесување на аудио-визуелните податоци (звук, слика и видео) во компјутерот и нивно претворање во дигитална форма. Такви уреди се микрофон, дигитален фотоапарат и видео камера.

Сл. 1. 26 Микрофон, дигитален фотоапарат и камера

1.3.3 Излезни единици

Излазните единици се користат за приказ на податоци кои претставуваат резултат на обработка на компјутерот. Неопходна излезна единица е монитор, а освен него се користат печатач, цртач, проектор, звучници итн.

Монитор

На екранот на *мониторот* се прикажуваат знаци, слики и цртежи. Мониторот со компјутерот се поврзува преку *графичка картичка* која обезбедува соодветна *графичка резолуција*. Графичката резолуција се претставува со бројот на точките на екранот (на пр. 800x600, 1024x768). Овие точки се наречени *пиксели* (pixels).

Постојат два основни вида на монитори:

- монитори со катодна цевка – CRT (Catode Ray Tube) и
- монитори со течни кристали – LCD (Liquid Crystal Display).

Порано кај статичните компјутери се користеле CRT мониторите, додека LCD мониторите се користеле само кај преносливите компјутери. Денес и кај статичните компјутери најчесто се користат LCD монитори.

Сл. 1. 27 CRT монитор

LCD мониторите припаѓаат на современата технологија. Тие се рамни, имаат подобар квалитет на сликата, помала потрошувачка на електрична енергија и помало зрачење (Low Radiation). TFT (Thin Film Transistor) монитори се подвид на LCD монитори.

Сл. 1. 28 LCD монитор

Големината на мониторот се изразува во инчи и се мери по дијагоналата на мониторот. Инч (inch) е единица мерка за должина и често се означува со знакот " (1" = 2,54 cm). Стандардните големини на мониторите се 17", 19", 22".

Печатач

Печатачот се користи за претставување податоци на хартија или на фолија каде тие можат трајно да се чуваат. Постојат повеќе видови печатачи кои се разликуваат по квалитетот и по брзината на печатење. Денес најчесто се користат матричните, млазните и ласерските печатачи.

Сл. 1. 29 Матричен печатач

Сл. 1. 30 Млазен печатач

Сл. 1. 31 Ласерски печатач

- *Иглични (матрични) печатачи* – печатат со помош на метални иглички кои преку лентата со боја удираат на хартијата. Печатат црно-бело.
- *Млазни (ink-jet) печатачи* – печатат на тој начин што бојата со голема брзина се прска директно на хартијата. Погодни се за печатење во боја.
- *Ласерски печатачи* – печатат со помош на ласер и специјален прав во боја – тонер. Печатат брзо и даваат отпечаток со висок квалитет.

Цртач

Цртач или плотер се користи за печатење цртежи, скици и на други графички прикази. Најголема примена имаат за цртање на технички цртежи, на географски карти и на постери со големи димензии.

Сл. 1. 32 Цртач

1.3.4 Влезно-излезни единици

Влезно-излезните единици служат за влез, меѓутоа и за излез на податоци. Најпознатите влезно-излезни единици се:

- модем,
- звучна картичка

Модем

Модемот служи за пренесување на податоци од еден до друг компјутер, најчесто преку телефонската врска. Модемот прво претвора бинарни информации во звучни

сигнали (модулација), а модемот кој се наоѓа на другиот крај на врската овие звучни сигнали ги претвара во бинарни (демодулација). Брзината на модемот се мери со битови во секунда (bps), односно со килобитови во секунда (kbps).

Постојат два вида на модеми: *внатрешни* (интерни) и *надворешни* (екстерни). Интерниот модем е во облик на картичка и се приклучува на некој од слотовите на матичната плоча. Екстерниот модем е посебен уред кој преку кабел се приклучува со компјутерот.

Сл. 1. 33 Интерен и екстерен модем

Звучна картичка

Компјутерот денес претставува и мултимедијален уред па е неопходно да се обезбедат звучни ефекти, музика и говор. За тоа се задолжени *звучните картички* кои можат да се приклучат на некој од слотовите на матичната плоча. Звукот може да се внесува преку тастатурата, преку клавијатура на музичките инструменти или преку микрофон. За слушање се користат звучници и слушалки.

Сл. 1. 34 Звучна картичка

Звучните картички имаат AD и DA конвертори. AD конвертор се користи на влезот за претворање на звучни сигнали во дигитални, а DA конвертор се користи на излезот за обавување на обратен процес.

1.3.5 Единици за надворешни мемории

Надворешните мемории се користат за трајно чување на податоци. Податоците и програмите кои се чуваат на надворешните мемории се пренесуваат во внатрешната меморија и се користат во процесот на обработка. Времето на пристап до надворешната меморија е многу поголемо од времето на пристап до внатрешната меморија, меѓутоа капацитетот на надворешните мемории е неспоредливо поголем.

Денес како единици на надворешни мемории се користат:

- магнетен или хард диск
- оптички или компакт диск
- полупроводнички мемории во кои спаѓаат USB Flash мемории и мемориски картички.

Хард диск

Хард дискот (HD – Hard Disc) се користи за чување податоци и програми кои се користат во секојдневната работа. Хард дискот може да биде внатрешен или надворешен и има капацитет од неколку стотини гигабајти (GB). Кај дискот за чување податоци се користат метални плочи кои се премачкани со магнетен слој. Секоја плоча има по две глави за читање и за запишување. Главите за читање и за запишување лебдат над плочите додека тие се вртат со огромни брзини (околу 7200 вртежи во минута). Дискот е чувствителен на надворешните влијанија па е затворен во метална кутија.

Сл. 1. 35 Внатрешен и надворешен хард диск

Компакт диск

Компакт диск (CD – Compact Disc) може да биде:

- само за читање (CD-ROM),
- за читање и за запишување само еднаш (CD-R),
- за читање и за запишување произволен број пати (CD-RW).

Сл. 1. 36 Компакт диск

Постојат и оптички дискови со неколку пати поголем капацитет наречени DVD (Digital Video Disk). CD има капацитет околу 700 MB, а DVD неколку GB.

Мемориски стик

Меморискиот стик или *флеш-меморија* денес сè повеќе се користи за пренесување на податоци поради малите димензии (3 - 6 cm) и релативно големите капацитети (денес 16 и повеќе GB). На компјутерот се приклучуваат преку универзалните (USB) порти.

Сл. 1. 37 Флеш-меморија

1.3.6 Конфигурација на компјутер

Основна карактеристика на еден компјутерски систем е неговата конфигурација, т.е. кој процесор, која внатрешна меморија и какви периферни единици тој содржи. Еден компјутер може да користи различни периферни единици во зависност од неговата намена и од потребата на корисникот.

Еден пример на компјутер за домашна употреба е компјутер со Intel Core2Duo процесор, 2GB RAM, хард диск од 160 GB, со тастатура, глувче, LCD монитор, ласерски печатач, CD/DVD единица и USB (Universal Serial Bus) порти.

Резиме

Машинскиот дел на компјутерот се состои од централната единица и перифериските единици – влезни, излезни, влезно-излезни единици и надворешни меморији. Во централната единица се наоѓаат процесор, внатрешна меморија, картички и надворешни меморији.

Процесорот е основна компонента на компјутерот во кој се извршуваат инструкции и се врши обработка на податоци. Основни карактеристики на процесорот се брзината и работниот такт.

Постојат два вида внатрешна меморија: RAM и ROM меморији. RAM е меморија до која може директно да се пристапи за читање и за запишување податоци и инструкции. Основни карактеристики на RAM меморијата се време на пристап, брзина на пренос и капацитет. RAM меморијата може да биде статичка и динамичка.

ROM меморија се користи за трајно чување на програми кои го проверуваат хардверот секогаш кога компјутерот ќе се вклучи во струја.

Влезните единици се користат за внесување на податоци и на инструкции во компјутерот. Тастатурата и глувчето се најчесто користени влезни единици.

Излезните единици се користат за приказ на податоци кои претставуваат резултат на обработка на компјутерот. Најчесто се користат мониторот и печатачот.

Влезно-излезните единици служат за влез, меѓутоа и за излез на податоци. Најкористените влезно-излезни единици се модемот и звучната картичка.

Податоци и програми трајно се чуваат на надворешните меморији Денес како единици на надворешни меморији се користат магнетен или хард диск, оптички или компакт диск USB Flash меморији и мемориски картички.

Прашања:

1. Од кои единици се состои машинскиот дел на компјутерот?
2. Што е централна единица? Кои делови се наоѓаат во неа?
3. Која е функција на матичната плоча во компјутерот?
4. Како се приклучуваат картички на матичната плоча?
5. Што е процесор и од кои делови се состои?
6. Кои се основни карактеристики на процесорот?
7. Од што најмногу зависи брзината на процесорот?
8. Кои се видови на внатрешната меморија?
9. Опиши ги накратко ROM и RAM меморииte! Направи споредба помеѓу нив!
10. Како уште се нарекува RAM меморијата? Зошто?
11. Кои видови на RAM меморија постојат? Објасни ги!
12. Што мислиш, зошто податоците зачувани во ROM меморија се заштитени од бришење и од менување?
13. Која е улога на влезните, излезните и влезно-излезните единици во компјутерот?
14. Наброј неколку влезни единици! Опиши ги тастатурата и глувчето!
15. Наброј неколку излезни единици! Опиши ги мониторот и печатачот!
16. Што е графичка резолуција и кој хардверски елемент определува колкава ќе биде таа?
17. Опиши ги модемот и звучната картичка!
18. Каква е улогата на надворешните мемории во компјутерот?
19. Наброј неколку надворешни мемории! Опиши го хард дискот!
20. Што се подразбира под конфигурација на компјутер?
21. Кои технички карактеристики ги има следнава RAM меморија: PQI DDRIII 512MB PC1066 CL7?
22. Кои технички карактеристики ги има следниов процесор: Intel Core i7-920 2.66 GHz?
23. Прочитај ги следниве конфигурации: Која конфигурација е најдобра?

а)	б)	в)
Intel Core i3-2100 3.1 GHz	Intel Core i7-950 3.06 GHz	Intel Pentium G530 2.40 GHz
4 GB RAM	8 GB RAM	2 GB DDR3 RAM
1 TB HDD	1000 GB (1TB) HDD	500 GB HDD
nVidia GT 440 2GB DDR3	nVidia GTX560 1GB GDDR5	Intregrated Graphics

На што се однесува последната карактеристика во дадените конфигурации?

24. Побарај каталог за компјутери и компјутерска опрема. Спореди ги дадените конфигурации и цените.

Истражи!

Графичката картичка е важен дел од компјутерската конфигурација, особено за видео игри и за работа со слики и со видео датотеки. Побарај на Интернет повеќе за графички картички и за нивните карактеристики!

1.4 Современи и најнови технологии на пазарот

Компјутерската технологија се развива со неверојатна брзина. Денес се појавуваат нови технологии кои работата со компјутерот ја прават едноставна и лесна. Така и компјутерите денес добиваат сè поголема примена во сите области во животот. На пример, модерните мобилни телефони со екран чувствителен на допир денес веќе се вообичаени.

1.4.1 Технологии засновани на допир

Технологии засновани на допир се базираат на површини под кои се поставени сензори чувствителни на допир. Заради своите предности – интуитивно и едноставно користење, овие технологии имаат голема примена. Со специјални програми допирите и движењата на прстите се претвораат во соодветни наредби.

Технологија на допир

Во технологија на допир (touch technology) припаѓа технологија со чувствителна површина која се допира.

Екранот чувствителен на допир (touch screen) е влезна единица која ги заменува тастатурата и глувчето. Тој овозможува внесување на податоци преку допир на соодветно место на екранот. Со ваквиот начин на работа се овозможува директна комуникација со компјутерот без потреба од претходно обучување на корисниците. Од овие причини тие имаат голема примена – на јавните места каде што е овозможен пристап на голем број луѓе, на продажните места за бирање на услуги, во рестораните за нарачка на храна, за покажување на слики од каталог, во банкоматите и слично.

Сл. 1. 38 Touch екран

Некогаш е потребен електронски молив за внесување на податоци. Овој молив симулира рачно пишување или цртање директно на екранот. Постојат два вида на ваквите моливи:

- оптички молив (light pen),
- дигитални табли (digitizing tablet) кои најчесто се користат за цртање – графички табли.

Сл. 1. 39 Оптички молив

Сл. 1. 40 Дигитална графичка табла

Тастатурата чувствителна на допир работи на сличен начин како механичката тастатура со таа разлика што не е потребна сила за притискање на копче, туку таа реагира на допир. Оваа тастатура ги заменува стандардните единици за внесување на податоци со што се намалува ризик од повреди при повторување на исти движења.

Сл. 1. 41 Тастатура чувствителна на допир

Технологија на повеќекратен допир

Технологија на повеќекратен допир (Multi touch) е технологија во која се препознаваат прсти или рака на корисникот при преоѓање преку површина покриена со сензори кои реагираат и на најмал допир. Со оваа технологија се прават екрани, тастатури, глувчиња или нивни комбинации.

Подлогата за допир (touchpad) претставува замена за глувче. Ова е дел чувствителен на допир со кој се управува со курсорот, а најчесто се користи кај преносните компјутери. Под подлогата се наоѓаат сензори кои реагираат на допир и движења на прстите. На тој начин се управува со стрелката на екранот. Со лесен удар се симулира кликнување на копче од глувчето. На подлогата за допир има и две копчиња кои имаат улога на лево и на десно копче од глувчето.

Сл. 1. 42 Подлога за допир

Кај екраните чувствителни на повеќекратен допир различни комбинации на движења со прсти можат да се користат за изведување на други операции, на пр. за отворање и за затворање на документи, за зголемување или за намалување на слики и сл. На истата површина се поместува и курсорот на екранот само што наместо еден прст се користат два прста. Истата површина може да се користи и за пишување или за цртање наместо графички табли.

Сл. 1. 43 Мобилен телефон со Multi touch екран

Сл. 1. 44 Лаптоп со Multitouch екран

1.4.2 Технологија без допир

Технологија без допир веќе се користи за читање на картички во банкарскиот систем, за обезбедување на влезови и слично, при што се користат специјални читачи. Во поново време оваа технологија се развива така што овозможува извршување на едноставни функции со движења на дланката над одредена површина без таа да се допира. Специјални програми овие движења ги претвараат во соодветни наредби. За сега технологијата без допир се применува кај некои мобилни телефони и кај некои преносливи компјутери.

Сл. 1. 45 Читач на картички

Сл. 1. 46 Мобилен со технологија без допир

Сл. 1. 47 Лап топ со технологија без допир

1.4.3 3Д технологија на слика

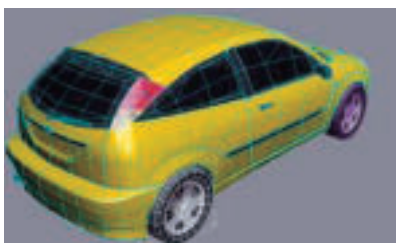
Визуелизација претставува графички приказ на податоци во форма на слика, анимација или видео запис. Денес, по неколку децении на развој, 3Д прикажувањето на слики дава големи можности во многу области. 3Д слика на некој објект се добива така што специјална камера го снима објектот од две перспективи, потоа сликите се прикажуваат со специјални проектори со што се постигнува чувство на длабочина.

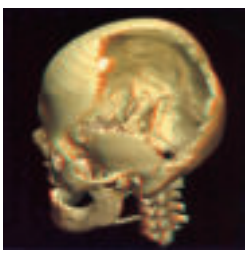
Сл. 1. 48 3Д филм

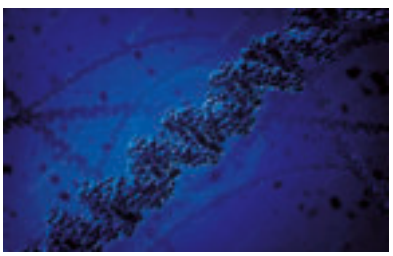
Сл. 1. 49 Очила за гледање на 3Д филмови

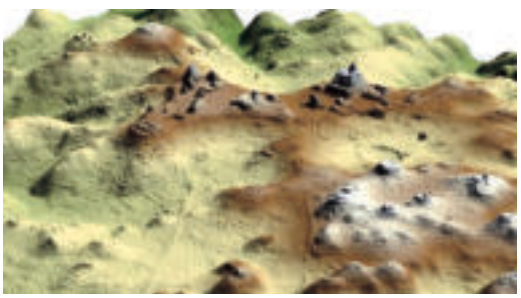
Сл. 1. 50 Лап топ со екран кој прикажува 3Д слики

Најпозната примена на 3Д технологија е кај компјутерските игри, во филмовите и на телевизијата, но таа се применува и за дизајн, за проектирање и за симулации во различни области (науката, медицината, индустријата и др.).









Сл. 1. 51 Примена на 3Д слики во разни области

Виртуелна реалност

Виртуелна реалност е технологија со која реалната околина се заменува со графички приказ на виртуелната околина во облик на 3Д слика или анимација која е генерирана на компјутерот. Виртуелната околина се прикажува преку уреди како што се монитор, LCD проектор, телевизија или на уреди како што се шлем и очила кои имаат екран за секое око.

Виртуелната реалност вклучува најразлични влезни и излезни единици со кои корисникот се поврзува со компјутерот.

Информациите добиени од влезните уреди можат да послужат за манипулирање со објекти во виртуелната околина или за управување со апликацијата.

Влезните единици во виртуелната реалност се:

- сензори на положба, односно на ориентација (motion tracker);
- сензори на сила: space ball;
- сензори на положба на тело: сензорска ракавица (data glove), сензорски костум (body suit);
- сензори на движење: подвижна лента, ергометри и слично;
- други сензори: за препознавање на говор, лице, очи итн.

Сл. 1. 52 Со помош на сензорска ракавица се мерат насоката и брзината на движењата на раката

Сл. 1. 53 Spaceball

Сл. 1. 54 Motion tracking

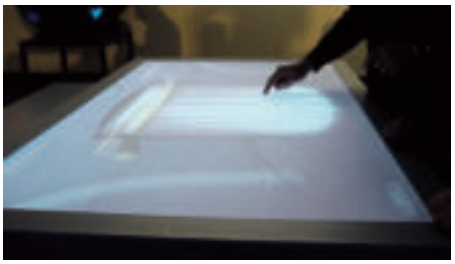
Сл. 1. 55 Сензорска ракавица

Сл. 1. 56 Сензорски костум

Излезните единици се користат за прикажување на компјутерски генерираните слика, звук и/или допир кои корисникот може да ги перцепира.

Излезните единици во виртуелната реалност се:

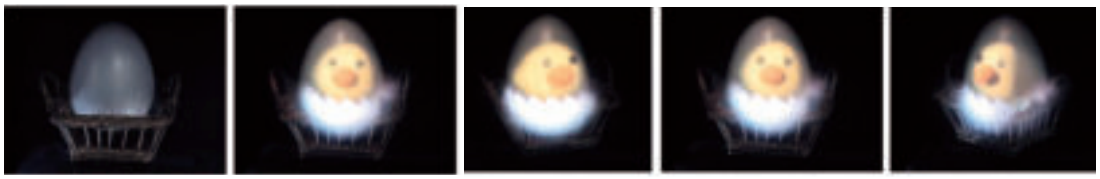
- визуелни излезни единици: шлем (HMD), стерео екрани, проекциски системи (на платно, виртуелна работна маса и слично);
- звучни излезни единици;
- хаптички (чувствителни на допир) излезни единици: подвижни платформи и други;
- други излезни единици: за чувствување на мирис, ветер, топлина итн.



Сл. 1. 57 Виртуелна маса

Сл. 1. 58 HMD

Сензорска ракавица е интерактивен уред кој се носи на рака и е опремен со голем број сензори. Со помош на сензорите се регистрираат положбата на шепата и позицијата на прстите. Корисникот на тој начин може да манипулира со движења во виртуелната средина. Сензорската ракавица може да се комбинира со симулаторите на сила или допир со што таа станува и хаптички излезен уред.



Сл. 1. 59 Демонстрација на проект кој може да прикаже 3Д слика на објекти од реален свет. 3Д слика на испилување е прикажана на вештачко јајце ставено во вештачко гнездо. Кога гледачите ќе го променат аголот на гледање се гледа друг дел од пилето. Проектирана слика може да се гледа од повеќе луѓе од повеќе агли, и секој го гледа како што би го гледал во реалниот свет.

Холографија

Холографија е техника за снимање на модели со која се произведува тридимензионален објект наречен *холограм*. Холограмите се добиваат со помош на ласерска светлина и плоча премачкана со холографска емулзија, но тие не можат да дадат тридимензионална слика сами од себе – неопходно им е осветлување на адекватен начин. Кога светлината паѓа врз холограм, таа се рефлектира и се расфрла што создава 3Д слика на оригиналниот објект. Кога холограмот се гледа од разни агли се добива впечаток дека тој се гледа од друга перспектива. Холографија може да даде и холограм на објект во движење.

Холограм може да се види на возачките дозволи, на личните карти, на кредитните картички или на CD, на DVD и на софтверските пакувања. За жал, овие холограми не се многу импресивни. Можете да забележите промени во боја и во форма кога ги движите напред – назад, но тие обично изгледаат како светливи сликички.

За разлика од другите 3Д технологии, холограмот не проектира слика на некоја површина (платно или слично). Набљудувачот има впечаток како да гледа 3Д фотографија некаде во воздух.

Холограмите имаат голем број практични примени. Научниците можат да ги употребат холограмите за проучување објекти или за прикажување модели во три димензии, може да се користат во мобилните телефони и во компјутерите. Холографската

меморија, исто така, станува практичен метод на зачувување на голем број податоци на многу мали меморији.

Сл. 1. 60 Холографска тастатура

Сл. 1. 61 Холографски телефони

Сл. 1. 62 Примена на холограмите

Сл. 1. 63 „Виртуелно присуство“ засновано на холографија која интегрира компјутерски генерирана графика во реална средина.

Препорака:

Погледни го интервјуто направено со помош на холографија: http://www.youtube.com/watch?v=js6b31_p5cc

Во блиска иднина треба да се очекуваат подобрени способности за препознавање на говор и примена на холографска 3Д технологија која ќе овозможи тридимензионални искуства без користење на специјални очила.

Сл. 1. 64 Мобилен телефон од иднината

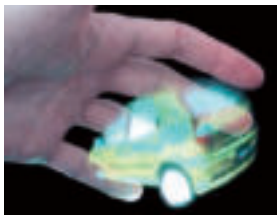
За љубопитните:

Холографијата е откриена во 1947 година од страна на унгарско-британскиот физичар Денис Габор и за нејзиното откривање овој научник ја добил Нобеловата награда во 1971 година. Првиот трансмисиски холограм на 3Д објект добиен со ласер бил воз играчка и птица.

Хелиодисплеј технологија

Хелиодисплеј (heliodisplay) е уред кој проектира слики во боја во тенок слој на воздух со користење на нова револуционерна технологија потполно различна од холографија. Тоа е како уред за прикажување на слики, но без екран. Извор на сигнал може да биде визуелна опрема или компјутер. Овие слики овозможуваат полна интеракција – селектирање, навигација и манипулација со движења на рака или прст, слично како на екраните чувствителни на допир.

Се очекува оваа технологија да најде примена на саемите, телеконференциите, рецепциите, во музеите и во луксузните канцеларии.



Сл. 1. 65 Хелиодисплеј слика создадена во воздух

Сл. 1. 66 Интерактивна хелиодисплеј слика

Сл. 1. 67 3Д хелиодисплеј слика

Резиме

Во *технологија со допир* се препознава дел од чувствителната површина на која се врши допир. Екранот чувствителен на допир овозможува внесување податоци преку допир на соодветно место на екранот. Некогаш е потребен електронски молив за внесување на податоци. Технологија на повеќекратен допир е технологија во која се препознаваат прсти или рака на корисникот при преоѓање преку површина покриена со сензори. Технологија без допир овозможува извршување на едноставни функции со движења на дланка над одредена површина.

3Д слика на некој објект се добива така што специјална камера го снима објектот од две перспективи, потоа сликите се прикажуваат со специјални проектори со што се постигнува чувство на длабочина.

Виртуелна реалност е технологија со која реалната околина се заменува со графички приказ на виртуелната околина во облик на 3Д слика или анимација која е генерирана на компјутерот.

Холографија е техника за снимање на модели со која се произведува тридимензионален објект наречен холограм. За разлика од другите 3Д технологии, холограмот не проектира слика на некоја површина.

Хелиодисплеј е уред кој проектира слики во боја во тенок слој на воздухот. Овие слики овозможуваат полна интеракција.

Прашања:

1. Наброј неколку видови на современите технологии!
2. Објасни ги технологиите засновани на допир!
3. Која е разлика помеѓу технологија на допир и технологија на повеќекратен допир?
4. Како функционира технологијата без допир?
5. Каде наоѓа примена технологијата без допир?
6. Што е 3Д технологија и каде се применува?
7. Што е виртуелна реалност?
8. Наброј неколку влезни и неколку излезни единици за виртуелна реалност!
9. Што е холограм? Направи споредба на холограм со слика од 3Д технологија!
10. Што е хелиодисплеј?
11. Каква е разликата помеѓу слики добиени со холографија и со хелиодисплеј технологија?

СОФТВЕР

Клучни зборови

- Системски софтвер
- Оперативен систем
- Услужни програми
- Управувачки програми
- Комуникациски софтвер
- Програмски систем
- Апликативен софтвер
- Модул
- Јадро
- Кориснички интерфејс
- Драјвери
- Датотека
- Папка (директориум)
- Хиерархиски систем на организација
- Екстензија
- Патека
- Архивирање
- Компресија
- Злонамерен софтвер
- Антивирус
- Антивирусна програма
- Слободен софтвер
- Пробна верзија
- Софтвер со отворен код
- Лиценциран софтвер

2.1 Софтвер: системски и апликативен

Потсети се!

Важен дел на компјутерскиот систем претставуваат и програмите. Сите програми инсталирани во компјутер сочинуваат програмски дел или софтвер (software).

За компјутерот да може да изврши некоја задача, корисникот мора да му даде инструкции кои операции да ги изведе и по кој редослед. Корисникот инструкциите ги задава преку соодветни програми. Значи, компјутерската програма содржи низа од инструкции дадени на компјутерот.

Програмите ги контролираат сите процеси во компјутерот од почетокот до крајот. Софтвер (software) е множество од програми врз основа на кои хардверот извршува одредени задачи. Функционалноста на еден компјутерски систем зависи од квалитетот на хардверот, но многу повеќе зависи од квалитетот на софтверот. Со извршување на различни програми, компјутерот извршува различни задачи што го прави да биде машина од општа намена.

За љубопитните:

Поимот софтвер прв пат е употребен од страна на John W. Tukey, инженер по информатика, 1957. година. Поимот е настанат како аналогија со поимот хардвер. На англиски хардвер означува цврсти делови од компјутерот кои можат да се допрат и да се видат. За разлика од хардверот кој поретко се менува, софтвер се менува често и лесно. Зборот софтвер потекнува од зборот soft што значи меко, односно може да се менува.

Софтверот не може да работи без хардвер, како што ни хардверот не може да работи без софтвер. Софтверот обично се наоѓа на хард дискот од каде се вчитува во RAM меморија, а оттука се проследува до процесорот кој извршува наредби од некоја програма.

Софтверот може да се подели во две групи:

- системски софтвер и
- апликативен (кориснички) софтвер.

2.1.1 Системски софтвер

Системскиот софтвер е неопходен за работа на компјутерот. Тој секогаш е присутен во комјпутерот и овозможува негово полесно, поедноставно и поефикасно користење.

Системските програми се делат на:

- оперативен систем (OS – Operating System)
- услужни (utility) програми
- управувачки (контролни) програми и комуникациски софтвер
- програмски систем

Оперативен систем

Оперативен систем – ОС (Operating System – OS) е множество од програми кои ја контролираат и ја координираат работата на хардверските единици и овозможуваат извршување на кориснички програми. Дел од оперативниот систем секогаш е во внатрешната меморија кога компјутерот е вклучен. Тој прима наредби од корисникот и

го овозможува нивното извршување. ОС се однесува како посредник помеѓу корисникот, програмите и хардверот.

Оперативните системи за персоналните компјутери се разликуваат од оперативните системи за поголемите и помоќните компјутери. На персоналните компјутери денес најмногу се користат различни верзии на Windows (Windows 95, Windows 98, Windows ME (Millennium), Windows NT (New Technology), Windows 2000, Windows XP (Experience) i Windows 2003, Windows 7. Компанија која го произведува овој ОС е Microsoft. Друг, сè попопуларен ОС е Unix и неговата јавно достапна верзија Linux чија предност е во отвореноста – секој може да го види и да го менува својот ОС. Macintosh компјутерите го користат Mac OS X.

Услужни програми

Услужните програми на корисникот му овозможуваат управување со податоци и со хардверските компоненти. Тие извршуваат работи како што се форматирање, чистење и одржување на хард диск, сортирање, копирање и пренесување на датотеки, организирање и наоѓање на податоци, компресија на податоци, инсталација на софтвер, заштита од вируси итн. Овие програми често се вградени во оперативниот систем, но некои од нив се произведуваат и продаваат посебно.

Управувачки програми и комуникациски софтвер

Управувачките програми се грижат програмските инструкции да се пренесат до соодветните хардверски единици и да се извршат (на пр. наредба за печатење). Овие програми често се вградуваат и се испорачуваат заедно со хардверот.

Комуникациските програми овозможуваат комуникација помеѓу компјутери.

Програмски систем

Програмскиот систем го сочинуваат програми кои им помагаат на програмерите како поддршка во изработка на апликативен софтвер. Пред сè, тоа се *едиmори* во кои се пишуваат програми и *програмски преведувачи* (compilers) кои наредбите напишани во некој од програмските јазици ги претвора во бинарни кодови разбирливи за процесорот.

2.1.2 Апликативен софтвер

Апликативниот софтвер го сочинуваат програми кои овозможуваат компјутерот да изврши конкретни задачи за потребите на корисникот. Со други зборови, тоа се програми наменети за корисници и уште се нарекуваат и *кориснички софтвер*. Може да биде деловен софтвер, едукативен софтвер, софтвер за канцелариско работење и друго.

Во апликативните програми спаѓаат:

- обработка на текст (Notepad, WordPad, Word, Writer, Quarck Exspres и др.)
- работа со табели (Excel, Calc, Lotus и др.)
- обработка на слики и цртежи (Paint, Photoshop, Corel Draw, Illustrator и др.)
- работа со бази на податоци (Access, Fox Pro, Oracle и др.)
- изработка на презентации (PowerPoint, Impress, FrontPage, Flash, Dreamweaver и др.)
- обработка на звук и видео записи – мултимедија (Windows Media Centar, Power DVD, BS Player Pro, Winamp и др.)
- пресметки во науката, во техниката и во статистиката (Mathematica и др.)

- работа со Интернет (Internet explorer, Firefox, Opera и др.)
- игри (Counterstrike, Fifa, Splintersell, Hitman, Sims, Barbie и др.)
- и уште многу други.

Често програми кои извршуваат слични работи се групираат во т.н. *програмски пакети*. Најкористените се пакетите за канцелариското работење (Microsoft Office, OpenOffice.org и др.) и програмите од областа на компјутерската графика (Corel Graphic Suite, Adobe и др.). Поголемиот дел од апликациите се пишувани за потребите на просечен корисник, но се пишуваат и програми по нарачка за специфични потреби на корисникот.

Резиме

Софтвер е множество од програми врз основа на кои хардверот извршува задачи. Софтверот може да се подели во две групи: системски и апликативен софтвер.

Системскиот софтвер овозможува полесно, поедноставно и поефикасно користење на компјутерот. Системските програми се делат на: оперативен систем, услужни програми, управувачки програми и програмски систем.

Апликативните програми се пишувани и се користат за извршување на специфични задачи за потреби на корисниците. Често програмите кои извршуваат слични работи се групираат во т.н. *програмски пакети*.

Прашања:

1. Што е софтвер?
2. Од што зависи функционалноста на компјутерскиот систем?
3. Како се дели софтверот?
4. Која е задача на системскиот софтвер во компјутерот?
5. Како се делат системските програми?
6. Што е оперативен систем?
7. Зошто, додека компјутерот е вклучен, дел од оперативниот систем е присутен во внатрешната меморија?
8. Наведи неколку оперативни системи за персоналните компјутери!
9. Наведи неколку работи кои ги извршуваат услужните програми!
10. Кои се задачи на управувачките програми и на комуникациски софтвер?
11. Што е програмски систем?
12. Кои програми сочинуваат апликативен софтвер?
13. Наведи неколку апликативни програмии и објасни за што се користат!
14. Што се програмски пакети? Наведи некој програмски пакет!

2.2 Оперативен систем: улога, структура

Потсети се!

На секој компјутер прво мора да се инсталира оперативен систем, без него ниту еден компјутер не може да работи.

Оперативен систем е основна компонента на програмскиот дел на компјутерот и без него ниту еден компјутер не може да работи. Под поимот оперативен систем се подразбира целокупниот

софтвер кој му е потребен на корисникот за управување со компјутерскиот систем и за извршување на програми кои можат да работат на него.

2.2.1 Улога и функции на оперативниот систем

Улога на ОС е да контролира и да управува со компјутерот со помош на наредбите од корисникот. Тој ги обединува сите делови на компјутерот во складна целина и ги сокрива од корисникот деталите за нивното функционирање со што на корисникот му е олеснето користење на компјутерот. На пример, корисник работи со некоја апликација (пишува текст, слуша музика, пресметува, црта и сл.), апликацијата со која тој работи го користи ОС за да се изврши обработка на податоците на хардверот.

Програмите кои го сочинуваат ОС се групирани во програмски целини наречени *модули*. Секој модул остварува една функција на ОС. Модулите се надополнуваат еден со друг. Постои и модул, наречен јадро, кој координира со сите останати модули и се грижи тие исправно да ја извршат својата функција.

Основните *функции* на оперативниот систем се:

- управување со процесорот
- управување со меморијата
- управување со В/И единици
- управување со податоците

Управување со процесорот подразбира решавање на следниве прашања: кој процес да се изврши, колку време му е потребно на процесорот да изврши одреден процес, колку процеси можат да чекаат на извршување и сл.

Управување со меморијата подразбира грижа за ефикасно користење на оперативната меморија. Модулите кои ја реализираат оваа функција водат сметка колку и кои делови на меморијата се слободни или зафатени и како правилно да се распредели расположивата меморија на секој процес.

Управување со В/И единици се однесува на целокупната работа на влезните и излезните единици. Модули задолжени за оваа функција одлучуваат кога и кои единици можат да се доделат на поедини задачи и процеси.

Управување со податоците подразбира ракување со податоци, начин на чување на податоци на надворешните мемории, пристап до податоците, запишување и читање на податоците, креирање, чување и бришење на датотеки и друго.

2.2.2 Структура на ОС

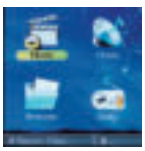
Денешните ОС имаат во поголема или во помала мерка идентична основна структура. Структурата на сите ОС ја сочинуваат: *јадро* како најважната компонента, *драјвери*, *програмски алати* и *кориснички интерфејс*.

Јадро (kernel) е основа на ОС, всушност тоа е модул кој ги надгледува и ги контролира другите модули и се обидува тие оптимално да ги извршуваат своите функции. Заради оваа важна функција, јадрото често се поистоветува со оперативниот систем иако тој претставува само една негова компонента.

Драјвери за уреди (device drivers) се специјализирани програми кои овозможуваат користење на одредени уреди (картички, печатач, скенер и др.). Тие можат да бидат дел од јадрото на оперативниот систем, дел од друга програма, или двете. За поедноставните и за стандардните уреди драјверите се испорачуваат со самиот ОС, додека за останатите уреди тие се испорачуваат самостојно и не се дел од ОС.

Програмски алати имаат улога на корисниците да им обезбедат пристап до функциите на ОС. Пример се програмите за навигација по системот на датотеки, како и програми кои овозможуваат основна обработка на податоци.

Оперативните системи се испорачуваат со т.н. *кориснички интерфејс* (работно опкружување) преку кој се ракува со оперативниот систем. Корисничкиот интерфејс не влијае на можностите на ОС, но е од голема важност за корисникот.



Најосновната поделба на корисничките интерфејси е на *текстуални* и на *графички*. Текстуален кориснички интерфејс подразбира употреба на тастатура за пишување и за задавање на наредби. DOS е пример на ОС со текстуален кориснички интерфејс. Графичките кориснички интерфејси (Graphic User Interface – GUI) се појавиле подоцна (во 70-тите години на минатиот век) и тие се засноваат на комуникација со корисникот преку знаци и слики и користење глувче за избор на наредби кои треба да се извршат. Графичките кориснички интерфејси се поедноставни за користење, но им недостасува флексибилност кога е во прашање користењето на компјутерите. Од овие причини сите модерни ОС овозможуваат истовремено користење на двата вида на корисничките интерфејси.

Резиме

Под поимот *оперативен систем* се подразбира целокупниот софтвер кој му е потребен на корисникот за управување со компјутерскиот систем и за извршување на програми кои можат да работат на него.

Улога на ОС е да контролира и да управува со компјутерот со помош на наредбите од корисникот. Основни функции на оперативниот систем се: управување со процесорот, управување со меморијата, управување со В/И единици и управување со податоците.

Структурата на ОС ја сочинуваат: кориснички интерфејс, драјвери, програмски алати и јадро.

Јадро е модул кој ги надгледува и ги контролира другите модули. Драјверите овозможуваат користење на одредени уреди. Програмските алати на корисниците им обезбедуваат пристап до функциите на ОС. Преку корисничкиот интерфејс се ракува со оперативниот систем. Постојат текстуални и графички кориснички интерфејси.

Текстуален кориснички интерфејс подразбира употреба на тастатура за пишување и за задавање на наредби. Графичките кориснички интерфејси се засноваат на користење глувче за избор на наредби кои треба да се извршат.

Прашања:

1. Што се подразбира под поимот оперативен систем?
2. Која е улогата на оперативниот систем?
3. Што е модул?
4. Наброј ги и објасни ги функциите на оперативниот систем!

5. Како се нарекува модулот кој координира со сите останати модули?
6. Кои компоненти ја сочинуваат структурата на оперативниот систем?
7. Што е јадро?
8. Што се драјвери?
9. Што е кориснички интерфејс?
10. Направи разлика помеѓу текстуален и графички кориснички интерфејс!

2.3 Систем на организација на податоците

2.3.1 Поим за датотека

Корисниците на компјутерите секојдневно работат со слики, со текстови, со табели итн. Сите овие се податоци кои компјутерот ги обработува со помош на некоја апликација. За понатамошно користење на овие податоци потребно е тие да се зачуваат на некоја од надворешните мемории. За оперативниот систем да знае како да ги зачува податоците, на кое место на дискот, како да им пристапи и многу други работи, потребно е да постои соодветен систем на организација на податоци. Оваа задача ја презема *системот на датотеки* (File System).

При тоа од голема важност е систематизацијата на податоци, односно групирањето на податоците во целини кои се нарекуваат *датотеки* (files). Датотеките на корисниците им овозможуваат своите податоци да ги организираат според нивните потреби.

Датотека е логичка целина од податоци кои имаат некоја конкретна смисла на употреба.

Тип на датотека

Во зависност од тоа какви податоци се зачувани во датотеките постојат повеќе *типови на датотеки*. На пример, сите податоци врзани за текст (самиот текст, фонтот, уредувањата итн.) се содржани во текстуална датотека. Датотеките во кои се зачувани слики или цртежи и податоци врзани за нив се нарекуваат графички датотеки, датотеките со музика се аудио датотеки итн. Ова е само воопштена поделба на датотеки. Постојат повеќе типови на сите овие датотеки.

Име на датотека

Секоја датотека има свое *име*. Името на датотека се состои од два дела. Првиот дел е име кое корисникот го доделува (*корисничко име*) и кое укажува на конкретната содржина, а вториот дел е *наставка* или *екстензија* која укажува на типот на датотеката. Корисничкото име и типот на датотеката се запишуваат разделени со точка. Тоа изгледа вака:

именадатотека.наставка

Корисничкото име на датотеката може да се состои од најмногу 256 знаци. Името на датотека не може да ги содржи специјалните знаци, (на пр. “_ ; . : / \ * ? < >).

Наставка или екстензија е различна за различни типови на датотеки и се состои најчесто од три знака. Во следната табела се дадени некои типови датотеки и нивните наставки.

Екстензија	Програми во кои датотеката може да се обработува или да се прегледа	Опис на датотека
.doc	MS Word	Текстуална датотека
.odt	Writer	Текстуална датотека
.txt	MS Word, Notepad	Текстуална датотека
.xls	MS Excel	Текстуална датотека – табела
.ods	Calc	Текстуална датотека – табела
.pdf	Acrobat Reader	Текстуална датотека
.xtm .html	Internet Explorer, Front page и др.	Интернет страница
.jpg .bmp	CorelDraw, Photo Shop, Windows picture and fax viewer, ACDSee и др.	Графичка датотека
.cdr	CorelDraw	Графичка датотека
.psd	Photo Shop	Графичка датотека
.mp3	Windows Media Player, Winmap и др.	Аудио датотека
.exe		Извршна датотека

Tabela 2. 1. Екстензии за некои типови датотеки

2.3.2 Систем на датотеки

Следната задача на системот на датотеки е групирање на датотеките во една организациона целина. За таа цел просторот на надворешните мемории се претставува како низа од помали целини во кои можат да се чуваат податоци. Компјутерот може да има повеќе диск единици, хард дискови, CD и DVD единици.Хард дискот прво мора да се форматира, односно да се подели на логички целини наречени *партиции*.

Во Windows на секоја партиција ОС и доделува *ознака*, на пр. C:, D:, E: итн. (ознаките A: и B: се резервирани за диск единици).

Во Edubuntu партиции се креираат како посебни виртуелни папки (директориуми).

Забелешка:

Хард дисковите на денешните компјутери веќе се форматирани. Партиција на која е зачуван оперативниот систем се нарекува системска партиција.

Папка (фолдер, директориум, каталог)

Датотеките на надворешните мемории се организирани на различни начини во зависност од оперативниот систем. Најчесто се користи *хиерархиски систем на организација* на датотеки во кој датотеките се организирани во целини наречени *папки* (фолдери). Тоа е имагинарна папка во која се чуваат датотеки или други папки – потпапки. Папка, всушност, е посебен вид на датотека која содржи список на датотеки и на потпапки кои ѝ припаѓаат. Папките имаат име, но немаат наставка.

Единица на надворешна меморија (или една нејзина партиција) се смета за основна папка наречена и *корен* (root directory). Сите останати папки се наоѓаат во коренската папка. На потполно празна надворешна меморија се наоѓа само основната папка – корен (root).

Во основната папка се креираат потпапки – папки од првото ниво. Со креирање на нивните потпапки се добиваат папки од второто ниво итн. На тој начин се добива *хиерархиска структура на дискот*. Хиерархиската структура на папки и потпапки потсетува на дрво (directory tree) (сл. 2.1).

Патека, полно име и работна папка

Секоја датотека или папка на хард дискот или на друга надворешна меморија единствено е одредена со своето име и со патеката (path) по која се доаѓа до неа.

Патеката е определена со имињата на папките преку кои корисникот треба да се движи од основната папка за да дојде до саканата датотека или папка. Имињата на папките кои се дел од патеката се одделуваат со знакот „\“ кај ОС Windows или со знакот „/“ кај ОС Ubuntu. Кај графичките оперативни системи тоа се прави со отварање на соодветните папки една по една.

Полно име на датотека или папка се состои од патеката и од името на датотеката односно папката. Патеката започнува со коренската папка која во Windows се означува со ознака на партиција и обратна коса црта (\) (на пример C:\ или D:\), а во Ubuntu само со коса црта (/).

Пр. 2. 3. Во хиерархиската организација (прикажана на сл. 2.1), за да се дојде до папката Ana треба да се отвори папката Informatika потоа папката I godina и на крај папката I-1.

Полното име на папката Ana е:

D:\Informatika\I godina\I-1\Ana

на ОС Windows, односно

D:/Informatika/I godina/I-1/Ana

на ОС Ubuntu.

Полното име на датотеката Test-1 е

D:\Informatika\I godina\I-1\Ana\Test-1.doc

на ОС Windows, односно

D:/Informatika/I godina/I-1/Ana/Test-1.doc

на ОС Ubuntu.

Сл. 2. 1 Хиерархиски систем на организација

Управување со папки и датотеки

Системот на датотеки располага и со основните алати за работа со датотеки – движење низ папките, креирање нови датотеки и папки, бришење, преместување, копирање, преименување датотеки и папки, прикажување на содржината на папките, прикажување на дрвото на организација итн.

Програмите за управување со папки и датотеки се File manager во Windows и Nautilus во Edubuntu. Овие програми овозможуваат да се организираат датотеки во папки и врз нив да се извршуваат други активности како што се:

- креирање и прикажување на датотеки и папки
- пребарување и класифицирање на датотеки
- пристап на компјутерот до локална мрежа (работа во компјутерско мрежно опкружување)
- снимање податоци на CD или DVD.

Резиме

За организација на податоците на дискот се грижи *системот на датотеки*. Податоците се групираат во датотеки. *Датотека* е логичка целина од податоци кои имаат некоја конкретна смисла на употреба.

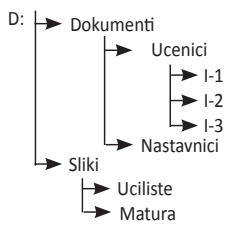
Името на датотеката се состои од два дела: корисничко име и наставка или екстензија која укажува на типот на датотеката.

Папка е посебен вид на датотека која содржи список на датотеки и на потпапки кои ѝ припаѓаат. Единицата на надворешната меморија е основна папка во која се наоѓаат сите останати папки. Ваков начин на организација на папки се нарекува *хиерархиска структура*.

Секоја датотека или папка на хард дискот или на друга надворешна меморија е единствено одредена со своето име и со патеката кои заедно го сочинуваат нејзиното полно име. Програмите за управување со папки и датотеки овозможуваат манипулирање со датотеки и со папки.

Прашања:

1. Како се организираат податоци на хард дискот?
2. Што е датотека?
3. Со што е одреден типот на датотеката?
4. Од кои делови е составено полното име на датотека?
5. Како се запишува име на датотека?
6. Што е корисничко име, а што е тип на датотека?
7. Наведи неколку типови на датотеки!
8. Што е партиција?
9. Што е папка?
10. Што е основна папка?
11. Колку папки има на празен диск?
12. Како се нарекува системот по кој се организираат датотеки и папки на дискот?
13. Напиши го полното име на папката Boris од сликата 2.1.
14. Реализирај хиерархиска структура како што е прикажано со шемата!



2.4 Архивирање и компресирање на датотеки

Дури и на најсовремените компјутерски системи се случуваат грешки кои доведуваат до губење податоци (невнимателно бришење на датотеки, оштетување на хард диск и сл.). Ова може да биде непријатно кога се работи за важни податоци за чие креирање е вложено многу време и труд. За да се спречат несакани ситуации треба редовно да се прават резервни копии на важните податоци. Во овој процес голема улога имаат *архивирање и компресија на податоци*.

2.4.1 Архивирање на податоци

Архива е множество од датотеки кои се спакувани заедно во една датотека со цел правење на резервна копија (backup), пренесување на друг компјутер, чување на друго место, праќање преку електронска пошта, поставување на сервер и слично. Постои голем број на јавно достапни програмски пакети за архивирање кои се едноставни за користење. Програмите кои вршат архивирање на датотеки најчесто вршат и компресија на податоци иако тоа не е неопходно за да се креира архива.

2.4.2 Компресија на податоци

Надворешните мемории на кои се чуваат архивирани датотеки имаат ограничен капацитет, затоа е важно датотеките да заземаат што помалку простор.

Намалување на датотеките е важно и за работа преку мрежи заради брзината на преносот. Процесот на намалување на потребниот физички простор за зачувување на податоци со користење на различни методи на нивно запишување се нарекува *компресија на податоци*. *Експанзија* или *декомпресија* е инверзна операција која има за цел да ја врати оригиналната содржина на датотеките.

Со компресија се постигнува:

- заштеда на простор на надворешните мемории и
- заштеда на време при пренесување на податоци преку мрежите

Постојат два основни вида на компресија:

- компресија *без губење на податоци* – декомпресија дава податоци идентични на оригиналот
- компресија *со губење на дел од податоци* – со декомпресија не се добиваат оригинални податоци но степенот на компресија е голем

Од типот на датотеките зависи кој вид на компресија ќе се примени и колкаво ќе биде намалувањето. Доколку се врши компресија без загуби најголемата заштеда се постигнува кај текстуалните датотеки. Кај графичките и кај музичките датотеки компресијата е многу помала, додека кај видео датотеките заштеда е незначителна. Кај овие датотеки се врши компресија со загуби, а најпознатите типови на компримирани датотеки се .jpg за графички датотеки, .mpg за видео датотеки и .mp3 за аудио датотеки.

За љубопитните:

Некои датотеки заземаат поголем простор отколку што тоа е потребно. Најдобар пример се текстуалните датотеки кои се чуваат по принципот еден знак во еден бајт меморија. За кодирање на еден знак се потребни 7 битови (ASCII), а еден бајт е составен од 8 битови, па доаѓа до непотребно користење на меморија. Се дошло до заклучок дека знаците можат да се кодираат и со помали групи битови. На пример ако податокот AAAAAAAAA се зачува како 9A тој наместо 9 бајти би заземал само 3 бајта што е голема заштеда. Со ваквите техники текстуалните датотеки можат да заземаат и до 50% помалку меморија.

2.4.3 Програми за архивирање и компресија во MS Windows

На компјутерите со ОС Windows најпознатите програми за архивирање и за компресија на податоци се: ARJ, ZIP и RAR.

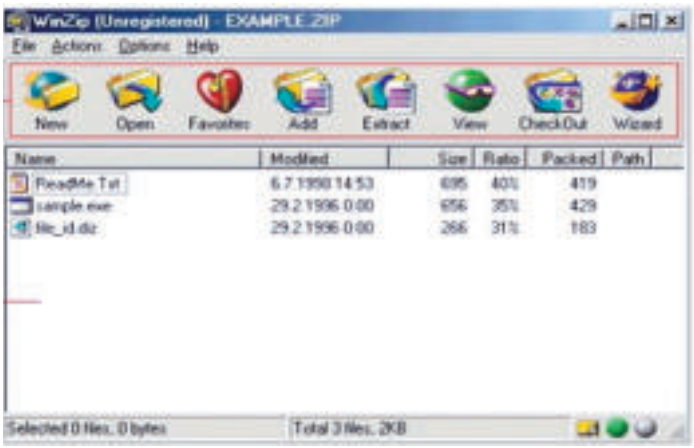
ARJ овозможува чување на една или на повеќе датотеки во компримиран формат. Компримираната датотека има наставка .arj.

ZIP датотеки се архиви кои се користат за дистрибуирање и чување на датотеки. Датотеките кои се архивирани во ZIP се компримирани заради заштеда на простор. ZIP датотеките овозможуваат полесен пренос на група датотеки.

WinRAR е моќна алатка за компресија. RAR датотеките во просек имаат 8 до 15 проценти поголема компресија од ARJ и ZIP датотеките. WinRAR ги подржува сите познати типови на компримирани датотеки, па може да ги отпакува и ARJ и ZIP архивите.

Програмата WinZip

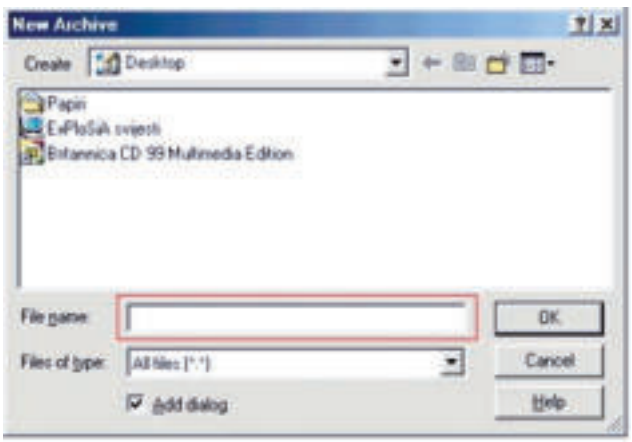
WinZip е едноставна и популарна програма за компресирање и за архивирање на датотеки. WinZip овозможува брз пренос на податоци (повеќе датотеки наеднаш) и архивирање на податоци кои не се користат но кои можеби еден ден ќе ни бидат потребни.



Сл. 2. 2 Прозорец на програмата WinZIP

Нова ZIP датотека може да се креира со кликнување на копчето *New* или со повикување наредбата *File→New archive*. Ќе се појави прозорецот *New archive*.

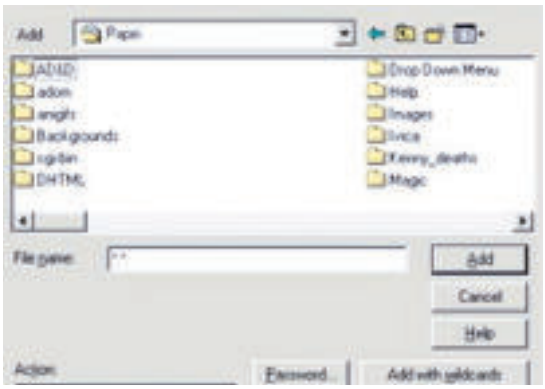
Во лентата *File name* се запишува име на архива која се креира (не име на датотека која се компресира). Потоа се кликнува на копчето *OK.*



Сл. 2. 3 Прозорец во кој се креира нова датотека

Се појавува нов прозорец во кој се бираат и означуваат датотеки кои ќе се компресираат и додадат во архивата.

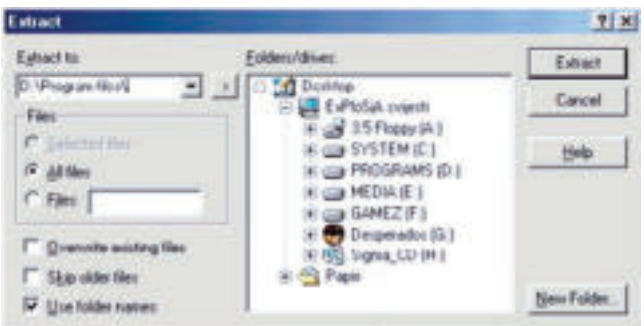
Поедноставен начин е во Windows Explorer да се означат саканите датотетеки и со десното копче да се избере *Add to archive*.



Сл. 2. 4 Прозорец за избор на датотеки за архивирање

Постоечка ZIP датотека може да се отвори со кликнување на копчето *Open* (сл. 2.2) или со повикување на наредбата *File→Open archive*. Ќе се добие сличен прозорец како при креирање на нова датотека, сега во полето *File Name* се запишува името на датотеката која се отвора.

За да се отпакува ZIP датотека на компјутер, се притиска копчето *Extract* (сл. 2.2) или се повикува наредбата *Actions→Extract*. Во прозорецот не смее да биде означена ниту една датотека од архивата затоа што тогаш само таа датотека ќе се ископира на компјутерот. Ќе се отвори прозорецот *Extract* во кој, во полето *Extract to* се пишува име на папка во која ќе се отпакува архивата.

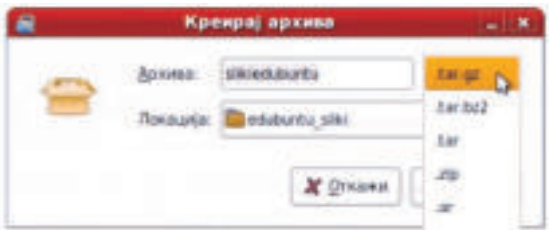


Сл. 2. 5 Прозорец за отпакување на архива

2.4.4 Програми за архивирање и за компресија во Edubuntu ОС

Во ОС Edubuntu датотеките можат да се пакуваат (компресираат) во различни формати. Програмата gzip (GNU ZIP) врши компресија на податоци потоа се креира датотека со наставка .gz. Програмата gzip може да врши компресија но не и архивирање. Секоја датотека мора посебно да се компресира, а доколку е потребно да се креира компресирана датотека во која се наоѓа поголем број на датотеки, gzip се користи заедно со програмата tar која врши архивирање и при тоа се креира нова датотека со наставка .tar.gz.

За креирање на архиви се користи наредбата *Креирај архива,* по што се добива прозорецот *Креирај архива*. Во полето *Архива* се запишува име на архива која се креира, во полето *Локација* се избира папка во која ќе се креира архива и се избира тип на архива (најчесто tr.gz).



Сл. 2. 6 Архивирање и компресирање на податоци во Edubuntu

Намалените датотеки можат да се вратат во оригинален облик со помош на програмата gunzip, односно gzip -d. Содржината на компримираните датотеки може да се прегледа со програмите zcat и zless без претходно да се направи декомпресија.

Повеќето архиви можат да се отворат директно или со програмата *Менаџер на архиви*. За да се отпакува архива се кликнува со десното копче на архивата:

- се избира *Отпакувај тука* за да се отпакуваат датотеките во тековниот директориум,
- за поголема контрола од менито се избира *Отвори со Менаџер на архиви.*

Сл. 2. 7 Отворање на архивирана датотека во Edubuntu

Резиме

Архива е множество од датотеки кои се спакувани заедно во една датотека.

Процесот на намалување на потребниот физички простор за зачувување на податоци со користење на различни методи на нивното запишување се нарекува *компресија на податоци*. Експанзија или *декомпресија* е инверзна операција која има цел да ја врати оригиналната содржина на датотеките.

Постојат два основни вида на компресија: компресија без губење на податоци и компресија со губење на дел од податоци.

На компјутерите со Windows најпознати програми за архивирање и за компресија на податоци се: ARJ, ZIP и RAR.

На компјутерите со Edubuntu најпозната програма за архивирање е Tar, а за компресија на податоците Gzip. Овие две програми најчесто се користат заедно.

Прашања:

1. Зошто е потребно да се врши архивирање на податоци?
2. Што опфаќа процесот на архивирање на податоци?
3. Дали е неопходно податоците да се компримираат пред тие да се архивираат?
4. Што се постигнува со процесот на компресија на податоци?
5. Кои се основни видови на компресија на податоци?

6. Како може да се врати оригиналната содржина на компримирана датотека?
7. Кај која од следните датотеки ќе се постигне поголем степен на компресија: Tekst.doc или Slika.bmp?
8. Наведи неколку типови на компримирани датотеки!
9. Наведи неколку програми за архивирање и компресија на податоци!
10. Една датотека архивирај со различни програми, податоците запиши ги во табела (користи програма за табеларно пресметување). Спореди го степенот на компресија!

2.5 Злонамерен софтвер и заштита од него

Дали сите програми се корисни и добронамерни? Дали некогаш ти се случило твојот компјутер да се однесува чудно, да работи бавно, не реагира на команди и слично? Постои голем број на штетни и злонамерни програми, затоа треба да се биде претпазлив и да се внимава какви програми се инсталираат на компјутерите.

Компјутерските саботери, наречени хакери (hackers), користат злонамерен софтвер (malware) кој се стартува на компјутерот на корисникот и прави компјутерот да работи она што напаѓачот сака. Хакерите можат неовластеното пристапување во компјутер да го искористат за да заработат, да украдат, да нанесат штета на сопственикот со бришење на податоци, да располагаат со доверливи податоци и сл.

2.5.1 Видови на злонамерен софтвер

Во злонамерен софтвер спаѓаат вируси, црви, спам пораки итн.

Вируси

Вируси се злонамерни програми кои ја користат секоја заразена програма или диск за понатаму да се шират со правење свои копии. Тоа се остварува така што вирусот се прикачува за некоја програма (извршна датотека најчесто од тип .exe или .com) и се извршува секој пат кога се извршува и програмата. Тие обично се сокриваат во оперативниот систем или во апликативните програми на корисникот. Некои вируси не работат ништо освен што се репродуцираат, некои прикажуваат пораки на екранот, додека некои можат да избришат податоци.

Вирусите обично се направени за одреден оперативен систем и ги напаѓаат дисковите на кои тој ОС е инсталиран. Исклучок се макро вируси кои се прикачуваат на документи кои содржат макро наредби (вметнати мали програми за автоматизација на задачи), најчесто Microsoft Office документи. Макро вирусите можат да се шират преку email прилози (attach). Овие вируси се нарекуваат email вируси.

За љубопитните:

Еден од најпознатите email вируси е вирусот Melissa од 1999. година. Овој вирус работел како повеќето email вируси: корисник прима „важна порака“ од пријател која содржи документ со примамлива содржина. Документот всушност содржи макро вирус и откако корисникот ќе го отвори документот, вирусот веднаш започнува со праќање на заразениот документ на првите 50 адреси од адресарот на корисникот. На овој начин вирусот Melissa се раширил со огромна брзина – за само неколку дена заразени се околу 100000 Windows системи.

Црви (worms)

Слично како и вирусот, *црв* (worm) е програма или серија од програми кои ги користат компјутерите домаќини за понатаму да се репродуцираат. Но за разлика од вирусите, црвите се самостојни програми и не им е потребна програма за која ќе се прилепат. Црвите се размножуваат со огромна брзина најчесто како додаток на email пораки. Тие имаат способност понатаму да се препратат на речиси сите адреси кои се запишани во адресарот на е-поштата.

За љубопитните:

Првиот црв кој привлекол големо внимание е направен од страна на студент во рамки на експеримент 1988. година. Црвот случајно е пуштен на Интернет и закочил околу 6000 компјутери во САД. Вкупната штета, од загубеното работно време потрошено на чистење на компјутерите во научните институции, била огромна.

Тројански коњи

Тројански коњи (Trojan horse programs) се навидум корисни програми, но тие ја загрозуваат безбедноста и нанесуваат голема штета на компјутерот. Слично со легендата според која и го добиле името, овие програми се претставени како атрактивни програми (игри, корисни алатки и сл.) и корисниците сами ги преземаат од Интернет мислејќи дека доаѓаат од сигурен извор. Кога таква програма ќе се стартува, таа може да избрише датотеки, да измени или да украде податоци или да предизвика друг вид штета. Тројанските коњи можат да се најдат во бесплатен софтвер кој се презема од Интернет. Тие ретко се размножуваат но можат да нанесат големи штети и да предизвикаат големи проблеми во компјутерските системи.

Логичка бомба е вид тројански коњ, тоа е програма која се активира како реакција на некое дејство, на пример ако се внесе специјална шифра, ако се логира одреден корисник или ако се извршат одредени наредби. Ако програмата се активира во одредено време тогаш се нарекува *темпирана бомба*.

Спам пораки

Спам пораките се непосакувани, комерцијални електронски пораки, кои се сместуваат во електронските поштенски сандачиња на корисниците. Овие пораки често се придружени со прикачени документи кои доколку бидат отворени, можат да го заразат компјутерот со вирус. Некои спам пораки ги мамат примачите да ги откријат своите лозинки и други вредни информации што би можело да ја наруши безбедноста на податоците и на работата.

Adware и Spyware

Adware е софтвер кој се појавува на екранот на корисникот во вид на светлечка рекламна порака во текот на извршувањето на друга програма. Овој софтвер редовно го забавува работењето на системот.

Spyware е софтвер кој без дозвола надгледува работа на корисникот и испраќа информации за неговите online активности со цел стекнување на корист за трето лице. Овој софтвер, за разлика од вирусите и црвите, обично не се размножува, туку е дизајниран да ги искористува заразените компјутери за комерцијална добивка. Типично применувани тактики се: активирање на небарани pop-up реклами; крадење на личните податоци (на пр. броеви на кредитните картички или лозинки), набљудување на активности и навики на корисниците на Интернет во рекламни цели и сл.

2.5.2 Заштита од злонамерен софтвер

Опасноста од злонамерниот софтвер веќе секојдневно се зголемува и со самото ширење на Интернетот. Секојдневно се пишуваат стотици нови штетни програми затоа е од голема важност компјутерот соодветно да се заштити од секојдневните закани.

Антивирусни програми

Најдобриот начин да се ослободи од малициозните програми и од хакерите е инсталирање на *антивирусна програма*. Антивирусни програми (AV softver) се проектирани да бараат вируси, да го известат корисникот кога ќе најдат вирус и истиот да го отстранат од заразениот диск, документ или програма.

Сите модерни антивирусни програми имаат неколку компоненти:

- дел за проверка на датотеките (scan)
- дел за чистење на заразените датотеки (clean) и
- дел што секогаш е активен и ги надгледува влезните и излезните операции на компјутерот со цел да спречи евентуално навлегување на вируси (monitor).

Делот за проверка (скенирање) ја проверува содржината на дискот и ги бара вирусите. Доколку се најде вирус автоматски се покренува делот за чистење. Чистењето се врши така што во заразената датотека се брише кодот кој претставува вирус. Некогаш единственото решение е да се избрише цела датотека, што е и најдоброто решение и треба да се применува секогаш кога тоа е можно. Делот за надгледување автоматски се стартува со вклучување на компјутерот.

Иако постојат голем број бесплатни антивирусни програми, мал е бројот на оние кои не го оптоварuваат системот, односно не ја намалуваат неговата брзина. Avast и AVG се примери антивирусни програми кои го задоволуваат ова барање.

Препорака:

Антивирусната програма Avast може да се преземе од веб страницата: www.avast.com. Антивирусната програма AVG може да се преземе од веб страницата: www.grisoft.com.

Меѓу најпопуларните антивирусни програми се вбројуваат и McAfee Virus Scan и Norton AntiVirus, но за разлика од претходно спомнатите антивирусни програми, нивното користење не е бесплатно.

Сепак, не постои антивирус кој може да ги открие сите вируси и тие постојано мора да ја надополнуваат својата база на познати вируси како би можеле да ги најдат најновите вируси. Денес поголемиот дел на антивирусни програми автоматски ги преземаат програмските дополнувања од Интернетот (update). Многу антивирусни програми можеш да ги обновиш преку веб страница на нивниот производител.

За намалување на ризикот од заразување на компјутерот со злонамерен софтвер се препорачува:

- софтверот да се набавува по легален пат,
- да се избегнува размена на датотеки со непознати корисници,
- антивирусните програми редовно да се надополнуваат,
- да не се отвора електронска пошта од непознати корисници,
- при секоја размена на датотеки да се провери дали истата е заразена.

Огнениот ѕид

Антивирусите не се совршени, тие ги откриваат само оние вируси кои им се познати и се наоѓаат во нивната база, така што нови вируси, особено тројански коњи, можат да поминат, а антивирусот да не ги открие. Најдобрата заштита од тројанските коњи е *огнениот ѕид* (firewall). Тоа е програма или хардверски уред кој ги надгледува сите податоци што компјутерот или локалната мрежа ги праќа или ги прима преку Интернет и нема да им дозволи на сомнителните програми да поминат. За исправно работење на огнениот ѕид, потребно е прецизно да се одреди низа од правила кои одредуваат што претставува дозволен сообраќај, а што забранет сообраќај. Денес многу антивирусни програми даваат и firewall заштита.

Совет:

Кога на компјутерот се пријавуваш со администраторски права го зголемуваш безбедносниот ризик. Црвите, тројанците и другите штетни програми би добиле поширок пристап до системот, а со тоа и поголема можност да направат штета. Затоа во редовна работа пријавувај се како корисник со ограничени права.

Резиме

Во *злонамерен софтвер* спаѓаат вируси, црви, тројански коњи, spyware, adware, spam итн. Вирусите се штетни програми кои се прикачуваат на некоја програма за понатаму да се шират со правење на свои копии. Црв е програма која ги користи компјутерите домаќини за понатаму да се репродуцира. Тројански коњи се навидум корисни програми кои паралелно со корисна работа нанесуваат штета на компјутерот. Спам пораки се непосакувани, комерцијални електронски пораки кои се сместуваат во електронските поштенски сандачиња на корисниците. Adware е софтвер кој се појавува на екранот на корисникот во вид на светлечка рекламна порака во текот на извршувањето на друга програма. Spyware е софтвер кој без дозвола ја надгледува работа на корисникот и испраќа информации за неговите online активности со цел стекнување на корист на трето лице.

Најдобриот начин да се ослободи од малициозните програми и од хакерите е инсталирање на *антивирусна програма*. *Огнениот ѕид* е програма или хардверски уред кој ги надгледува сите податоци што компјутерот или локалната мрежа ги праќа или ги прима преку Интернет.

Прашања:

1. Што е злонамерен софтвер?
2. Со која цел хакерите користат злонамерен софтвер?
3. Кои се видовите на злонамерен софтвер?
4. Која е разлика помеѓу вирусите и црвите?
5. Што се тројански коњи?
6. Како се нарекуваат несаканите електронски пораки?
7. Што е adware, а што е spyware?
8. Како можеш да се заштитиш од злонамерен софтвер?
9. За кои антивирусни програми си слушнал/а? Дали ти користиш некоја антивирусна програма?
10. Кои компоненти ги имаат сите современи антивирусни програми?
11. Опиши го начинот на работа на антивирусните програми!
12. Што е огнен ѕид?

2.6 Слободен софтвер, пробна верзија, лиценциран софтвер

Денес производство на програми е огромна индустрија во целиот свет која е во голем пораст. Секоја компјутерска програма доаѓа со *лиценца* која на корисникот му дозволува одредени права на користење. Производувачот со лиценцата ја заштитува својата програма од копирање и од злоупотреба. Најчесто користените лиценци се:

- сопственичка (proprietary) комерцијална лиценца,
- „делен“ (пробен) софтвер (shareware),
- јавен (public domain software) софтвер,
- слободен софтвер (freeware),
- софтвер со отворен код (open source).

Сопственичка (proprietary) комерцијална лиценца се употребува за деловна примена. Производувачот дава право на користење на софтвер за паричен надомест. Софтверот може да се користи под точно утврдени услови додека сопственоста секогаш останува кај производувачот. Купувачот добива и соодветна документација и можност да се регистрира кај производувачот како подоцна би можел да добива нови верзии на програмата и би имал поддршка од производувачот. Примери на софтвер со сопственичка лиценца се Microsoft Office, Adobe Photoshop, Corel Draw и др.

„Делен“ (пробен) софтвер (shareware) е софтвер со лиценца според која корисникот може да го употребува софтверот определено време, одреден број пати или со ограничени можности и да се запознае со неговите можности. По истекување на пробниот период корисникот треба да купи комерцијална лиценца инаку повеќе не може да го користи. *Демо верзија* (demo) е еден вид покажувачка програма при што не функционираат сите опции. *Пробна верзија* (*trail*) е верзија на програмата при што таа работи комплетно, но по одредено време престанува со работа. Програмите со оваа лиценца воглавно се дистрибуираат преку download од веб-локацијата на производувачот. Познати shareware производи се WinZip, многу антивирусни програми како Eset Nod32 и други.

Јавен (public domain software) софтвер се дистрибуира бесплатно и може слободно да се копира и разменува. Корисниците обично немаат поддршка од производувачот.

Слободен софтвер (*freeware*) е софтвер со лиценца според која на корисникот му се овозможува потполно бесплатно користење на софтверот, но често под некој услов, на пример користење во некомерцијални цели или за домашна употреба. Производувачите на софтвер понекогаш поставуваат ограничени верзии (ограниченост на нивоа во игри, ограничен број на изработени документи и сл.) на својот софтвер како freeware верзија со цел да ги покажат можностите на својот главен производ. Примери на freeware програми се Internet Explorer, 7-zip, Adobe Reader и антивирусните програми AVG free edition и Avast home.

Софтвер со отворен код (*open source*) е софтвер со кој доаѓа и неговиот изворен код. Сите лиценци за софтвер со отворен код дозволуваат промени во кодот и понатамошна дистрибуција. Така софтверот е бесплатен за сите и сопственоста не е поставена од страна на поединец или претпријатие. GPL, LGPL и BSD се типови на open source лиценци. Проекти со ваква лиценца се Linux, OpenOffice и Apache web server.

Постојат и други видови софтвер: *adware* – софтвер кој е бесплатен за корисникот но е финансиран од реклами, *abandonware* – софтвер чиј производувач повеќе не постои или не е поддржан од страна на производувачот, *приватен софтвер* – софтвер развиен по нарачка и според тоа нема примена за други цели.

Програмите често се копираат и се дистрибуираат без знаење на авторот (т.н. *пиратски софтвер*). Се поставува прашање за авторските права и надомест на штета на произведувачите. Во многу земји ова е уредено со Закон. Продавање и купување, како и поседување на пиратски софтвер подлежи на кривично гонење, според Законот за интелектуална сопственост, во кој се предвидени големи казни за прекршителите.

Резиме

Секоја компјутерска програма доаѓа со лиценца која на корисникот му дозволува одредени права на користење. Произведувачот со лиценца ја заштитува својата програма од копирање и од злоупотреба.

Сопственичка, комерцијална лиценца се употребува за деловна примена. Произведувачот дава право на користење на софтвер за паричен надомест.

Делен софтвер е софтвер со лиценца според која корисникот може да го употребува софтверот одредено време, одреден број пати или со ограничени можности за да се запознае со неговите можности.

Јавниот софтвер се дистрибуира бесплатно и може слободно да се копира и разменува.

Слободен софтвер е софтвер со лиценца според која на корисникот му се овозможува потполно бесплатно користење на софтверот, но често под некој услов, на пример користење во некомерцијални цели или за домашна употреба.

Софтвер со отворен код е софтвер со кој доаѓа и неговиот изворен код и се дозволени промени во кодот и понатамошна дистрибуција.

Продавање и купување, како и поседување на пиратски софтвер подлежи на кривично гонење, според Законот за интелектуална сопственост, во кој се предвидени големи казни за прекршителите.

Прашања:

1. Набroј ги видовите на софтвер и накратко опиши ги!
2. Наведи примери на програми со различни видови лиценци?
3. Која е разликата помеѓу слободен софтвер и софтвер со отворен код?
4. Што е пиратски софтвер?
5. Дали со Закон е уредено правото на користење на програмите?
6. Што мислиш, дали компјутерските програми треба да бидат бесплатни? Разговарај со некој од соучениците кои имаат спротивно мислење од твоето. Наведи аргументи за твоето мислење.

ПРОГРАМИ ЗА ОБРАБОТКА НА ТЕКСТ

Клучни зборови

- Стил
- Вграден стил
- Прилагоден стил
- Наслов
- Поднаслов
- Содржина
- Индекс
- Маркирање
- Образец
- Волшебник за образец
- Форма
- Поле за контрола на текст
- Копче за избор на можност
- Паѓачко мени
- Лозинка

3.1 Програми за обработка на текст

Програмите за обработка на текст овозможуваат пишување, уредување и печатење писма, статии, книги и други текстови. Карактеристика на овие програми е тоа што внесувањето текст е одделено од печатењето, така што секој текст пред да се отпечати може да се провери и да се отстранат грешките. Освен тоа, секој текст може дополнително да се измени и да му се определи изглед по желба. Текстот се внесува во работната површина на местото на покажувачот. Кога ќе се пополни еден ред, покажувачот поминува во нов ред. Копчето *Enter* се притиска само кога се започнува нов пасус.

При работа со текст може да се промени името на фонтот, големината, бојата на буквите, стилот на буквите (задебелени, искосени и/или подвлечени) и друго. На пасусите може да се промени проред на линии, простор пред и по пасусот, вовлекување/испакнување на првата линија во пасусот и друго. Текстот може да се уреди во колони. Во документ може да се вметнат и да се уредат слики или други графички објекти. Во текстот можат да се вметнат коментари и фусноти и да се постават врски кон други делови од текст или кон други документи. Исто така, можат да се креираат листи со набројување или листи со знаци, да се вметнат и да се уредат табели, може да се пребарува текст или еден дел од текст да се замени со друг текст. На страница може да се постави ориентација, големина, маргини, може да се вметне заглавие и подножје, да се нумерираат страници и друго.

Програмите за работа со текстуални датотеки нудат уште многу можности. Во програмите за обработка на текст може, на пример, сите наслови да се уредат на ист начин или може да се додаде табела со содржина во документ со неколку клика. Доколку правилно се искористат овие и други можности документите можат брзо и лесно да се сработат и тие ќе имаат добар изглед.

Потсети се!

Јазичната поддршка се менува со кликнување на иконата за јазична поддршка која обично се наоѓа на десната страна на лентата со задачи. За поддршка на македонска кирилица треба да се избере ознаката МК. Кратенка: Alt+Shift.

Ќе се запознаеме со две најчесто користени програми за обработка на текст. Тоа се Microsft Word 2010 (или само Word) и OpenOffice.org Writer (или само Writer). Документ изработен во Word ја има наставката .doc. Документ изработен во Writer ја има наставката .odt.

Важно!

Кога е можно содржините за Word и за Writer ќе ги пишуваме заедно, при што наредбите и опциите ќе ги одделиме со знакот „/“. Ако има многу разлики, содржините ќе ги обработиме во посебни поднаслови.

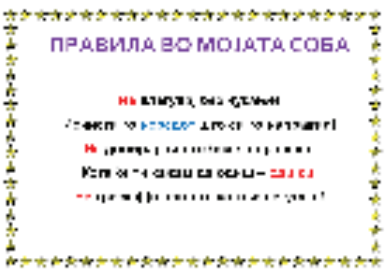
Задачи за повторување:

1. Направи постер со правила за однесување во твојата соба кој можеш да го закачиш на врата. Следи ги упатствата:
   - Отвори нов документ;
   - Постави ориентацијата на страница да биде легната (*Landscape / Пејзаж*);
   - Постави маргини (сите 4 см);

- Смени ја јазичната поддршка во МК и внеси текст, на пример листа на правила како во примерот подолу:

ПРАВИЛА ВО МОЈАТА СОБА
Не влегувај без чукање!
Исчисти го нередот што си го направил!
Не допирај ништо без да прашаш!
Кога ќе ти кажам да одиш – оди си!
Не трескај ја вратата кога излегуваш!

- Постави го насловот во средина, смени го стилот, големината на буквите (48 точки) и бојата. Остави простор по насловот (60 точки / 3 см);
- Уреди го и останатиот текст: смени го фонтот, стилот, големината (28 точки), бојата на буквите, израмни го текстот во средина и по секој пасус остави простор (12 точки / 0,6 см);
- Вметни рамка по твој избор околу страницата;



Сл. 3. 1 Постерот „Правила во мојата соба“

- Погледни ја страницата пред да ја отпечатиш во погледот *Print Preview/ Преглед на страница*. Доколку сакаш направи измени во уредувањето, на пример големината на фонтот, порамнувањето, стилот на буквите;
- Зачувај го и отпечати го постерот.

2. Најди на интернет текст на пр. за несакани пораки во е-пошта. Ископирај го текстот во нов документ.
   - Целиот текст, освен насловот, уреди го со големина на буквите 12, израмнет од двете страни;
   - Насловот уреди го со големина на фонт 14 точки и со задебелени букви, израмнет од десната страна;
   - По насловот остави простор 18 точки/1 см;
   - Сите пасуси уреди ги на следниов начин:
     · првата линија вовлечи ја за 1см,
     · проред 1,5 линија,
     · пред сите пасуси постави простор 3 точки/0,2 см,
     · по сите пасуси постави простор 6 точки/0,5 см.
   - Постави фуснота на зборот е-пошта (во насловот); Како објаснување напиши „Електронска пошта“;
   - Страницата уреди ја на следниот начин:
     · формат на страница B5,
     · горната и долната маргина по 2,5 см, левата и десната маргина по 2 см.

- Во заглавието напиши го твоето име и презиме;
- Во подножјето вметни нумерација на страници;
- Зачувај го документот.

3.2 Работа со стилови

Програмите Word и Writer користат *стилови* со што се овозможува креирање на документи со професионален изглед. ***Стил претставува множество од карактеристики за уредување како што се фонт, големина на букви, боја, израмнување, проред итн.*** Користење на стилови овозможува брзо и лесно уредување на текст во документите и ја намалува можноста за грешки, особено кога се работи за големи документи.

Сл. 3. 1 Документ уреден со примена на стилови

Во програмите Word и Writer веќе постојат дефинирани стилови за обичен текст, за наслови, за набројувања итн. Овие стилови можат да се изменат или да се направат нови сопствени стилови.

Видови на стилови

Постојат неколку видови на стилови: стилови на пасус, стилови на знак, поврзани стилови, стилови на табела и стилови на листа.

Стиловите на знаци содржат карактеристики кои можат да се применат на текст, на пример фонт, големина, боја, подвлекување и слични уредувања кои инаку можат да се постават во прозорецот *Format Font*. Тие не ги вклучуваат уредувањата на пасус.

Стиловите на пасус содржат карактеристики кои ги содржат и стиловите за знаци, но ги опфаќаат и уредувањата на пасус како што се проред, растојание пред и по пасус, вовлекување на првата линија и други уредувања кои инаку можат да се регулираат во прозорецот Format Paragraph.

Некои стилови содржат уредувања на знаци и уредувања на пасуси и се нарекуваат *поврзани стилови*.

Стилови на листи содржат уредувања на листи со броеви или со знаци како што се избор на ознака за листа, стил на знакот, растојание на ознаката од левата маргина и слично.

Стиловите на табели содржат уредување на табели како што се рабови, боја на ќелии, ширина на редови итн.

3.2.1 Стилови во MS Word 2010

Потсети се!

Фонт се уредува со алатките од групата Font на рибонот Home, а повеќе уредувања се дадени во прозорецот Font.

Пасус се уредува со алатките од групата Paragraph на рибонот Home, а повеќе уредувања се дадени во прозорецот Paragraph.

Во MS Word 2010 на картичката *Home*, во групата *Styles*, се наоѓа галерија со брзи стилови. Името на секој стил е прикажано токму онака како што тој стил изгледа, односно е уреден. На пример:

1 текстот автоматски се обликува со основниот стил *Normal*,

2 за насловите и за поднасловите се дефинирани стилови за неколку нивоа на наслови (*Heading 1, Heading 2* итн.),

3 за текстот кој треба да се нагласи дефиниран е стилот *Strong*.

Сл. 3. 2 Галерија со брзи стилови

Дополнителните информации за секој стил можат да се најдат во *листа со стилови* која се отвора со кликнување на копчето за отворање на дијалог прозорец во групата *Styles* (кратенка: *Ctrl+Alt+Shift+S*).

Сл. 3. 3 Отворање на листа со стилови

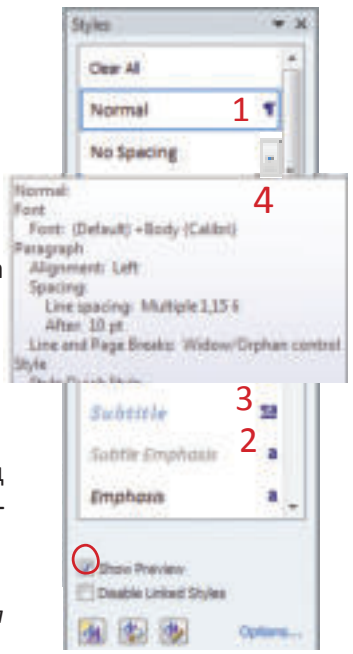
Од десната страна до името на секој стил се наоѓа ознака која кажува од кој вид е стилот (сликата десно):

1 Стиловите на пасуси се означени со ознаката ¶;

2 Стиловите на знаци се означени со ознаката *a*;

3 Поврзаните стилови се означени со ознаката *¶a;*

4 Кога со глувчето ќе се доближи до името на некој од стиловите, се појавуваат информации како стилот е уреден – фонт, големина, порамнување, вовлекувања итн.



Сл. 3. 4 Листа со стилови и карактеристики на стил

Забелешка:

Изгледот на стилот се гледа доколку во листата е потврдена опцијата Show Preview.

Примена на вградените стилови

Стиловите кои веќе се дефинирани и се достапни во листата со стилови се нарекуваат *вградени стилови*. На сите нови документи им се придружува стилот *Normal*. Но, каде е потребно тоа, може да се примени и некој друг стил.

За да се примени стил на пасус се кликнува каде било во пасус или се означуваат повеќе пасуси, потоа се кликнува на саканиот стил во галеријата или во листата со стилови.

За да се примени стил на знак се означува дел од текст, потоа се кликнува на саканиот стил во галеријата или во листата со стилови.

Поврзаните стилови можат да се применат на пасуси (параграфи) или само на зборови.

За да се внесе текст со некој стил, прво се избира стил па се внесува текст.

Чекор по чекор:

1. Отвори или напиши документ од неколку пасуси во кој има барем еден наслов и еден поднаслов.
2. Постави го покажувачот каде било во насловот, во галеријата со стилови кликни на стилот *Heading 1*.
3. Постави го покажувачот каде било во поднасловот, во галеријата со стилови кликни на стилот *Heading 2*.
4. Означи текст кој сакаш да биде истакнат, во галеријата кликни на *Strong*.
5. Зачувај го документот.

Повеќе:

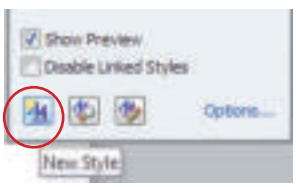
Во Word постојат множества на брзи стилови и теми на бои и фонтови кои се комбинираат со дефинираните стилови и овозможуваат креирање на документи со професионален изглед. Множествата од брзи стилови и темите се прикажуваат со кликнување на копчето Change Styles во групата Styles. На картичката Page Layout се наоѓа картичката Themes од која можат да се изберат теми кои претставуваат комбинации на теми од бои, теми од фонтови и теми од графички елементи.

Креирање на прилагодени стилови

Доколку е потребно уредување кое не е достапно во вградените стилови, може да се креира сопствен стил. Стиловите креирани од корисникот се нарекуваат *прилагодени стилови*. За креирање на прилагодени стилови постојат повеќе начини.

Првиот начин на креирање на прилагоден стил

Во листата со стилови се кликнува на копчето *New Style* по што се отвора прозорецот *Create New Style from Formatting...*



Сл. 3. 5 Копче за креирање на нови стилови

Во прозорецот *Style from Formatting...*:

1 Во лентата *Name* се пишува името на новиот стил;

2 Во лентата *Style type* се избира видот на стилот (за пасус, за знак, поврзан стил итн.);

3 Во лентата *Based on* се избира стилот врз основа на кој новиот стил ќе се базира;

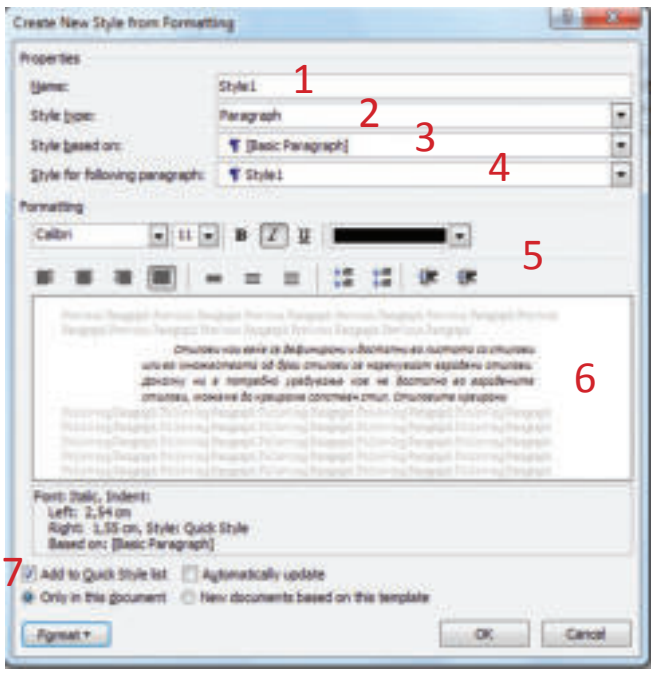
4 Во лентата *Style for following paragraph* (не е достапно за стил на знак) се избира стилот кој ќе следува по стилот кој се креира;

5 Во групата *Formatting* се поставуваат уредувањата на стилот;

6 Изгледот на новиот стил се гледа во делот веднаш под алатките за уредување;

7 Со потврдување на копчето *Add to Quick Style List* стилот ќе се прикаже во галеријата и во листата со стилови;

8 Повеќе уредувања се добиваат со кликнување на копчето *Format* по што се добива листа од која можат да се изберат уредувања на повеќе елементи.



Сл. 3. 6 Прозорец за уредување на карактеристики на нов стил

Сл. 3. 7 Листа за повеќе уредувања

Забелешка:

Сите избори не се овозможени за различни видови на стилови.

Чекор по чекор:

Креирај нов поврзан стил со име Podnaslov базиран врз стилот *Heading 1*.

1. Во листата со стилови кликни на копчето *New Style*;
2. Во прозорецот *Create New Style from Formatting* во лентата *Name* напиши „Podnaslov“;
3. Во лентата *Style type* избери *Linked*.;
4. Во лентата *Based on* избери *Heading 1*;
5. Во лентата *Style for following paragraph* избери *Normal*;
6. Постави ги следниве уредувања: фонт Arial, големина 12 точки, задебелени букви, израмнување во средина;
7. Кликни на копчето *Format* и избери *Paragraph*;
8. Во прозорецот *Paragraph* уреди простор пред пасус 12 точки и простор по пасус 6 точки;
9. Кликни на копчето *OK* во двата прозорци;
10. Примени го стилот Podnaslov во некој документ.

Вториот начин на креирање на прилагоден стил

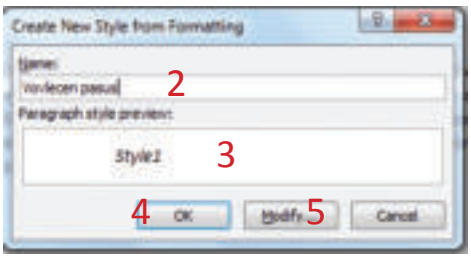
Се пишува текст и се уредува по желба, потоа:

1 Во галеријата на стилови се кликнува на надолната стрелка и се избира *Set Selection as a New Quicke Style*;

2 Ќе се отвори прозорецот *Create New Style for Formatting* во кој се запишува името на новиот стил во лентата *Name*;

3 Во лентата *Paragraph Style Preview* се прикажува изгледот на пасусот;

4 За стилот да се прикаже во галеријата со стилови како поврзан стил се кликнува на копчето *ОК*;



Сл. 3. 8 Креирање на нов стил

5 За да се креира стил на знак или стил на пасус, се кликнува на копчето *Modify* по што прозорецот *Create New Style from Formatting* ќе се прошири. Понатаму се постапува како во претходниот начин на креирање на нов стил.

Чекор по чекор:

Креирај нов поврзан стил со име „Citat“.

1. Напиши текст и уреди го со искосени букви со големина 11 точки;
2. Пасусот вовлечи го од левата и од десната страна за 1см;
3. Постави единечен проред;
4. Кликни во пасусот;
5. Отвори ја галеријата со стилови и кликни на *Set Selection as a New Quicke Style*;
6. Во прозорецот *Create New Style for Formatting*, во лентата *Name*, напиши „Citat“;
7. Кликни на копчето *ОК*.

Чекор по чекор:

Креирај нов стил за знак со име „Istaknato“!

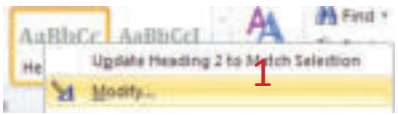
1. Напиши текст и уреди го: фонт *Arial*, задебелени букви, големина 11 точки и темносина боја;
2. Селектирај го текстот;
3. Отвори ја галеријата со стилови;
4. Кликни на *Set Selection as a New Quicke Style*;
5. Во прозорецот *Create New Style for Formatting*, во лентата *Name* напиши „Istaknato“;
6. Кликни на копчето *Modify*;
7. Во лентата *Style type* избери *Character*;
8. Кликни на копчето *ОК*.

Измена на стилови

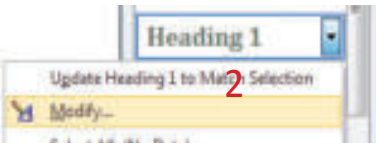
Вградените и прилагодените стилови можат да се изменат според нашите желби и потреби. При тоа текстот на кој стилот е применет автоматски ќе се ажурира. Тоа може

да се направи на два начини:

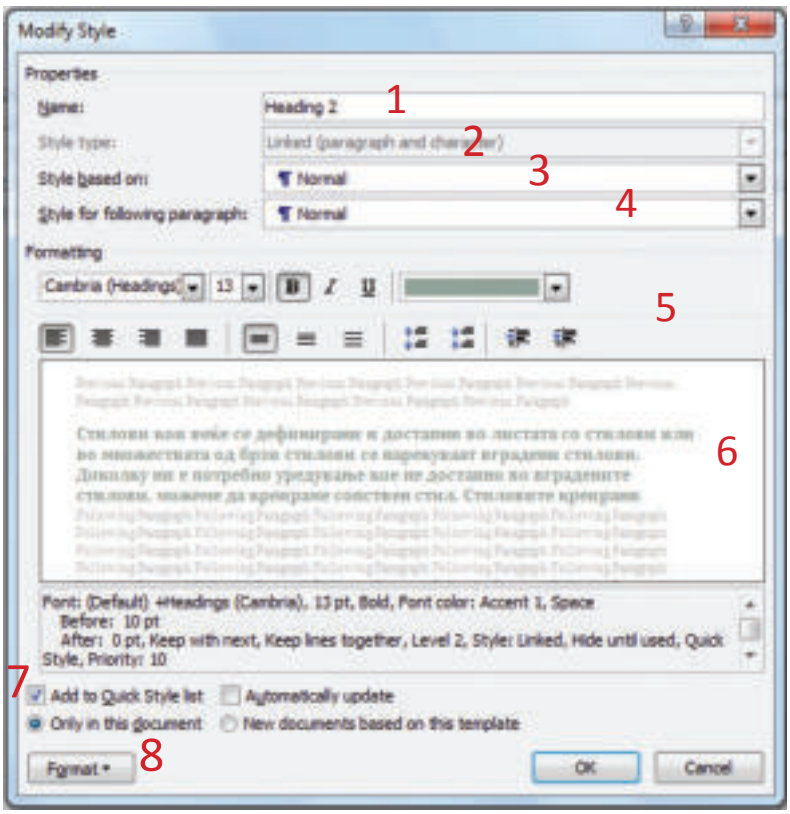
1 Во галеријата на стилови се кликнува со десното копче од глувчето на името на стилот и од листата се избира *Modify*....



2 Во листата со стилови се доближува со глувчето до името на стилот кој се менува по што ќе се појави стрелка; Со кликнување на стрелката се добива листа од која се избира *Modify*...;



И со двата начини се отвора прозорецот *Modify Style*.



Сл. 3. 9 Прозорец за менување на карактеристики на стил

1 Во лентата *Name* е напишано име на стилот;

2 Во лентата *Style type* е прикажан стилот (за пасус, за знак, поврзан стил итн.), овој избор не може да се менува;

3 Во лентата *Based on* се избира стил врз основа на кој новиот стил се базира;

4 Во лентата *Style for following paragraph* (не е достапно за стил на знак) се избира стил кој следува по стилот кој се креира;

5 Во групата *Formatting* се поставуваат уредувања на стилот;

6 Изгледот на новиот стил се гледа во делот веднаш под алатките за уредување;

7 Со потврдување на копчето *Add to Quick Style List* стилот ќе се прикаже во галеријата и во листата со стилови;

8 Повеќе уредувања се добиваат со кликнување на копчето *Format*.

Забелешка:

Постоечките стилови во Word се уредени така што тие соодветствуваат меѓусебно. Уште повеќе, тие се базирани врз некои од основните стилови. На пример, сите стилови се базирани врз стилот Normal. Кога ќе се изменат некои карактеристики на базниот стил, тие ќе се сменат и кај сите стилови кои ги имаат истите карактеристики, а се базирани врз него.

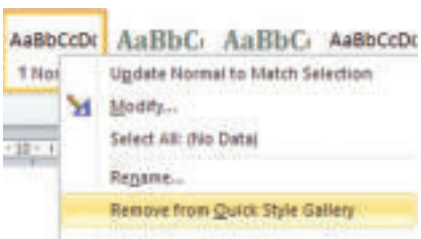
Чекор по чекор:

Измени го стилот *Normal*.

1. Во галеријата на стилови кликни со десното копче на стилот *Normal*;
2. Од листата избери *Modify...*;
3. Во прозорецот *Modify Style* постави ги следниве уредувања: фонт Calibri, големина 11 точки, двострано порамнување, проред 1,5 линија;
4. кликни на копчето *OK*.

Отстранување на стилови од галеријата со стилови

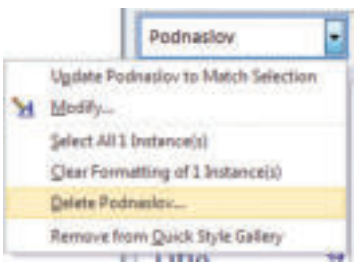
Стиловите кои повеќе не се потребни можат да се отстранат од галеријата со стилови. Во галеријата на стилови со десното копче од глувчето се кликнува на името на стилот и од листата се избира *Remove from Quick Style Galery*.



Сл. 3. 10 Отстранување на стил од галеријата

Бришење на стилови

Стиловите кои повеќе не се потребни можат да се избришат. Во листата или во галеријата со стилови се доближува со глувчето до името на стилот кој треба да се избрише по што ќе се појави стрелка, се кликнува на стрелката па во листата се кликнува на *Delete*;



Сл. 3. 11 Бришење на стил

Забелешка:

Можат да се избришат само прилагодени стилови.

3.2.2 Стилови во Writer

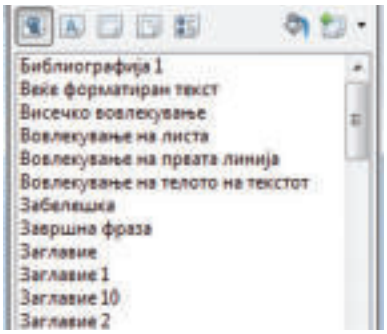
Потсети се!

Фонт се уредува со наредбата Форматирање→Фонт. Пасус се уредува со наредбата Форматирање→Пасус.

Листа со стилови

За да се види листата со стилови се избира наредбата *Форматирање→Стилови и форматирања* или се кликнува на копчето во лентата за форматирање.

Преку копчињата се добиваат различни видови на стилови: на пасус, на знак, на рамка, на станица и на листа.



Сл. 3. 12 Листа со стилови

За да се примени стил треба два пати да се кликне на неговото име. За стил на пасус доволно е претходно да се позиционира во пасусот, додека за примена на стил на знак претходно треба да се означи текстот за кој се применува стил.

Забелешка:

За да се видат сите вградени стилови од паѓачката листа на дното на листата со стилови се избира опцијата ***Сите стилови****.*

Чекор по чекор:

1. Отвори или напиши документ од неколку пасуси во кој има барем еден наслов и еден поднаслов;
2. Постави го покажувачот каде било во во насловот;
3. На лентата за форматирање кликни на копчето за да ја отвориш листата *Стилови и форматирање*, забележуваш дека стилот кој моментално е поставен е *Нормален*;
4. На листата два пати кликни на стилот *Заглавие 1*;
5. Зачувај го документот!

Измена на стилови

За да се измени стил со десното копче се кликнува на името на стилот и од паѓачката листа се избира *Измени...***.** Се отвора прозорец *Стил на знак* или *Стил на пасус*, зависно од тоа кој стил се менува. На прозорецот постојат неколку картички во кои се дадени карактеристики на стилот. Овие карактеристики можат да се изменат и да се постават по желба. Зависно од тоа дали се менува стил на знак, стил на пасус или некој друг вид на стил, се добиваат различни прозорци со картички кои се достапни за тој вид на стил. Во прозорецот *Стил на знак* не можат да се уредуваат карактеристики за пасус.

На картичката Организатор,

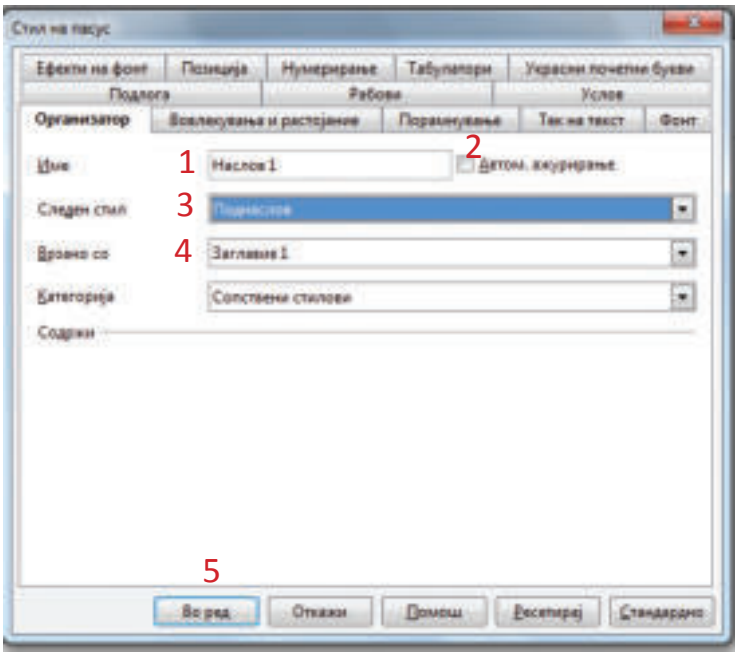
1 Во лентата *Име* се појавува името на стилот чии карактеристики се менуваат;

2 Опцијата *Автоматско ажурирање* се потврдува доколку сакаме измените автоматски да се применат на деловите од документот пишувани со тој стил;

3 Во лентата *Следниот стил* се избира стил што ќе следи по овој стил;

4 Во лентата *Врзано со* е напишано име на стилот врз кој овој стил е базиран;

5 Кога ќе се постават уредувањата, се кликнува на копчето *Во ред*.



Сл. 3. 13 Прозорец за уредување на стил на пасус

Чекор по чекор:

Измени го стилот *Стандардно***.**

1. Отвори ја листата со стилови;
2. Кликни со десното копче од глувчето на стилот *Стандардно,* од паѓачкат листа избери *Измени*;
3. Ќе се отвори прозорецот *Стил на пасус*; Постави уредувања: Arial со големина 11 точки, а пасусот уреди го со двострано порамнување и проред меѓу линии 1,5;
4. Кликни на копчето *Во ред*.

Креирање на нов стил

За да се креира нов стил, со десното копче се кликнува на името на стилот врз основа на кој новиот стил ќе се базира и од паѓачката листа се бира *Нов...*

Се отвора прозорец (*Стил на знак, Стил на пасус* итн.) кој е сличен со прозорецот за менување на стил. Единствената разлика е што во лентата *Име* треба да се напише име на новиот стил.

Чекор по чекор:

Креирај нов стил наречен *Поднаслов* кој се основа врз стилот *Заглавие***.**

1. Ако листата со стилови не е отворена кликни на копчето  на лентата за форматирање;
2. Во листата *Стилови и форматирање* со десното копче од глувчето кликни на стилот *Заглавие 2*;
3. Од паѓачката листа избери *Нов...;*
4. Во лентата *Име* напиши *Поднаслов*;

5. Во лентата *Следен стил* избери *Тело на текст*;
6. Во лентата *Врзано со* веќе е избрано *Заглавие 2*;
7. Кликни на картичката *Фонт*, постави фонт Arial со големина на буквите 14 точки;
8. Кликни на картичката *Вовлекувања и растојанија*; Постави растојание пред пасус 0,5 см, а по пасус 0,3 см;
9. Кликни на картичката *Порамнување*; Избери двострано порамнување;
10. Кликни на копчето *Во ред*.

Бришење на стил

За да се избрише стил, со десното копче се кликнува на името на стилот и од паѓачката листа се избира *Избриши...*

Забелешка:

Можат да се избришат само сопствени стилови.

Чекор по чекор:

Избриши го стилот *Поднаслов*!

1. Ако листата *Стилови и форматирања* не е отворен кликни на копчето  на лентата за форматирање;
2. Со десното копче од глувчето кликни на стилот *Поднаслов*;
3. Од паѓачката листа избери *Избриши...*.

За љубопитните:

Обиди се да креираш стил за табела во кој ќе ги поставиш уредувањата на рабови, боја на ќелии, израмнување на текстот во ќелии итн., како и стил за листа со броеви и листа со знаци во кои ќе ги поставиш уредувањата на листите, на пример симбол на знакот, уредување на знакот, вовлекување на текстот итн.

Резиме

Стил претставува множество од карактеристики за уредување како што се фонт, големина на букви, боја, израмнување, проред итн. Користење на стилови овозможува брзо и лесно уредување на текст во документите.

Постојат неколку видови на стилови: стилови на пасус, стилови на знак, поврзани стилови, стилови на табела и стилови на листа.

Стиловите кои веќе се дефинирани и се достапни се нарекуваат *вградени стилови*. Стиловите креирани од корисникот се нарекуваат *прилагодени стилови*.

Стиловите можат да се менуваат и да се креираат прилагодени стилови. Прилагодените стилови можат да се избришат.

Вештини што треба да ги усовршиш:

Да применуваш стилови.

Да менуваш вградени и да креираш сопствени стилови.

Да отстраниш стилови од галерија на брзи стилови.

Да избришеш стил.

Прашања:

1. Што е стил?
2. Зошто во документите треба да се применуваат стилови?
3. Какви видови на стилови постојат?
4. Како се прикажува листа со стилови?
5. Како се применува стил во документ?
6. Што е вграден, а што е прилагоден стил?
7. Како ќе изградиш прилагоден стил?
8. Како се менува постоечки стил?
9. Како ќе избришеш стил кој веќе не ти е потребен?

Задачи:

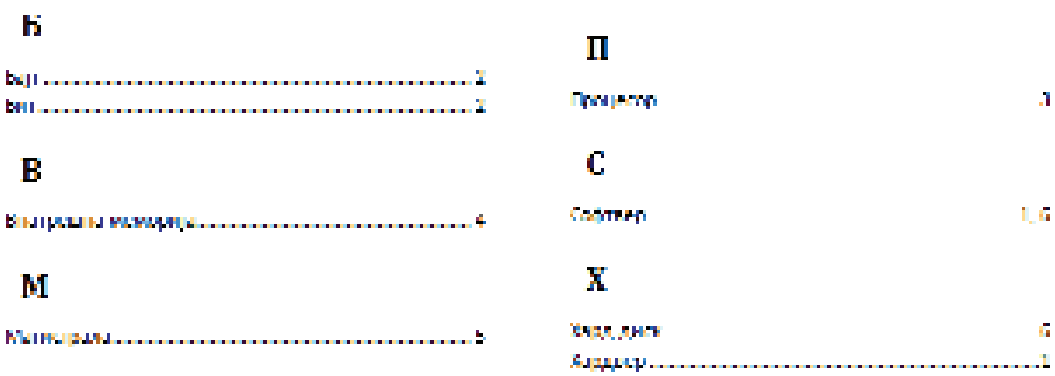
1. Уреди документ (проект по некој предмет) со примена на стилови. Дефинирај сопствени стилови. Спореди го изгледот на документот пред и по примена на стилови? Каков документ е полесно да се уреди: документ на кој се применети стилови или документ на кој не се применети стилови?

3.3 Содржина и индекси

Програмите за обработка на текст овозможуваат автоматско креирање на специјални табели во кои спаѓаат табели на содржини и табели на индекси.

Содржина е листа на наслови на главните теми и на поттеми во некој документ заедно со броеви на страниците на кои тие се наоѓаат. Автоматски изработена табела со содржина овозможува таа лесно да се ажурира како лесно и брзо движење по документот. Насловите и поднасловите во содржината се хиперврски до страниците каде се наоѓаат тие наслови.

На крајот од секој добар учебник или енциклопедија се наоѓа *табела со индекс*, односно список на поими или на теми кои се важни за некој документ, со броеви на страниците на кои тие се наоѓаат, сортиран по азбука. Индексот овозможува лесно и брзо да се најде на која страница се наоѓаат важните поими од книгата.



Сл. 3. 14 Табела со индекс

3.3.1 Табела со содржина во MS Word

Дефинирањето и примената на стилови е од голема помош кога треба да се изработи табела со содржина на некој документ, на пример на семинарска или матурска работа, прирачник, скрипта итн. Прво треба да се испланира концепција на документот. Значи, документот треба да се подели на главни теми, поттеми и сл.

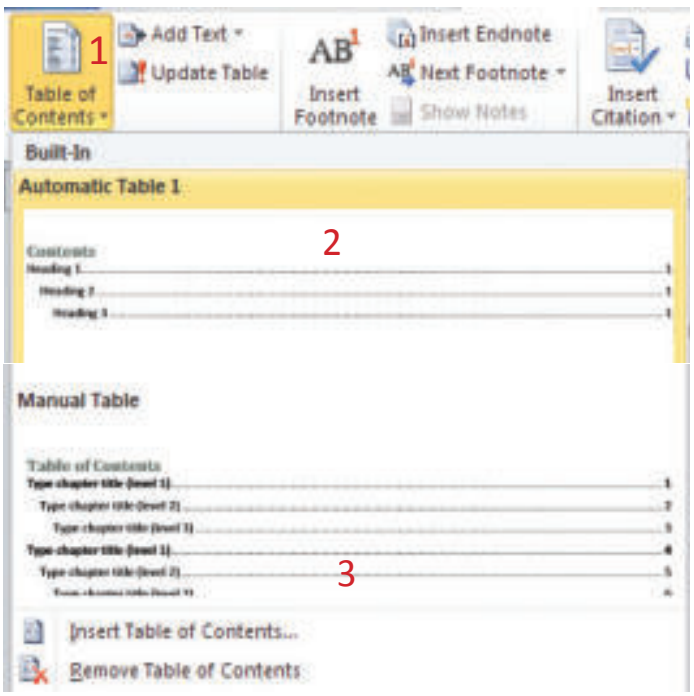
Важно!

Особено е важно на насловите да се применат соодветни стилови (Heading 1, Heading2 итн.). Програмата ги бара насловите и ги вметнува во табелата на содржината. Пред да се почне со внесување на текстот, потребно е страниците да се нумерираат.

Автоматско креирање на табела со содржина

Ms Word 2010 содржи галерија со автоматски стилови на табели со содржина.

Откако ќе се означат елементите на содржината (со примена на стиловите од групата Heading) во документ може да се вметне табела со содржина. Се кликнува на место на кое ќе се вметне табелата (обично на почетокот на документот).



Сл. 3. 15 Вметнување табела на содржина во документ

1 На картичката *References*, во групата *Contents*, се кликнува на T*able of Contents*;

2 Во листата се кликнува на саканиот стил на содржината;

3 За да се наведат повеќе можности, на пр. бројот на прикажаните нивоа на насловите, се кликнува на *Insert Table of Content*.

По ова се отвора прозорецот *Table of Content* во кој:

1 Со вклучување/исклучување на опцијата *Show page number* се избира дали ќе се прикажат редните броеви на страниците;

2 Со вклучување/исклучување на опцијата *Right align page numbers* се избира дали броевите да бидат израмнети од десната страна;

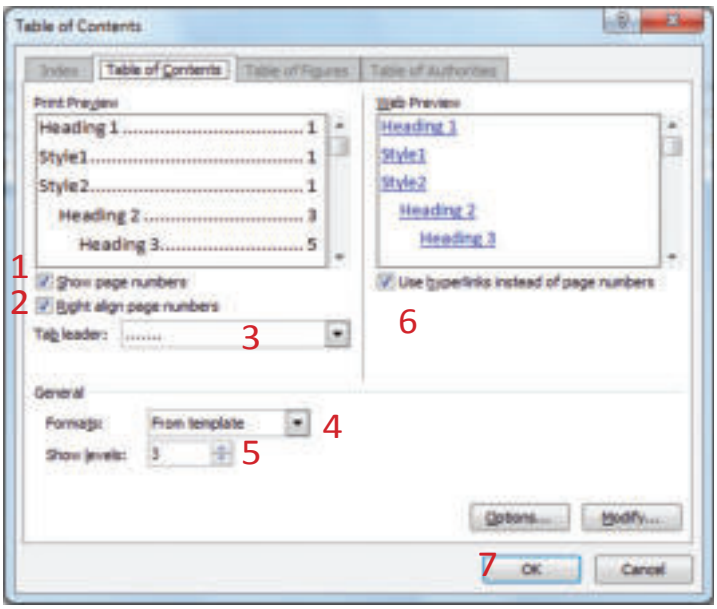
3 Во листата *Tab leader* се избира стил на линија водилка до броевите на страниците (празен простор, линија, точки или цртички);

4 Во полето *Format* се избира еден од форматите на табелата на содржина;

5 Во полето *Show levels* се определува до кое ниво ќе се прикажат насловите;

6 Ако се креира содржина за документ кој ќе се објави на веб-локација елементите на содржината се уредуваат како хиперврски. За таа цел се потврдува опцијата *Use hyperlinks instead of page numbers*;

7 Кога ќе се заврши со поставувањата се кликнува на *OK*.



Сл. 3. 16 Прозорец за поставување параметри на табела на содржина

Чекор по чекор:

1. Отвори го документот на кој веќе се применети стилови за наслови;
2. Позиционирај се на место на кое сакаш да вметнеш содржина;
3. На картичката *References*, во групата *Contents*, кликни на *Table of Contents*;
4. Во листата кликни на *Insert Table of Content*;
5. Потврди ја опцијата *Show page number* за да се прикажат броеви на страници;
6. Потврди ја опцијата *Right align page numbers* за броевите на страниците да бидат израмнети од десно;
7. Во листата *Tab leader* за линии водилки избери цртички;
8. Кликни на копчето *OK*, табелата со содржина ќе се појави во документот;
9. Зачувај го документот со име *Sodrzina*.

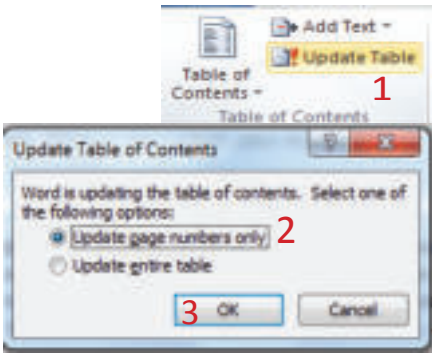
Ажурирање на табела со содржина

Доколку се менува документ кој содржи табела со содржина, тогаш и табелата треба да се ажурира.

1 На картичката *References*, во групата *Contents*, се кликнува на *Update Table* (Кратенка: *F9*);

2 Во прозорецот што ќе се појави се избира *Update pages number only* доколку не е менуван текстот во насловите, во спротивно се избира *Update entire table*;

3 Се кликнува на копчето *OK*.



Сл. 3. 17 Ажурирање на табела со содржина

Чекор по чекор:

1. Отвори го документот *Sodrzina*;
2. Избриши го вториот наслов;
3. Кликни со десното копче во содржината и избери *Update Filed*;
4. Во прозорецот *Update Table of Contents* избери *Update entire table*;
5. Кликни на копчето *OK*.

Бришење на табелата со содржина

Табелата со содржина може да се избрише од документ со наредбата *Remove Table of Contents* од паѓачката листа добиена со кликнување на копчето *Table of Contents*.

Чекор по чекор:

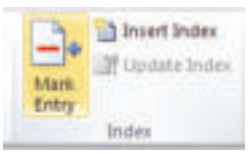
1. Отвори го документот *Sodrzina*;
2. На картичката *References*, во групата *Contents*, кликни на *Table of Contents*,
3. Од паѓачката листа избери *Remove Table of Contents*.

3.3.2 Табела со индекси во Ms Word

Табелата со индекси може да се креира автоматски на сличен начин како и табела со содржина, само претходно треба да се означат (маркираат) поими кои сакаме да ги сместиме во индексот (*елементи на индексот*).

Означување на елементи на индексот

Елемент на индекс се означува така што прво се означува текст (збор, фраза и сл.), потоа во картичката *References* се кликнува на копчето *Mark Entry* (кратенка: *Alt+Shift+X*).



Се кликнува на копчето Mark Entry по што се отвора дијалог прозорецот *Mark Index Entry* (сл. 3.18).

1 Означениот текст се прикажува во полето *Mark entry*;

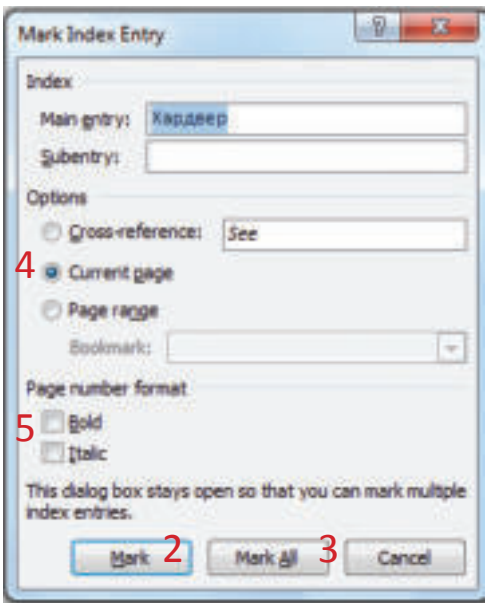
2 Се кликнува на копчето *Mark* со што текстот се означува како елемент на индекс;

3 За да се означи секое појавување на текстот во документот, се кликнува на копчето *Mark All*;

4 Во делот *Options* обично е вклучена опцијата *Current page*, како во индексот со

поимот би се прикажал и бројот на страницата на која тој се наоѓа;

5 Во делот *Page number format* се избира стил на фонт со кој ќе се покаже бројот на страницата (*Bold* и/или *Italic*).



Сл. 3. 18 Прозорец за означување на елементите на индексот

Кога некој елемент ќе се означи, Ms Word во документот додава специјално поле XE (елемент на индекс):

Ова поле е само ознака за означените елементи и спаѓа во ознаки кои не се печатат (како што е и на пр. ознаката за нов ред ¶), чие прикажување на екранот може да се исклучи или вклучи по потреба.

Потсети се!

Вклучување и исклучување на знаците што не се печатат се врши со алатката Show / Hide (¶) од стандардната лента.

Забелешка:

За да се селектира некој друг текст и да се означи како елемент на индекс, прозорецот Mark Index Entry не мора да се затвори. Тој останува отворен и автоматски се освежува кога ќе се селектира некој друг текст.

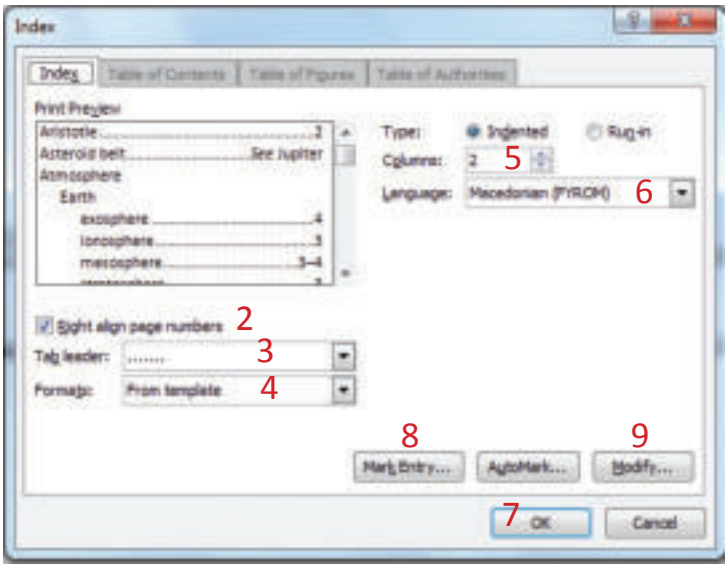
За да се отстрани означување на некој поим, се селектира ознаката XE (заедно со заградите {} и зборот во заградите) и се притиска копчето *Delete*.

Креирање на табела со индекс

Откако ќе се означат елементите се затвора прозорецот Mark Index Entry и може да се креира табела со индекс. Ms Word ги собира елементите, ги сортира по азбучен редослед, ги поставува броевите на страниците каде што тие се наоѓаат, наоѓа и отстранува дупли елементи и го прикажува индексот во документот.

1 Се поставува покажувачот на местото каде сакаме да се појави индекс (обично на крајот од документот) и на картичката *References*, во групата *Index* се кликнува на копчето *Insert Index*.

По ова се отвора прозорецот *Index* во кој:



Сл. 3. 19 Прозорецот за уредување на табела на индекс

2 Доколку се вклучи опцијата *Right align page numbers* броевите на страниците ќе се израмнат десно;

3 Тогаш може да се додадат линии водилки, нивниот стил се избира од лентата *Tab leader*;

4 Од лентата *Formats* се избира изглед на индексот. Може да се избере еден од изгледите *From template, Classic, Fancy, Bulleted, Modern, Formal* и *Simple*;

5 Во полето *Columns* се поставува број на колони;

6 Во лентата *Language* се бира јазик на индексот;

7 Кога сите параметри се поставени се кликнува на копчето *OK* и табела со индекси ќе се појави;

8 Со кликнување на копчето *Mark Entry* може да се означат уште елементи на индексот;

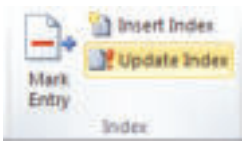
9 Со кликнување на копчето *Modify* може да се сменат карактеристиките на стилот на елементите на индексот.

Чекор по чекор:

1. Отвори документ во кој има текст од неколку страници;
2. Селектирај некој поим (еден или повеќе зборови);
3. На картичката *References*, во групата Index, кликни на копчето *Mark Entry...*;
4. Во прозорецот *Mark Index Entry* во делот *Page number format* означи ја опцијата *Italic*;

5. Кликни на копчето *Mark*;
6. Додека прозорецот е отворен означи уште неколку поими;
7. Кликни на копчето *Close* за да го затвориш прозорецот *Mark Index Entry*;
8. Позиционирај се на последната страница во документот;
9. На картичката *References*, во групата *Index*, кликни на копчето *Insert Index*;
10. Од лентата *Formats* избери *Modern*;
11. Во лентата *Columns* постави го бројот на колоните на 2;
12. Кликни на копчето *OK*;
13. Зачувај го документот со име *Indeks*.

Ажурирање на табелата со индекс



Доколку се означени нови поими како елементи на индексот, или се избришани некои поими кои се наоѓаат во табелата со индекс, тогаш табелата со индексот треба да се ажурира. За тоа е доволно да се позиционира во индексот и на картичката *References*, во групата *Index*, да се кликне на копчето *Update Index* или во табелата да се кликне со десното копче и да се избере наредбата *Update Field*.

Чекор по чекор:

1. Отвори го документот *Indeks*;
2. Одбери уште еден поим и означи го како индекс;
3. Позиционирај се во табелата со индекси;
4. На картичката *References*, во групата *Index*, кликни на копчето *Update Index*;

3.3.3 Табели со содржина и табела со индекс во Writer

Дефинирањето и примената на стилови е од голема помош кога треба да се изработи табела со содржина на некој документ, на пример на семинарска или матурска работа, прирачник, скрипта итн. Прво се планира концепција на документот, односно документот се дели на главни теми, поттеми и сл.

Важно!

Особено е важно на насловите да се применат соодветни стилови (Заглавие 1, Заглавие2 итн.). Програмата ги бара насловите и ги вметнува во табелата на содржината. Пред да се почне со внесување на текстот, потребно е страниците да се нумерираат.

Означување на елементи на индекс

Во табелата со индекс се внесуваат само делови од текст кои претходно се означени како елементи на индекс. Текстот (збор, фраза, израз и сл.) прво се селектира, потоа се повикува наредбата *Внесување→Индекси и Табели→Внес...* или се притиска на копчето на лентата *Вметнување*, по што се отвора прозорецот *Вметни елементи на индекс* (сл. 3.20).

Означениот текст се појавува во полето *Елемент*. Во полето *Индекс* треба да се избере *Алфабетски индекс*. По кликнување на копчето *Вметни*, текстот ќе биде означен како елемент на индекс со што ќе се најде во табелата на индекси.

Сл. 3. 20 Означување на елементи на индекс

Креирање на табела со содржина и на табела со индекс

Табела со содржина и табела со индекс се креираат со наредбата *Вметнување→Индекси и табели→Индекси и табели*, по што се појавува прозорецот *Вметнување индекс/табела*. Во прозорецот *Вметнување индекс/табела*, во картичката *Индекс/табела* во полето *Тип* се избира *Содржина* или *Алфабетски индекс*. Од левата страна е прикажан изгледот на табела со содржината, односно на табелата со индексот.

1 Доколку се креира табела со содржина, во лентата *Наслов* е предложен насловот *Содржина*, а доколку се креира табела со индекс, предложен е насловот *Алфабетски индекс*. Насловите можат да се изменат;

2 Каква табелата ќе се креира се одредува во полето *Тип*;

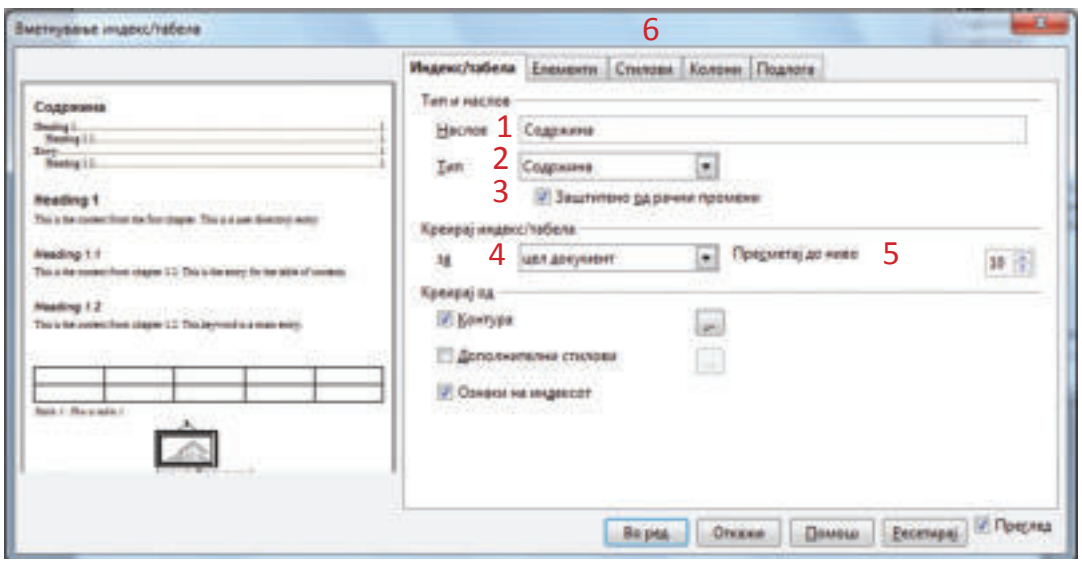
3 Полето *Заштитено од рачни промени* е потврдено што значи дека табела со содржина, односно табелата со индекс не може рачно да се менува. Со отстранување на потврдата се дозволува рачно менување на табелите;

4 Табела може да се креира за цел документ или за поглавје, што се избира во делот *Креирај индекс/табела* во лентата *За*;

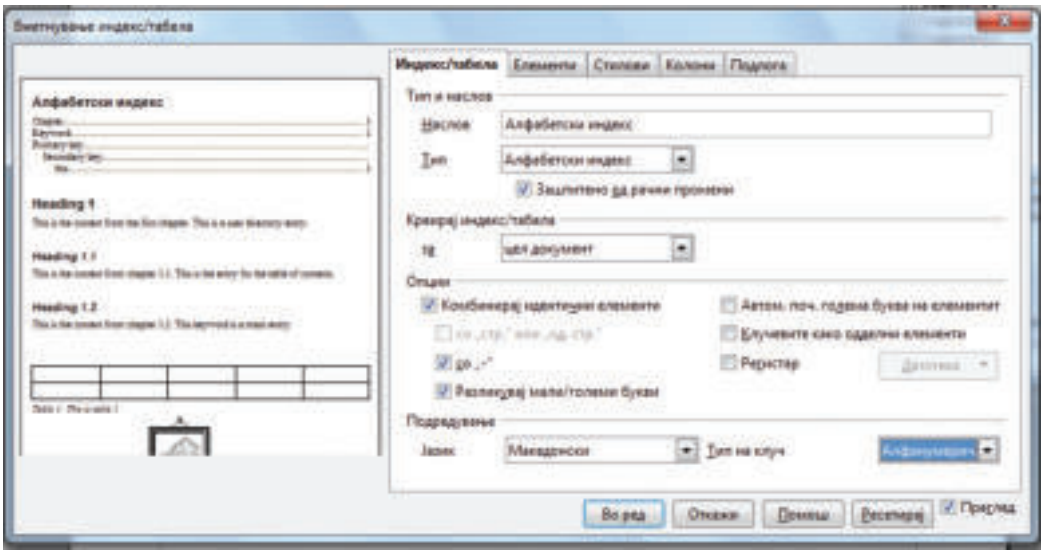
5 Во полето *Пресметај до ниво* се избира до кое ниво наслови ќе се внесат во содржината (оваа опција има само за табела со содржина);

6 Во останатите картички во прозорецот *Вметнување индекс/табела* се поставуваат дополнителни уредувања – *Елементи*, *Стилови*, *Колони* и *Подлога*;

7 Со кликнување на копчето *Во ред* табелата со содржина, односно табелата со индекс се појавува во документот.



Сл. 3. 21 Поставување параметри на табелата со содржина



Сл. 3. 22 Поставување параметри на табелата со индекс

Чекор по чекор:

1. Отвори документот на кој веќе се применети стилови за наслови;
2. Позиционирај се на место на кое сакаш да вметнеш табела со содржина;
3. Од менито *Вметнување* повикај ја наредбата *Индекси и табели→Индекси и табели*;
4. Во прозорецот *Вметнување индекс/табела*, доколку не е отворена, кликни на картичката *Индекс/табела;*
5. Во полето *Тип* избери *Содржина*;
6. Полето *Заштитено од рачни промени* нека остане потврдено;
7. Доколку сакаш, во полето *Наслов* напиши друг наслов;
8. Во делот *Креирај индекс/табела* во лентата *За* избери *цел документ*;
9. Во полето *Пресметај до ниво* избери 3;
10. Кликни на копчето *Во ред*, табелата со содржина ќе се појави во документот;
11. Зачувај го документот со име Sodrzina.

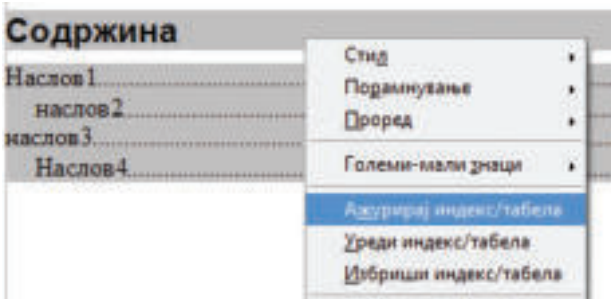
Чекор по чекор:

1. Отвори документ во кој има текст од неколку страници;
2. Селектирај некој поим (еден или повеќе зборови);
3. Од менито *Вметнување* повикај ја наредбата *Индекси и Табели→Внес...*;
4. Во прозорецот *Вметни елементи на индекс* во полето *Индекс* избери *Алфабетски индекс*;
5. Кликни на копчето *Вметни*;
6. Додека прозорецот е отворен селектирај уште неколку поими, кликни на полето *Елемент* па кликни на копчето *Вметни*;
7. Кликни на копчето *Затвори* за да го затвориш прозорецот *Вметни елементи на индекс*;
8. Позиционирај се на последната страница во документот;

9. Од менито *Вметнување* повикај ја наредбата *Индекси и табели→Индекси и табели*;
10. Во прозорецот *Вметнување индекс/табела*, доколку не е отворена, кликни на картичката *Индекс/табела;*
11. Во полето *Тип* избери *Алфабетски индекс*;
12. Кликни на копчето *Во ред*;
13. Зачувај го документот со име Indeks.

Ажурирање на табела со содржината и на табела со индекс

Доколку документ кој содржи табела со содржина или табела со индекс се менува, тогаш и табелата треба да се ажурира. Со десното копче се кликнува каде било во табелата со содржина по што се отвора помошно мени од кое се избира *Ажурирај индекс/табела*.



Чекор по чекор:

1. Отвори го документот *Sodrzina*;
2. Избриши го вториот наслов;
3. Кликни со десното копче во содржината и избери *Ажурирај индекс/табела*.

Резиме

Табела со содржина е листа на наслови на главните теми и поттеми во некој документ заедно со броеви на страниците на кои тие се наоѓат. *Табела со индекс* е список на поими или на теми кои се важни за некој документ, со броеви на страниците на кои тие се наоѓаат, сортиран по азбука.

Вештини што треба да ги усовршиш:

Да креираш и уредиш табела со содржина.
Да ажурираш табела со содржина.
Да избришеш табела со содржина.
Да означиш елементи за индекс.
Да креираш и уредиш табела со индекс.
Да ажурираш табела со индекс.
Да избришеш табела со индекс.

Прашања:

1. Што претставува табела со содржина во документ? Зошто таа е потребна?
2. Како ќе вметнеш табела со содржина во документ?
3. Што претставува табела со индекс во документ? Зошто таа е потребна?
4. Како ќе вметнеш табела со индекс во документ?

5. Што треба да направиш пред да креираш табела со индекс?
6. Како се означуваат елементи на индекс?
7. Дали табела со содржина и табела со индекс можат да се ажурираат? Како?

Задача:

1. Во твоите проекти за на училиште вметни табела со содржина. Доколку има потреба, вметни и табела со индекс.

3.4 Шаблони и формулари

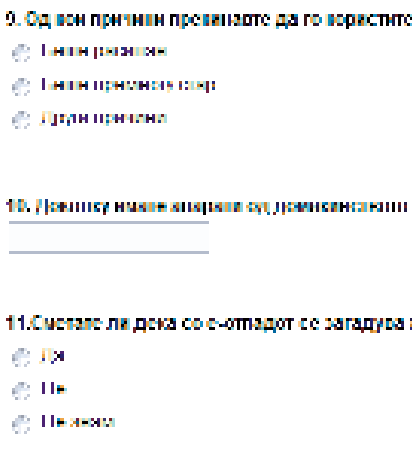
Шаблон (template) е посебен вид на документ кој има однапред дефинирани уредувања, распореди и дизајн според кои ќе се креираат некои идни документи. Тие можат да содржат текст, стилови, заглавие и подножје, нумерирани страници и слично.

Шаблони се корисни за пишување на меморандуми, писма, извештаи и други деловни обрасци. Пример за образец е меморандум на некоја фирма кој содржи основни податоци за фирмата, како што се лого, адреса и телефон. На пр. може да се направи шаблон кој ќе го користат сите ученици за изработка на семинарски и матурски работи уреден според некои правила (на пр. пасусите израмнети од двете страни со големина на буквите 12, проред 1,5 и т. н).

Сите документи кои се работат во MS Word/Writer се базираат на шаблони. Секој нов празен документ се базира на појдовниот шаблон со име *Normal/Нормален* кој има дефинирано неколку основни стилови за наслови и за пасус (параграф).

Формулари се посебен вид шаблони кои содржат фиксни делови кои не се менуваат и полиња кои корисникот треба да ги пополни. На пример, за сите ученици се издава потврда дека се редовни ученици со ист текст освен во следните елементи: име и презиме на ученикот, цел за издавање на потврдата и класен раководител. На сличен начин може да се направат формулари за анкети, за тестови и за други документи.

Полиња (fields) во формулар се подрачја во кои корисниците сами внесуваат текст или одговор на поставено прашање. Полето може да биде текстуално, паѓачка листа со опции за избор, поле за означување, поле за вметнување датум и слично.



Сл. 3. 23 Пример за формулар

Забелешка:

Формуларите можат да се зачуваат како веб страница и да се постават во локална мрежа или на веб-локација.

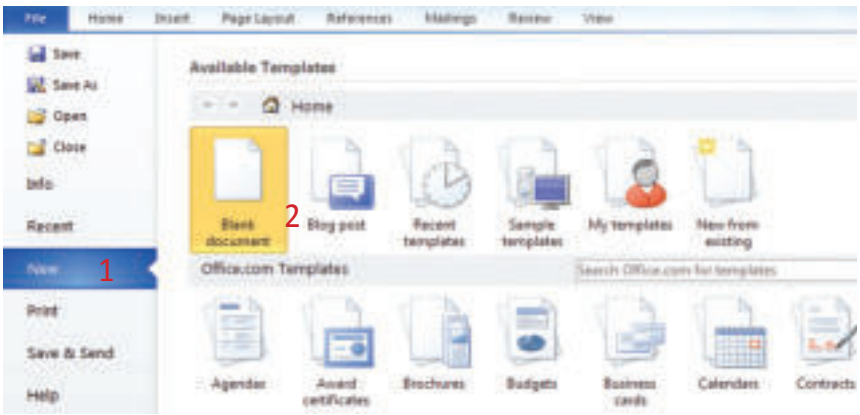
3.4.1 Шаблони и формулари во Ms Word

Шаблоните и формуларите во Ms Word имаат наставка .dotx и се чуваат во посебна папка наречена *Templates*.

Во Word има голем број на шаблони поделени во неколку категории.

1 На картичката *File* се избира наредбата *New* по што ќе се појави галерија со достапни шаблони и формулари поделени по категории;

2 Од галеријата се избира соодветна категорија и се кликнува на неа. Ќе се добијат шаблоните од избраната категорија (кои можат да бидат поделени во поткатегории);



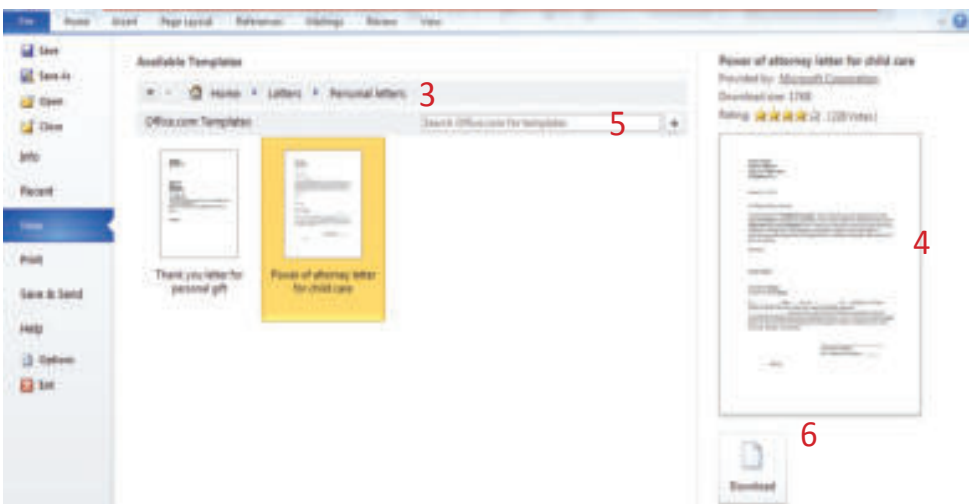
Сл. 3. 24 Категории на шаблони и формулари во Ms Word 2010

3 За полесно снаоѓање прикажана е патека со помош на која може да се врати назад и да се избере друга категорија;

4 Изглед на избраниот шаблон е прикажан на десната страна. Се избира некој од достапните шаблони и двапати се кликнува на него по што тој ќе се отвори;

5 Со алатката *Search* може да се побара соодветен шаблон;

6 Некои шаблони треба да се симнат од веб-локацијата на компанијата *Microsoft*.



Сл. 3. 25 Избирање на соодветен шаблон/формулар

Во некои шаблони има и резервирани места за текст кој се заменува со сопствен текст, тие се нарекуваат формулари. Кога ќе се избере еден од овие шаблони се отвора нов документ кој содржи стилови дефинирани во тој шаблон. Понатаму во документот се работи нормално, дадениот текст се заменува со сопствен текст и документот се зачувува како секој друг документ.

Чекор по чекор:

Напиши писмо со користење на соодветен шаблон од категоријата *Letters*.

1. На картичката *File* кликни на наредбата *New*;
2. Во галеријата со шаблони кликни на категоријата *Letters*;
3. Кликни на папката *Personal Letters*;
4. Избери еден од понудените шаблони и два пати кликни на него;
5. Откако шаблонот ќе се отвори искористи го за да напишеш лично писмо;
6. Кога ќе завршиш зачувај го документот со име *Pismo*.

Креирање на шаблон

За сопствените потреби може да се креираат *прилагодени* шаблони. За таа цел може да се започне од празен документ, од некој завршен и уреден документ или од некој од достапните шаблони. Шаблони се креираат како секој друг документ при што од големо значење е да се користат вградени или прилагодени стилови и теми.

За зачувување на шаблон се повикува наредбата *Save as...* од картичката *File*:

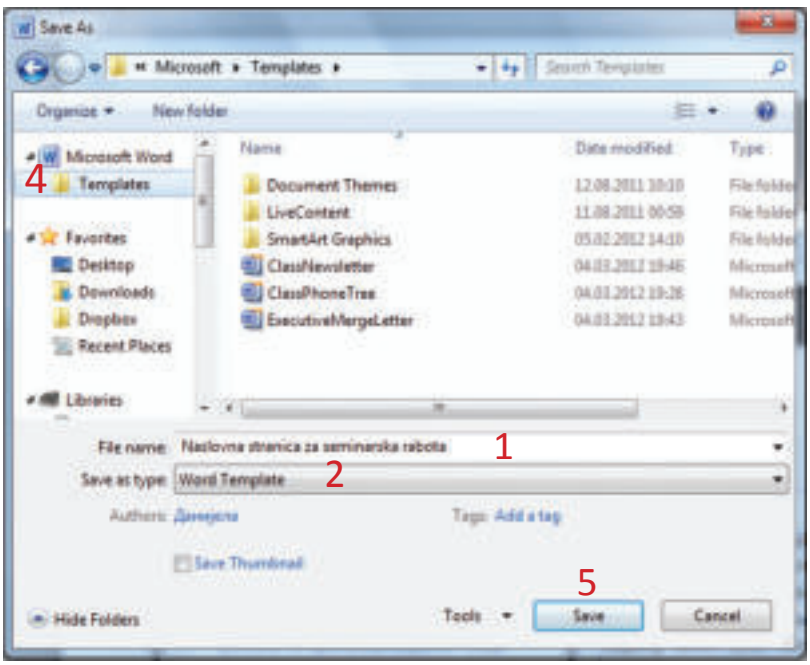
1 На шаблонот му се доделува име во лентата *Name*;

2 Од лентата *Save as type* се избира опцијата *Document Template*;

3 Автоматски ќе биде избрана папката *Templates*;

4 Доколку тоа не се случи, во левото окно на прозорецот се кликнува на папката *Templates*;

5 На крај се кликнува на копчето *Save.*



Сл. 3. 26 Зачувување на шаблон

Со тоа е креиран шаблон кој понатаму може да се користи на ист начин како и веќе постоечките шаблони во *Word*.

Важно!

Клучна работа е зачувувањето на шаблон. Со наредбата Save as од картичката File се отвора познатиот прозорец Save as за зачувување на документи. Прилагодените шаблони се наоѓаат во папката My Templates.

Измена на постоечки шаблон

Измена на постоечките шаблони е многу едноставна. Откако ќе се отвориме, шаблонот се менува по желба и се зачувуваат измените на ист начин како при креирање на нов шаблон.

Вежба:

Во оваа вежба ќе креираш насловна страница за семинарски и проектни задачи. Како пример може да ти послужи слика од десната страна.

1. Отвори нов документ. Во заглавието вметни лого на твоето училиште; На врвот на страницата напиши го името на твоето училиште; Останатиот текст препиши го како на сликата;
2. Уреди го текстот според твојата желба (користи прилагодени стилови);
3. Зачувај го документот како шаблон:
   - Од картичката *Home* избери ја наредбата во прозорецот *Save as*;
   - Во прозорецот *Save as*, во лентата *Name* напиши *Naslovna stranica za seminarska rabota*;
   - Во лентата *Save as type* избери *Document Template*;

Сл. 3. 27 Пример за шаблон

   - Во левото окно на прозорецот кликни на папката *Templates*;
   - Кликни на копчето *Save*.

3.4.2 Креирање на формулар

Во Ms Word може да се креираат формулари кои содржат полиња за контрола, како што се текстуални полиња, полиња за означување, бирачи на датуми, паѓачки листи итн. Корисниците ќе ги користат формуларите така што ќе ги пополнуваат полињата додека останатиот текст, или други содржини доколку ги има, нема да можат да ги менуваат.

За креирање на формулар може да се искористи некој од постоечките документи или шаблони или може да се тргне од празен документ. Важно е формуларот да се зачува како шаблон со веќе опишаната постапка за зачувување на шаблони.

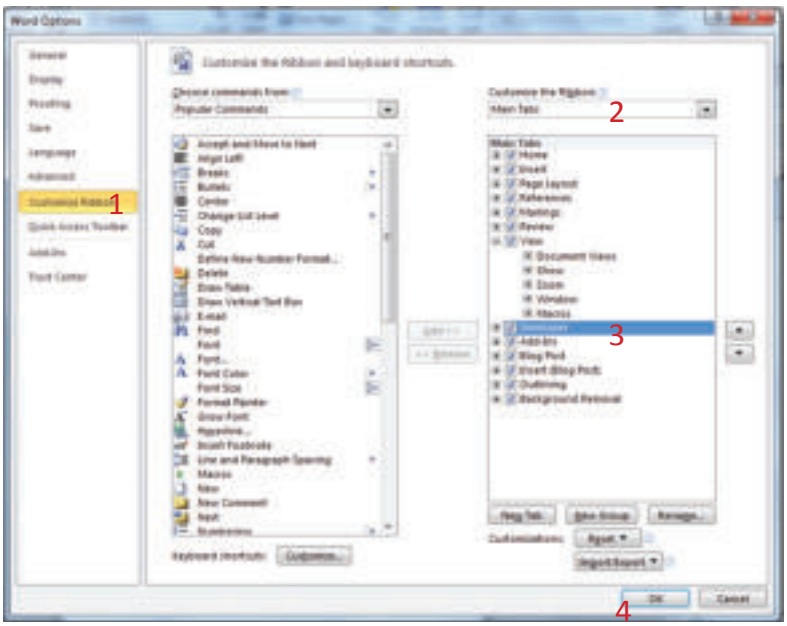
Совет:

Формулари можат да се креираат и со помош на некој шаблон. Многу шаблони можат да се најдат на веб-локацијата Microsoft Office Online.

Прикажување на картичката Developer на рибонот

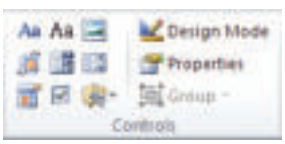
За вметнување на полиња во формуларот е потребна картичката *Developer*. Доколку картичката *Developer* не е прикажана на рибонот од картичката *File* се избира *Options*, потоа:

1. Во прозорецот *Options* се кликнува на *Customize Ribbon*;
2. Во групата *Customize Ribbon* се избира *Main Tabs*;
3. На листата се потврдува изборот на картичката *Developer*;
4. Се кликнува на копчето OK.



Сл. 3. 28 Прикажување на картичката Developer

Вметнување на полиња за контрола



Полињата за контрола се наоѓаат во групата *Controls* на картичката *Developer*. Најпрво се позиционира на местото каде што треба да се вметне соодветно поле, потоа се кликнува на полето.

Текстуалните полиња со збогатен текст (*Rich Text Content Control*) се полиња во кои корисниците можат да уредат текст како задебелен или искосен и можат да внесат текст во повеќе редови. Во текстуалните полиња со обичен текст (*Plain Text Content Control*) овие можности не им се достапни на корисниците.

Во полето за контрола на слика (*Picture Content Control*) корисниците можат да вметнат слика од својот диск.

Во полето со комбинирана рамка (*Combo Box Content Control*) корисниците можат да изберат некоја ставка од листата за избор или можат да напишат свои информации, додека во полето со паѓачката листа (*Drop-Down List Content Control*) корисниците можат само да изберат ставка од листа за избор.

Во полето за означување (*Check Box Content Control*) корисниците по желба потврдуваат или не понуден избор.

Поставување и менување на својства на полињата

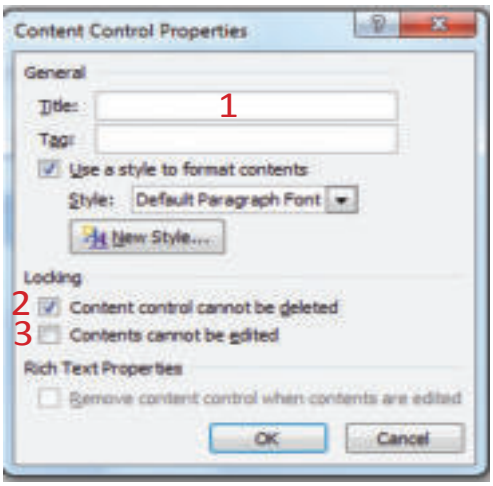
Откако ќе се додаде поле за контрола во формулар може да се постават и да се менуваат неговите својства. На пример, во полето со паѓачка листа треба да се внесат ставки за избор, во полето за контрола на слика може да се вметне слика итн. За да се постават или изменат својства на некое од полињата прво се кликнува во него, а потоа во група *Controls* се кликнува на копчето *Properties* . За секое од полињата се отвора соодветен прозорец *Content Control Properties*.

Општи својства на полиња за контрола на содржините

1 На поле може да се додели наслов во лентата *Title*;

2 Ако се потврди опцијата *Content control cannot be deleted* корисникот нема да може да го избрише полето;

3 Ако се потврди опцијата *Content control cannot be edited* корисникот нема да може да ја смени содржината на полето.



Сл. 3. 29 Прозорец за поставување на својства за текстуални полиња

Сите полиња за контрола ги имаат својствата прикажани на сликата горе (сл. 3.29), но некои полиња имаат и дополнителни својства, соодветни на содржината на полето. Текстуалните полиња ги имаат само општите својства.

Својства на поле со комбинирана рамка и на поле со паѓачка листа

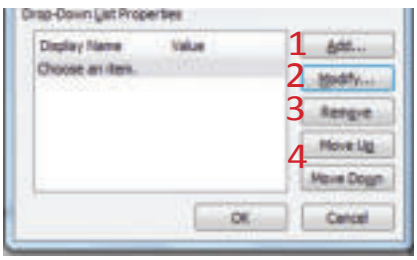
Во полето со комбинирана рамка и во полето со паѓачка листа, освен поставувања на општите својства, треба да се креира и листа за избор. Ставките во листата за избор можат да се:

1 додадат со кликнување на копчето *Add*,

2 менуваат со кликнување на копчето *Modify*,

3 избришат со кликнување на копчето *Remove* или

4 да им се менува редослед со кликнување на копчињата *Move Up* и *Move Down*.



Сл. 3. 30 Уредување на ставки за избор

5 Кога се додава или менува ставка се отвора прозорец во кој, во полето *Display Name,* се пишува текст кој ќе се појави во листата.

Сл. 3. 31 Додавање на ставка за избор

Забелешка:

Ако е потврдена опцијата Content control cannot be edited корисниците нема да можат ништо да изберат.

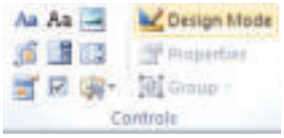
Својства на полето за означување

Во полето за означување може да се изберат различни симболи за означување во полето.

Сл. 3. 32 Избор на симбол за означување на поле

Додавање на упатства во формуларот

Во полињата за контрола на содржина е добро да се напишат упатства за корисниците.



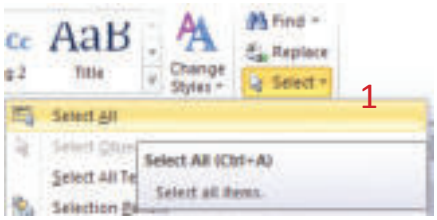
За да се додаде упатство, прво треба да се помине во режим на дизајнирање така што на картичката *Developer*, во групата *Controls*, се кликнува на копчето *Design Mode*.

Потоа се кликнува во поле во кое што треба да се напише упатство. Наместо прикажаниот текст *Click here to enter a date* се пишува текст со упатството (на пр. „Внеси име и презиме“) и се уредува по желба. На крај повторно се кликнува на копчето *Design Mode* со што се излегува од режимот на дизајнирање.

Заштита на формулар

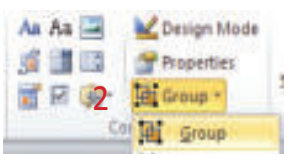
Пред да се зачува и проследи до корисниците формуларот треба да се заштити. Така корисниците ќе можат да го менуваат формуларот само во полињата каде што тоа им е дозволено, додека текстот на формуларот нема да можат да го сменат.

Формулар се заштитува така што сите негови елементи се групираат во еден елемент.



1 За да се селектираат сите елементи во формуларот, на картичката *Home*, во групата *Editing*, се кликнува на копчето *Sellect*, од паѓачката листа се избира *Sellect All* (кратенка *CTRL+A*);

2 На крај, на картичката *Developer*, во групата *Controls*, се кликнува на копчето *Group* и во листата повторно се кликнува на *Group*.



Сл. 3. 33 Заштита на формулар

Забелешка:

Заштитата се отклонува на ист начин, само што овој пат се избира Ungroup.

Чекор по чекор:

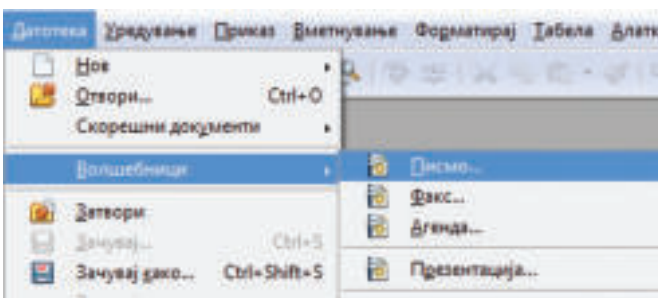
Креирај насловна страница за семинарска работа како формулар.

1. Отвори нов документ (*File→New*);
2. Зачувај го документот како шаблон во папката *Templates* со име „Naslovna za seminarska rabota“ (*File→Save as...*);
3. Во заглавието вметни лого на твоето училиште; На врвот на страницата напиши го името на твоето училиште, текстот уреди го по желба;
4. Остави неколку редови празни па напиши „Тема:“;
5. Кликни на картичката *Developer* за да ја отвориш;
6. Позиционирај се едно место по текстот „Тема:“ па во групата *Controls* кликни на копчето *Rich Text Content Control* за да додадеш текстуално поле;

7. Во групата *Controls* кликни на копчето *Properties*, во прозорецот *Content Control Properties* во полето *Title* напиши „Тема“, потоа потврди го полето *Content control cannot be deleted* и кликни на копчето *OK*;
8. Во следниот ред напиши „Предмет:“;
9. Остави едно место празно па кликни на копчето за да додадеш паѓачка листа;
10. Повтори го чекорот 7, во полето *Title* напиши „Предмет“;
11. Во долниот дел од прозорецот во делот *Drop-Down List Properties* кликни на текстот *Choose an item*, потоа кликни на копчето *Modify* и во полето *Display name* напиши „Избери предмет...“ и кликни на копчето *OK*;
12. Кликни на копчето *Add* па во полето *Display name* напиши „Информатика“ и кликни на копчето *OK*;
13. Претходниот чекор повтори го за сите предмети;
14. На дното на страницата лево напиши „Ученик“, а десно напиши „Ментор“;
15. Позиционирај се во следниот ред; повтори ги чекорите 6 и 7 и нацртај полиња за полињата на кои ќе им доделиш наслови „Ученик“, и „Ментор“ ;
16. Кликни на копчето *Design Mode* за да поминеш во режимот на дизајнирање;
17. Кликни во полето Наслов избриши го прикажаниот текст и напиши го упатството: „Наслов на темата“, текстот уреди го по желба;
18. Претходниот чекор повтори го за полињата Ученик и Ментор;
19. Повторно кликни на копчето *Design Mode* за да го исклучиш режимот на дизајнирање;
20. Селектирај ги сите елементи во документот (Ctrl+A) па кликни на копчето *Group* за да го заштитиш формуларот;
21. Зачувај го документот!

3.4.3 Користење и креирање на формулари во Writer

Наједноставниот начин за користење на формулар во Writer е тој да се отвори преку волшебник кој се повикува со наредбата *Датотека → Волшебници*. Може да се избере еден тип од понудените шаблони или формулари, на пр. писмо, факс итн.



Сл. 3. 34 Волшебник за формулар

Понатаму волшебникот не води низ неколку чекори при кои може да се изберат и постават некои својства на формуларот. На пример, за писмо може да се избере деловно писмо, формално лично писмо или лично писмо, еден од понудените дизајни и слично.

Од еден до друг чекор се поминува со копчето *Следно*. На крај се кликнува на копчето *Заврши* по што формуларот ќе се отвори.

За користење на овој формулар доволно е тој да се пополни и да се зачува како обичен документ. Формуларот може да се измени според потребите. Може да се додадат уште полиња или да се изменат својства на постоечките полиња.

Забелешка:

Формулар може да се креира и од празен документ кој задолжително треба да се зачува како шаблон со наставката .ott.

За вметнување на полиња на контрола, во формуларот се повикува лентата *Контрола на формата* преку наредбата *Приказ→Алатници→Контрола на формата*.

На оваа лента копчето се користи за да се помине од режим на дизајн во режим на користење на формуларот. Формуларот мора да биде во режим на дизајн додека се внесува текст и се додаваат полиња, но за корисникот да може да го користи формуларот важно е режимот на дизајн да биде исклучен.

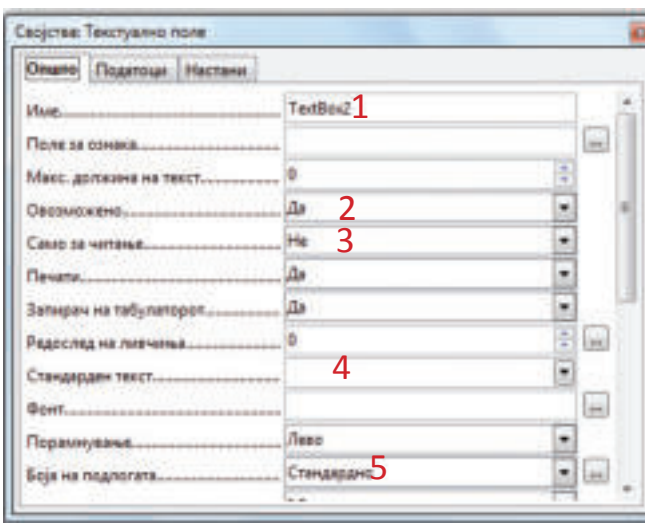
Додавање на полиња

Текстуално поле се додава со копчето , поле за означување се додава со копчето , поле со паѓачка листа се додава со копчето . Откако ќе се избере вид на поле, тоа треба да се нацрта во документ.

Поставување и менување на својства на полињата

Откако ќе се додаде поле во формулар треба да се постават и неговите својства. Својствата се поставуваат во прозорец кој се добива со двоклик на соодветното поле или со кликнување на копчето од лентата за форми.

Својства на текстуалното поле се поставуваат во прозорецот *Својства Текстуално поле*:



Сл. 3. 35 Прозорец за поставување на својства за текстуално поле

1 Во полето *Име* се доделува име на полето;

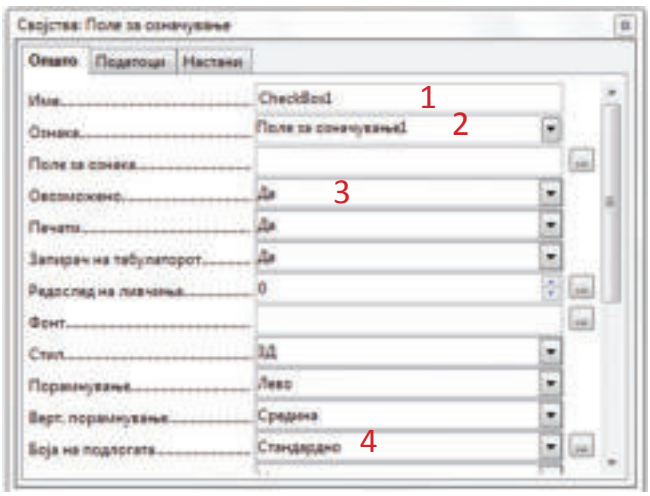
2 Во полето *Овозможено* се дозволува или забранува пристап до текстот во полето (корисникот не може да го означи текстот);

3 Во полето само за читање се уредува дали текстот во полето ќе биде само за читање (не може да се менува);

4 Во полето *Стандарден текст* се внесува текст кој ќе се прикаже во полето кога корисникот ќе го отвори формуларот;

5 Доколку сакаме полето да биде воочливо се менува бојата на подлога во лентата *Боја на подлога*.

Својства на полето за означување се поставуваат во прозорецот *Својства: Поле за означување:*



Сл. 3. 36 Прозорец за поставување на својства за поле за означување

1 Во полето *Име* се пишува име на полето;

2 Во полето *Ознака* се пишува текст кој ќе стои до полето. Ова е текст чиј избор корисникот треба да го потврди доколку сака;

3 Во полето *Овозможено* се дозволува или се забранува пристап до полето;

4 Доколку сакаме полето да биде воочливо се сменува бојата на подлога во лентата Боја на подлога.

Чекор по чекор:

Креирај насловна страница за семинарска работа како формулар.

1. Отвори нов документ (*Документ→Нов→Текстуален документ*);
2. Зачувај го документот како шаблон во папката *Templates* со име Naslovna za seminarska rabota (*Документ→Зачувај како...*);
3. Во заглавие вметни лого на твоето училиште;
4. На врвот на страницата напиши го името на твоето училиште, текстот уреди го по желба;
5. Остави неколку редови празни па напиши „Тема:“;
6. Отвори ја лентата *Контрола на формата* (*Приказ→Алатници→Контрола на формата*);

7. Позиционирај се едно место по текстот „Тема:“, во лентата *Контрола на формата* кликни на копчето *Форматирано поле* и нацртај текстуално поле;
8. Кликни двапати на полето за да го отвориш прозорецот *Својства на формата*. Во прозорецот, во картичката *Општо*, во полето *Име* напиши „Тема“, затвори го прозорецот;
9. Во следниот ред напиши „Предмет:“;
10. На дното на страницата лево напиши „Ученик“ а десно „Ментор“;
11. Позиционирај се во следниот ред, повтори ги чекорите 7 и 8 и креирај полиња со наслови „Ученик“ и „Ментор“;
12. Кликни на копчето *Режим на дизајн* за да го исклучиш режимот на дизајнирање;
13. Зачувај го документот.

Резиме

Шаблон е вид на документ кој има однапред дефинирани уредувања, распореди и дизајн според кои ќе се креираат некои идни документи со иста структура.

Формулари се вид шаблони кои содржат фиксен тескт кој не се менува и полиња кои корисникот треба да ги пополни. Полињата во формулар се подрачја во кои корисниците сами внесуваат текст или одговор на поставено прашање.

Шаблоните и формуларите во Ms Word имаат наставка .dotx и се чуваат во посебна папка наречена Templates.

Шаблоните и формуларите во Writer имаат наставка .ott.

Вештини што треба да ги усовршиш:

Да отвориш и примениш постоечки шаблон на документ.
Да измениш постоечки шаблон според твоите потреби.
Да креираш шаблон со одредени стилови и уредувања.
Да зачуваш документ како шаблон.
Да поставиш поле за контрола на текст во формулар.
Да поставиш поле за контрола на слика во формулар.
Да поставиш поле со паѓачка листа во формулар.
Да поставиш поле со комбинирана рамка во формулар.
Да поставиш поле со поле за потврда во формулар.
Да поставиш поле за контрола на датум во формулар.
Да поставиш и измениш својства на полиња за контрола.
Да заштитиш формулар.

Прашања:

1. Што е шаблон? Зошто шаблоните се корисни?
2. Што е формулар? За што можат да се користат формулари?
3. Како се креираат шаблони и формулари?
4. Кој тип на датотека треба да се избере при зачувување на шаблон и на формулар?
5. Што се полиња за контрола во формулар? Какви тие можат да бидат?
6. Како се поставуваат полиња за контрола во формулар?

7. Како се поставуваат својства на полиња за контрола во формулар?
8. Зошто е потребно формуларот да се заштити?
9. Како се заштитува формулар?

Задачи:

1. Креирај анкетен лист.
   - Отвори нов празен документ и веднаш зачувај го како шаблон (додели име по желба);
   - Напиши го насловот на анкетата;
   - Напиши ги прашањата на анкетата, предвиди да има прашања на кои корисникот ќе одговори со пишување на одговор во текстуално поле, со бирање на еден одговор од паѓачката листа и со избирање на повеќе одговори преку полињата за означување;
   - Под секое прашање вметни соодветно поле за одговор;
   - За прашања на кои е можен повеќекратен избор напиши ги понудените избори и до нив вметни поле за потврдување;
   - Во паѓачките листи напиши ги ставките за избор;
   - За сите полиња постави својство тие да не можат да се избришат;
   - Доколку има потврда од упатства напиши ги;
2. Креирај покана за роденденска забава! Постави полиња за контрола за име и презиме на поканетиот гостин и за место и за време на одржување на забавата.

3.5 Заштита на документи

Често се јавува потреба документите да се заштитат. Тие некогаш се заштитуваат од недозволено читање, а некогаш само од недозволено менување.

Постојат два вида на заштита на документи: заштита со лозинка и заштита со давање на атрибутот „Read Only“ (само за читање).

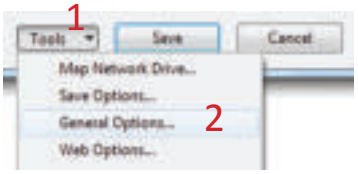
3.5.1 Заштита на документи во MS Word 2010

Заштита на документ со поставување лозинка

При *заштита со лозинка* документот можат да го отворат или да го менуваат само корисниците кои ја знаат лозинката. Постојат два типа на лозинка:

- лозинка за заштита при отворање на документ (Password to Open) – за да се отвори документ мора да се внесе лозинка и
- лозинка за заштита од менување на документ (Password to Modify) – документот може да се отвори за читање, но за вршење на било какви измени мора да се внесе лозинка.

Еден од начините да се постави лозинка на документ е при зачувување на документот. Во прозорецот *Save as*:



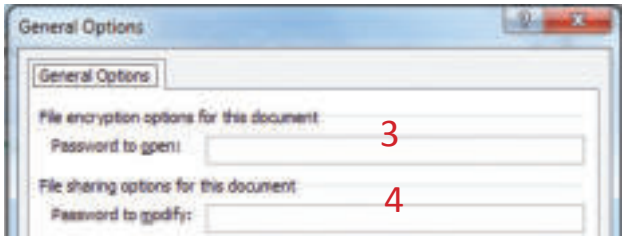
1 Се кликнува на копчето *Tools* (десно од копчето *Save*);

2 Во листата се кликнува на *General Options*;

Во прозорецот *General Options*:

3 Во лентата *Password to Open* се запишува лозинка за заштита при отворање на документ;

4 Во лентата *Password to Modify* се запишува лозинка за заштита од менување на документ.

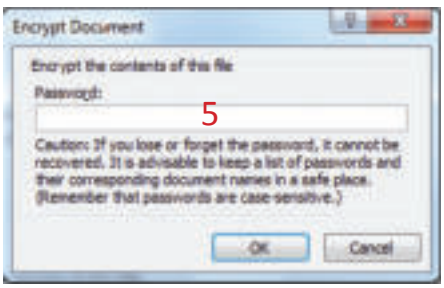


Сл. 3. 37 Прозорецот General Options

Со кликнување на копчето *ОК*

5 се добива прозорецот *Confirm Password* во кој лозинката треба повторно да се внесе за да се потврди.

Доколку не се внесе истата лозинка се добива порака за грешка и постапката треба да се повтори. Лозинката е чувствителна на разликата помеѓу големите и малите букви.



Сл. 3. 38 Заштита на документ со лозинка

Важно!

По кликнување на копчето ОК документот треба да се зачува за лозинката да се активира.

Чекор по чекор:

1. Отвори некој од постоечките документи;
2. Отвори ја картичката *File* па кликни на наредбата *Save as...*;
3. Во прозорецот Save as кликни на копчето *Tools*;
4. Во листата кликни на *General Options*;
5. Во прозорецот *General Options* во лентата *Password to Open* напиши лозинка за заштита при отворање на документ и кликни на копчето *ОК*;
6. Во прозорецот *Confirm Password* лозинката напиши ја уште еднаш и кликни на копчето *ОК*;
7. Не заборавај да го зачуваш документот!

3.5.2 Заштита на формулар

За еден од начините на заштита веќе се зборуваше во претходната лекција. Тоа е заштита преку групирање на сите објекти во формуларот. Всушност ова и не е вистинска заштита затоа што секој може да ги одгрупира објектите и да го измени формуларот. Затоа се препорачува заштита на формулар со лозинка која и понатаму им овозможува на корисниците да внесуваат податоци во полињата.

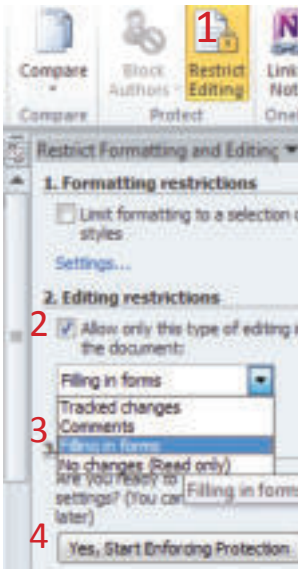
За да се дозволи менување на содржината на формуларот само во полињата за контрола:

1 Од картичката *Review*, во групата *Protect*, се кликнува на копчето *Restrict Editing;*

2 Во панелот *Protect Document*, во делот *Editing Restriction*, се потврдува опцијата *Allow only this type of editing in the document*;

3 Потоа од лентата под неа се избира *Filling in forms*;

4 Се кликнува на копчето *Yes, Start Enforcing Pretection*;

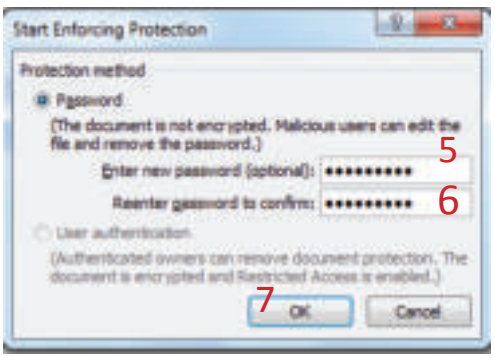


Сл. 3. 39 Заштита на формулар со дозвола на внесување во полиња за контрола

5 Во добиениот прозорец се пишува лозинката во лентата *Enter new password*;

6 Лозинката се внесува уште еднаш во лентата *Reenter password to confirm* за да се потврди;

7 Се кликнува на копчето *OK*.



Сл. 3. 40 Поставување лозинка за заштита на формулар

Со ваков вид на заштита само оние корисници кои ја знаат лозинката ќе можат да ја отстранат заштитата и да го изменат формуларот. Корисниците кои не ја знаат лозинката и понатаму ќе можат да внесуваат податоци во контролните полиња.

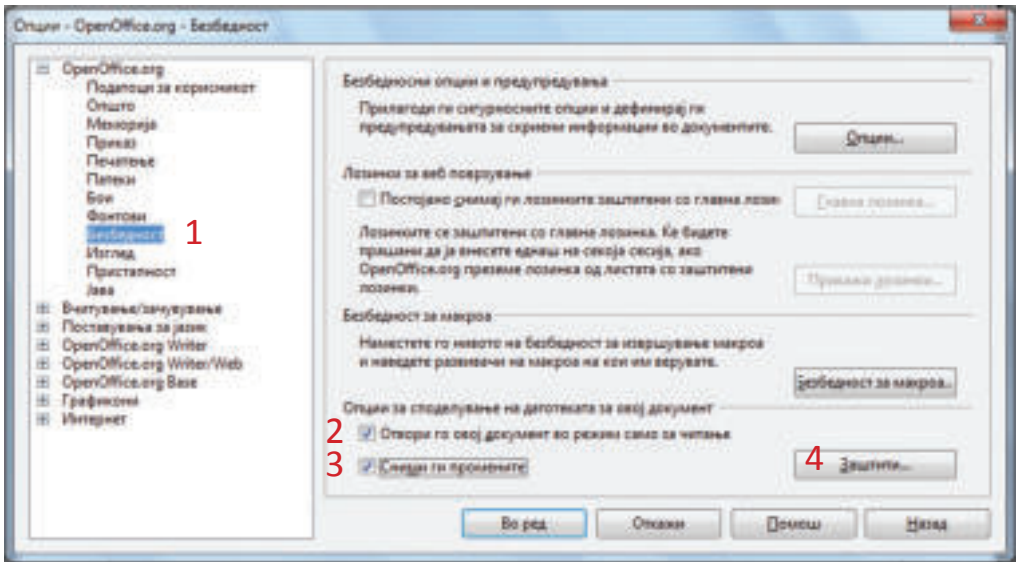
Чекор по чекор:

1. Отвори го формуларот Naslovna stranica za seminarska rabota;
2. На картичката *Review*, во групата *Protect*, кликни на копчето *Restrict Editing*;
3. Во панелот *Protect Document*, во делот *Editing Restriction*, потврди ја опцијата *Allow only this type of editing in the document*;
4. Од лентата под неа избери *Filling in forms*;
5. Кликни на копчето *Yes, Start Enforcing Pretection*;
6. Во лентата *Enter new password* напиши ја твојата лозинка;
7. Лозинката потврди ја во лентата *Reenter password to confirm*;
8. Кликни на копчето *OK*.

3.5.3 Поставување заштита на документ во Writer

За да се постави заштита на документ во Writer се повикува наредбата *Алатки→Опции*,

1 Во прозорецот *Опции* се избира *Безбедност*;



Сл. 3. 41 Заштита на документ

2 Опцијата *Отвори го овој документ во режим само за читање* го штити документот од несакани промени. Сепак постои можност да се уреди копија од документот и таа да се зачува со исто име како оригиналот;

3 Опцијата *Сними ги промените* овозможува снимање на промените;

4 Се кликнува на копчето *Заштити...* по што се отвора прозорецот *Внесување лозинка*.

5 Во лентата *Лозинка* се пишува лозинката;

6 Во лентата *Потврда* лозинката се пишува уште еднаш за да се потврди;

7 Ако лозинката точно е внесена и по втор пат, се кликнува на копчето *Во ред* во двата прозорци.

Сл. 3. 42 Внесување лозинка при заштита на документ

Други корисници на документот може да направат промени, но не може да го зачуваат документот без лозинката.

Забелешка:

Лозинката мора да биде долга најмалку пет карактери и е чувствителна на разликата помеѓу големите и малите букви.

Чекор по чекор:

1. Од менито *Алатки* кликни на наредбата *Опции*;
2. Во прозорецот *Опции* избери *Безбедност*;
3. Потврди ја опцијата *Отвори го овој документ во режим само за читање*;
4. Потврди ја опцијата *Сними ги промените*;
5. Кликни на копчето *Заштити*;
6. Во прозорецот *Внесување лозинка* напиши ја твојата лозинка во лентата *Лозинка*;

7. Внеси ја лозинката уште еднаш во лентата *Потврда*;
8. Кликни на копчето *Во ред* во двата прозорци.

Отстранување заштита на документ во Writer

Доколку документот е заштитен, копчето *Заштити* се вика *Отстрани заштита*. Се кликнува на копчето *Отстрани заштита* и се внесува лозинка за да се исклучи заштитата.

Резиме

При заштита со лозинка документот може да го отворат или менуваат само корисниците кои ја знаат лозинката. Постојат два типа на лозинка: лозинка за заштита при отворање на документ и лозинка за заштита од менување на документ.

Формуларите можат да се заштитат така што ќе се дозволи менување на содржината на формуларот само во полињата за контрола.

Вештини што треба да ги усовршиш:

Да заштитиш документ од недозволено читање со поставување на лозинка.

Да заштитиш документ од недозволено менување со поставување на лозинка.

Да заштитиш формулар со тоа што ќе дозволиш менување на содржина во контролните полиња.

Да отстраниш заштита од документ.

Прашања:

1. Зошто се поставува заштита на документ?
2. Што е лозинка?
3. Какви типови на лозинка постојат при заштита на документ во MS Word?
4. На колку начини знаеш да поставиш заштита на документ во MS Word? Објасни ги!
5. Постави заштита на документ во MS Word со различни типови на лозинка!
6. Како се поставува заштита со лозинка на документ во Writer?
7. Постави заштита на документ во Writer!

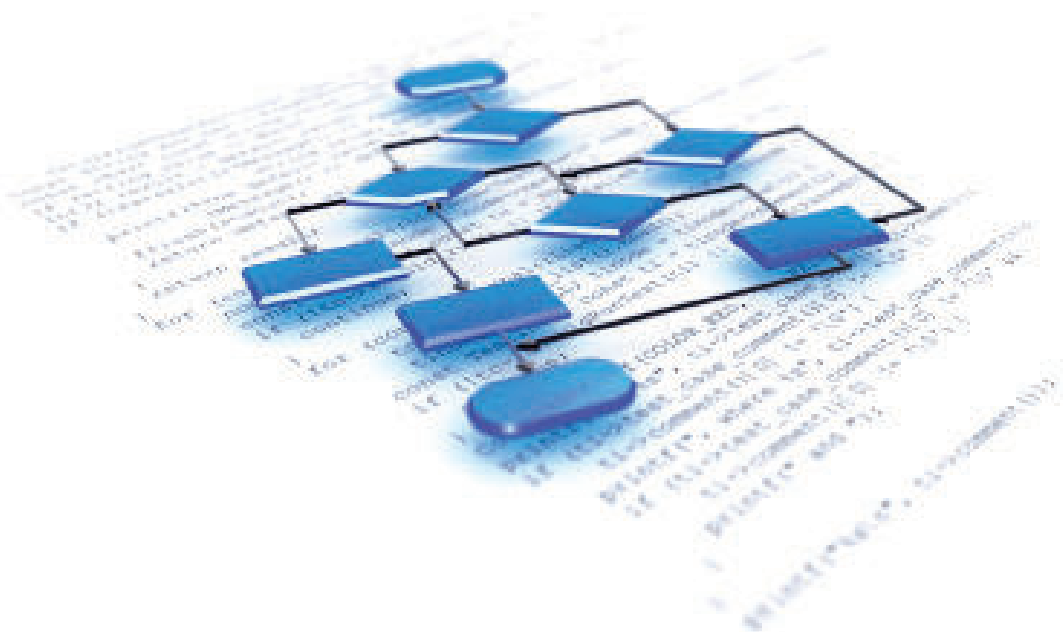
Задачи:

1. Заштити ги со лозинка формуларите „Naslovna stranica za seminarska rabota“ и „Anketa“ со тоа што ќе дозволиш менување на контролните полиња.

ПРОГРАМИРАЊЕ ВО C++

Клучни зборови

- Алгоритам
- Блок дијаграм
- Псевдо код
- Кодирање
- Разгранување
- Циклус
- Програма
- Програмирање
- Програмски јазик
- Интегрирана околина за програмирање
- Преведувач (компајлер)
- Интерпретер
- Изворна програма
- Извршна програма
- Дебагирање
- Тестирање
- Индентација
- Исказ
- Променлива
- Тип на променлива
- Идентификатор
- Декларирање
- Иницијализација



4.1 Поим за алгоритми и програми

4.1.1 Алгоритми

Кога решаваме некоја задача, користиме логика, искуства, знаења и интуиција. Компјутерот ни помага во решавање на многу задачи кои секојдневно ги извршуваме како што се пресметување, пишување на текстови, цртање и слично. Но компјутерот не е интелигентна машина и ниту една задача не може да изврши самостојно. Тој задачите ги извршува според однапред зададени упатства, изработени од човекот, во кои се дефинирани и прецизно формулирани чекори за решавање на некоја задача. Така разработена постапка се нарекува *алгоритам*.

Пр. 4. 1. Алгоритам за користење на автомат за кафе:

Чекор 1. Во ценовникот прочитај ја цената на саканиот напиток;

Чекор 2. Во отворот за пари уфрли пари според наведената цена;

Чекор 3. Притисни го копчето до името на напитокот;

Чекор 4. Ако сакаш напиток со шеќер, притисни го копчето до натписот „шеќер“;

Чекор 5. Почекај напитокот да се подготви и чашата да се наполни;

Чекор 6. Земи ја чашата со напитокот.

Со ваквите упатства се среќаваме во секојдневниот живот, само не ги нарекуваме алгоритми. Така, на пример, рецепт за подготвување на некое јадење, упатство за ракување со некој апарат или машина, упатство за однесување во случај на земјотрес, упатство за изработка на училиштен проект и многу други упатства или правила за однесување претставуваат алгоритми. Со алгоритмите често се среќаваме и во математиката. Секоја формула и секоја постапка за решавање на некоја задача претставува алгоритам. Но, вистинското значење на поимот алгоритам се наоѓа во информатиката каде тој означува прецизна низа од инструкции дадени на компјутерот. Алгоритамот се дефинира на следниов начин:

Алгоритам претставува постапка од конечен број на прецизно формулирани дејства со точно зададен редослед на нивното извршување. Дејствата од кои се состои алгоритамот се нарекуваат алгоритамски чекори.

За љубопитните:

Зборот алгоритам доаѓа од зборот Alchwarizmi или Kovarezma што е прекар на персискиот писател, математичар и астроном од 9-иот век Muhammeda ibn Muse al Khowarizmi. Тој во својот учебник по математика прикажал решенија на некои аритметички проблеми во форма на упатства кои се состоеле од точно одредени чекори.

Преку алгоритмите решавањето на задачите се сведува на нивното расчленување и на решавање на повеќе едноставни задачи кои меѓусебно се поврзани во една целина. Алгоритмите треба да се недвосмислени, јасни и прецизни. Секоја операција треба да е јасно определена, треба прецизно да е утврден редоследот на извршувањето на операциите и треба да е разбирлив за секого, независно кој го напишал. Многу задачи можат да се решат на повеќе начини и за нивното решение можат да се напишат повеќе алгоритми. Секогаш треба да се стреми кон најбрзото, најефективното и најсигурното решение.

Зад. 4. 1. Напиши алгоритам за праќање на СМС порака!

Секој алгоритам може да се запише на природен јазик (македонски, англиски и др.), како што тоа е направено во претходниот пример. Ќе разгледаме уште еден пример:

Пр. 4. 2. Алгоритам за пресметување на цена за такси услуга претставен со природен јазик:

Чекор 1. Прочитај вредности за влезните податоци: почетната цена S, цената по километар К и бројот на поминатите километри L;

Чекор 2. Пресметај ја цената за такси услугата според формулата C=S+K·L;

Чекор 3. Прикажи ја (на монитор) цената C.

Во секој алгоритам мора јасно да бидат дефинирани влезни податоци (во примеров S, K, L) врз кои се извршуваат операции.

На крајот на секој алгоритам се добиваат излезни податоци или резултати (во примеров C).

Алгоритамот мора да биде составен од конечен број чекори кои укажуваат на редослед на операциите што треба да се извршат врз влезните податоци за да се добие резултат. Секој чекор се опишува со инструкција (во примеров *дефинирај, прочитај, пресметај, прикажи*).

Карактеристики на алгоритам се:

- *Конечност* – алгоритамот мора да доведе до решение по конечен број на чекори.
- *Дефинираност и недвосмисленост* – секој алгоритамски чекор мора да биде еднозначно дефиниран, треба да се предвидат сите случаеви за различни влезни податоци.
- *Влез* – алгоритамот може но не мора да има еден или повеќе влезни податоци.
- *Излез* – алгоритамот мора да има еден или повеќе излезни податоци.
- *Ефикасност* – алгоритамот треба да доведе до решение во што пократко време со примена на што помалку чекори.
- *Остварливост* – алгоритамот мора да биде остварлив на компјутер.

Зад. 4. 2. Напиши алгоритам за пресметување на плоштина на правоаголник! Кои се влезни податоци на алгоритамот? Што е резултат на алгоритамот? Според која формула ќе ја пресметаш плоштината?

Претставување на алгоритми

Запишувањето на алгоритми со природен јазик е лесно и разбирливо, но кога се работи за сложени проблеми тоа може да биде предолго. Од тие причини се користат други начини за претставување на алгоритми.

Најчесто се користат следниве два начина:

- *графички приказ*
- *псевдо јазик*

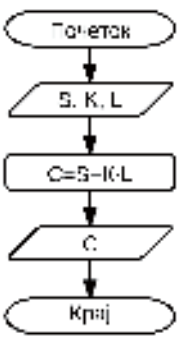
Претставување на алгоритам со графички приказ

Графичкиот приказ на алгоритмите се врши со тнр. блок дијаграм (flowchart). Во блок дијаграмот се користат посебни графички симболи за одредени дејства. Ќе ги наведеме само симболите кои најчесто се користат:

Почеток Крај	Се дефинираат почетокот и крајот на алгоритамот. Сите останати симболи мора да бидат помеѓу овие два симбола.
	Се опишува влез, односно читање на податоци.
	Се опишува излез, односно прикажување на резултатите.
	Се опишуваат влезно-излезни величини. Во поново време со овој симбол се заменуваат претходните два симбола со кои се опишуваат влез и излез.
	Се дефинира обработка на податоци, на пр. пресметување или доделување на вредност на некоја променлива.
не да	Се дефинира процес на одлука или проверка, односно разгранување. Горниот врв е влез, а двата излези се означени со зборовите „да“ и „не“. Ако условот кој е запишан во симболот е исполнет, алгоритамот продолжува со извршување на чекори по излезот означен со „да“, а ако условот не е исполнет алгоритамот продолжува со извршување на чекори по излезот означен со „не“.
	Се поврзуваат делови од блок дијаграмот.

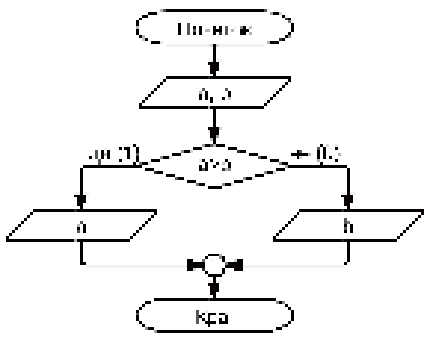
Симболите се цртаат во низа од горе кон долу и меѓусебно се поврзуваат со линии. Оваа насока е логичен редослед на случувањата. Линиите кои означуваат некоја друга насока, на пр. од долу кон горе или наназад, задолжително се означуваат со стрелки.

Пр. 4. 3. Алгоритам за пресметување на цена за такси услуга претставен со блок дијаграм:



Сл. 4. 1 Блок дијаграм за пресметување на цена за такси услуга.

Пр. 4. 4. Алгоритам за наоѓање на поголемиот од два броја претставен со блок дијаграм:



Сл. 4. 2 Блок дијаграм за наоѓање на поголем од два броја.

Графичкиот приказ на алгоритмите е попрегледен и олеснува пронаоѓање на грешки доколку ги има. Но, ако се работи за сложени задачи, блок дијаграм може да стане заплеткана мрежа која зафаќа повеќе страници во која тешко се снаоѓаме.

Зад. 4. 3. Прикажи го графички алгоритамот за пресметување на плоштина на правоаголник!

Претставување на алгоритам со псевдо јазик

Алгоритмите можат да се претстават и со псевдо јазик кој е сличен со природниот јазик освен што користи помал број на точно определени зборови и е прецизен и недвосмислен. На пример, псевдо јазик може да содржи зборови како што се *читај*, *пресметај*, *печати, извршувај, повторувај, ако, тогаш, инаку* итн., како и симболи, на пр. +, -, *, /, = . Ваков јазик е доволно едноставен и разбирлив за човекот, а од друга страна е погоден за трансформирање на алгоритам во програма.

Пр. 4. 5. Алгоритамот за пресметување на цената за такси услуга претставен со псевдо јазик:

```
почеток
    читај S, K, L;
    пресметај C=S+K*L;
    печати C;
крај
```

Пр. 4. 6. Алгоритамот за наоѓање на поголемиот од два броја претставен со псевдо јазик:

```
почеток
    читај a,b;
    ако a>b тогаш печати a
        инаку печати b;
крај
```

Вака запишаниот алгоритам се нарекува *псевдо код* (лажен код). Ова сè уште не е програма која може да ја изврши компјутерот, бидејќи се користат зборови од природниот јазик кои компјутерот не ги разбира. Алгоритамот запишан со псевдо код е најсличен на програма и тој најлесно се трансформира во програмски код.

Зборовите кои одредуваат инструкции се нарекуваат *клучни* или *резервирани зборови*. Нив во псевдо кодот ќе ги пишуваме со задебелени букви. Инструкциите се одделуваат со знакот точка и запирка (;). За поголема прегледност на псевдо кодот се користи начин на запишување наречен *назабување* – одредени делови од инструкции се пишуваат малку вовлечено кон десно.

Зад. 4. 4. Прикажи го со псевдо јазик алгоритамот за пресметување на плоштина на правоаголник!

Алгоритамски структури

Под структура на алгоритам се подразбира редослед на извршување на одделни чекори во алгоритамот. Секој алгоритам може да се претстави со користење на една од трите основни структури:

- редоследна структура (секвенца) или линиска структура
- структура за избор (селекција) или разгранета структура
- структура со повторување (итерација) или циклична структура.

Линиска (линеарна) структура

Линиската алгоритамска структура подразбира извршување на алгоритамските чекори еден по друг како што тие следат. Од почетокот до крајот постои само еден пат, нема повторување и нема разгранување. Ова обично се алгоритми кај кои има влез, обработка на податоци и излез (на пр. пресметување на цена на такси услуга).

Разгранета структура

Со разгранетата алгоритамска структура е овозможено решавање на проблеми каде податоците треба да исполнат некој услов (дали бројот е позитивен, дали бројот е парен и сл.). Во зависност од тоа дали условот е исполнет или не е исполнет, се извршуваат едни или други чекори. Значи, постои точка на одлучување, односно разгранување на алгоритам (на пр. алгоритамот за печатење на поголемиот од два броја).

Циклична структура

Понекогаш е потребно некои алгоритамски чекори да се повторат, односно да се извршат повеќе пати. Такви алгоритамски чекори претставуваат циклуси, а алгоритамските структури кои содржат циклуси се нарекуваат циклични алгоритамски структури.

Можни се две ситуации:

- Однапред се знае колку пати циклусот ќе се повтори (на пр. наполни 10 шишиња со вода);
- Бројот на повторувањата зависи од некој услов и тој број не е однапред познат. При тоа условот може да биде:
  - на почетокот на циклусот (на пр. додека е црвено светло не поминувај преку улица) или
  - на крајот на циклусот (на пр. печати копии додека има листови).

Резиме

Алгоритам претставува постапка од конечен број на прецизно формулирани дејства со точно зададен редослед на нивното извршување. Дејствата од кои се состои алгоритамот се нарекуваат *алгоритамски чекори*. Карактеристиките на алгоритамот се: конечност, дефинираност, влез, излез, ефикасност и остваривост.

За претставување на алгоритмите најчесто се користат графичкиот приказ и псевдо јазикот. Графичкиот приказ на алгоритмите се врши со *блок дијаграм*. Секој алгоритам може да се претстави со користење на една од трите основни структури: линиска структура, разгранета структура и циклична структура.

Прашања:

1. Што е алгоритам?
2. Наведи неколку алгоритми за решавање на задачи од секојдневниот живот!
3. Какви особини мора да има секој алгоритам?
4. Кои се карактеристики на алгоритамот?
5. Како се претставуваат алгоритмите?
6. Што е блок дијаграм, а што е псевдо код?
7. Набroј ги и објасни ги основните структури на алгоритмите!
8. Наведи примери за секоја од основните алгоритамски структури!

Задачи:

1. Напиши алгоритам за пресметување на поминатот пат на автомобил кој одредено време се движел со одредена средна брзина! Што е влез, а што е излез на овој алгоритам?
2. Алгоритамот од задачата 1 претстави го со блок дијаграм и со псевдо код!

3. Нацртај блок дијаграм и напиши псевдо код за алгоритам со кој ќе се реши линеарната равенка ax+b=0!

4.1.2 Улога на програмите во компјутерот

Интуитивно имаме претстава за тоа што е програма, веќе сме работеле во многу програми, на пр. за цртање, за пишување, за слушање музика, за играње итн. Програмите му кажуваат на компјутерот како да реши одреден проблем. Всушност, задачата на програмите е да му кажат на компјутерот како да ги прифати влезните податоци, како да оперира со нив и како да врати излезни податоци.

Програма е начин на запис на алгоритам во форма разбирлива за компјутерот. Алгоритмите се поопшти и поапстактни од програмите. Еден алгоритам може да биде решен од човек или машина, или од двете. Програмата мора да биде извршена од компјутер. Врската помеѓу програма, алгоритам и податоци е дадена во формулата:

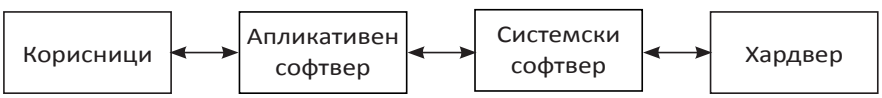
Програма = Алгоритам + Податоци

Програмите ги пишува човек и ги внесува во меморијата на компјутерот преку тастатура или преку други влезни единици.

Програма е низа од наредби кои се изведуваат по точно одреден редослед и со точно дефинирана цел. Наредба е основен елемент на програмата и претставува инструкција до компјутерот да изврши одредени дејства.

Наредба е темелен елемент на една програма. Со наредбите точно е дефинирано што и како компјутерот треба да направи. Со правилна употреба на програмите корисникот управува со компјутерот. Компјутерите можат да извршуваат најразновидни програми и корисникот лесно преоѓа од една на друга програма. На тој начин програмите обезбедуваат повеќекратна намена на хардверот за различни потреби.

Програмите се дел од софтверот. Софтверот го сочинуваат програми кои од една страна ја контролираат функционалноста на сите хардверски делови на компјутерот (системскиот софтвер), а од друга страна извршуваат обработка на податоци (апликативниот софтвер). На тој начин софтверот претставува врска помеѓу корисникот и компјутерот.



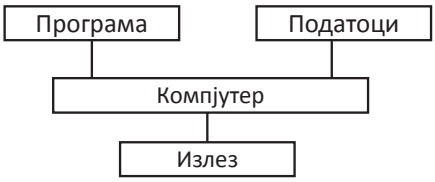
Сл. 4. 3 Улога на софтверот како врска помеѓу корисникот и хардверот

Системските програми ги пишуваат стручните тимови во големите софтверски компании како што се Microsoft, IBM и други, додека апликативните програми може да ги напише и поединец според потребите на корисникот.

Апликативните програми му овозможуваат на корисникот да изврши различни задачи од различни области, како што се обработка на текст, креирање на презентации, работа со бази на податоци, работи од сметководството, управување со производството, статистичките истражувања, забава и многу други работи. За секоја работа мора да постои посебна програма.

Поголемиот дел од корисниците на компјутерите користат веќе готови програми кои се инсталирани во компјутерот. Кога програма ќе се активира од некоја надворешна меморија, таа се вчитува во оперативната меморија на компјутерот и процесорот започнува со извршување на наредбите од програмата.

Влезот во компјутерот го сочинуваат две компоненти: програма и податоци. Компјутерот ги следи инструкциите дадени со наредби во некоја програма и извршува дејства врз податоците. Податоците претставуваат влез за програма. На пр. ако се собираат два броја, тогаш тие броеви се влезни податоци за програмата за собирање.



Сл. 4. 4 Поедноставен приказ на извршување на програмите

Резиме

Програма е низа од наредби кои се изведуваат по точно одреден редослед и со точно дефинирана цел. *Наредба* е основен елемент на програма и претставува инструкција до компјутерот да изврши одредени дејства.

Софтверот го сочинуваат програми кои, од една страна, ја контролираат функционалноста на сите хардверски делови на компјутерот (системски софтвер), а од друга страна извршуваат обработка на податоци (апликативен софтвер).

Влезот во компјутерот го сочинуваат две компоненти: програма и податоци. Компјутерот ги следи инструкциите дадени со наредби во некоја програма и извршува дејства врз податоците.

Податоците претставуваат влез за програма.

Прашања:

1. Што е програма?
2. Што е темелен елемент на програма? Зошто?
3. Која е задачата на програмите во компјутерот?
4. Дали програмите се дел од софтверот?
5. Опиши ја улогата на софтверот како врска помеѓу корисникот и хардверот!
6. Опиши ја врската помеѓу компјутерот, програмата и податоците!

4.2 Програмирање и програмски јазици

4.2.1 Програмски јазици

Со креирање на алгоритам се добиваат прецизни и недвосмислени инструкции за решавање на некоја задача. Сега е потребно тие инструкции да се пренесат на компјутерот на начин на кој тој може да ги разбере, односно потребно е да се напише програма според која компјутерот ќе ја изврши дадената задача.

Луѓето кои пишуваат програми се нарекуваат *програмери,* тие програмите ги пишуваат во *програмските јазици* – посебен вид на вештачки јазици развиени за комуникација помеѓу човекот и компјутерот. Постапката на запишување на инструкции преку наредби на некој програмски јазик се нарекува *програмирање*.

Забелешка:

Вештачките јазици се јазици кои се направени од поединец или од група луѓе за комуникација помеѓу луѓе кои не зборуваат со ист јазик, како и за комуникација помеѓу луѓето и машината или помеѓу две машини.

Од појавата на компјутерите до денес се развиени голем број програмски јазици кои обично се делат на

- нижи програмски јазици и
- виши програмски јазици.

Главните особини на нижите програмски јазици

Главната карактеристика на нижите програмски јазици е што тие се *машински ориентирани*, т. е. зависат од машина на која се изведуваат. За секој вид на процесор постои посебен нижи програмски јазик. Сите овие јазици се разликуваат меѓусебно. Ако се напише програма за еден процесор, не е сигурно дека таа програма ќе работи и на друг процесор.

Првите програмски јазици кои се појавиле се ***машински јазици*** (machine language, machine code). Програма пишувана на машински јазик е единствената програма која компјутерот ја разбира, па затоа секоја друга форма на програма мора да се преведе на машински јазик. Наредбите во машинскиот јазик се изразуваат со бинарен запис (низа од нули и единици). Таков запис за луѓето е многу тежок за разбирање. Пишување на програма во машинскиот јазик е сложено и бара добро познавање на градбата на компјутерот.

Набргу се развиени нови програмски јазици наречени ***асемблери*** или ***симболички јазици***. Програмите пишувани во симболички јазик се поблиски до човекот, секоја бинарна наредба е претставена со мнемоник (збор кој лесно се памти и потсетува на нешто), односно со симбол разбирлив за човек. Така, собирање се претставува со мнемоникот ADD, преместување на податоци се претставува со мнемоникот MOV итн. Со ова е олеснето пишување и читање на програми, но сè уште е потребно на компјутерот да се дадат и најмали упатства за секоја операција.

Типична наредба во симболички јазик изгледа вака:

```
ADD X Y Z
```

Оваа наредба значи: Бројот зачуван во мемориската локација со името X да се додаде на бројот зачуван во мемориската локација со името Y и резултатот да се зачува во мемориската локација со името Z.

Иако асемблерот е многу сличен со машинскиот јазик, оваа наредба треба да се преведе во низа од нули и единици, па наредбата во машинскиот јазик може да има форма:

0110 1001 1010 1011

Наредбите запишани во симболички јазик различно се преведуваат на машински јазик зависно од машината на која тие ќе се извршуваат. Преведувањето го извршува компјутерот преку посебна програма за преведување наречена асемблер. Секој компјутер има свој симболичен јазик и свој преведувач.

Главни особини на вишите програмски јазици

Пишување програми во машинските и симболичките јазици претставува голем проблем заради непреносливост, а освен тоа за луѓето е неразбирлив и е тежок за учење.

Решение на овој проблем е најдено во вишите програмски јазици кои со тек на време луѓето ги осмислувале и развивале. Денес постојат многу виши програмски јазици, некои од нив се: BASIC, Pascal, C, C++, C#, Java, Prolog, SmallTalk, Modula 2, FORTRAN, LISP, ADA.

Главните особини на вишите програмски јазици се можност да се извршат на компјутери со различни процесори и разбирливост бидејќи се доста слични со природниот (англискиот) јазик. На пр. за компјутерот да собере два броја и резултатот да го запомни на некое трето место во меморијата, се пишува наредба од тип C=A+B што значи: собери ги броевите што се запишани во A и B, а резултатот зачувај го во C.

Еве како изгледа програма за собирање на два цели броеви во некои од вишите програмски јазици:

QBASIC:
`INPUT A`
`INPUT B`
`C=A+B`
`PRINT C`
`END`
	LOGO:
	`TO SOBERI`
	`MAKE "A READ`
	`MAKE "B READ`
	`MAKE "C :A+:B`
	`PR :C`
	`END`

PASCAL:
`program soberi;`
`var a, b: integer;`
`begin`
`readln (a);`
`readln (b);`
`c:=a+b;`
`writeln (c);`
`end.`
	C:
	`#include <stdio.h>`
	`main ()`
	`{`
	`int a,b,c;`
	`scanf ("%d,%d", &a, &b);`
	`c=a+b;`
	`printf ("%d", c);`
	`}`

Секој програмски јазик користи сопствено ограничено множество од зборови и симболи со помош на кои се запишуваат наредби во програма. Како што во природните јазици постојат правила за конструкција на реченицата, така и во програмските јазици постојат правила со кои се опишува начинот на конструирање на наредби. Множеството од овие правила се нарекува *синтакса* на јазикот.

4.2.2 Процес на изработка на една програма

Програмерите своите програми ги пишуваат во некој од вишите програмски јазици, меѓутоа, компјутерот може да изврши само програма напишана на машински јазик. Преведувањето на програмите во машински јазик се изведува на два начини:

- со помош на програмите за преведување при што се креира извршна верзија на програма
- со помош на програмите за интерпретирање.

Креирање на извршна програма

Програма напишана на некој од вишите програмски јазици се нарекува *изворна програма* или *изворен код* (source code). Програма која ја извршува компјутерот се нарекува *извршна програма* или *извршен код* (executable code). Претворањето на изворниот код во извршен код се врши во два чекора со помош на посебни системски програми наречени *преведувачи* и *поврзувачи*.

Значи, процесот на пишување на програмата е сложен процес кој се состои од четири фази:

1. пишување на изворен код,
2. преведување на изворен код,
3. поврзување во извршен код,
4. тестирање на програма.

Пишување на изворен код

Првата фаза е пишување на изворен код или *кодирање*. Откако ќе се заврши пишувањето на изворниот код, тој се зачувува во датотека на изворниот код на дискот. На оваа датотека обично ѝ се дава некое описно име и ѝ се доделува наставка соодветна со програмскиот јазик во кој е пишувана. За датотеките пишувани во програмскиот јазик C++ обично им се доделува наставката .cpp, на пример sobiranje.cpp.

Преведување на изворен код

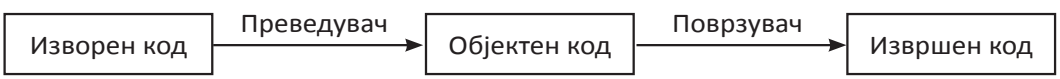
Втората фаза е преведување на изворниот код со помош на програмите преведувачи или компајлери (compilers). Преведувачот ја проверува синтаксата на изворниот код и во случај на воочени грешки испишува соодветни пораки за грешките. Овие грешки се нарекуваат *синтаксни грешки* или *грешки при преведување* (compile-time errors). Синтаксните грешки се однесуваат на неправилно напишани зборови од програмскиот јазик, неправилно користени или испуштени интерпункциски знаци, неправилно користење на загради и сл.

По добивањето пораки за грешки, програмерот се обидува истите да ги поправи и повторно го преведува изворниот код. Дури кога сите грешки ќе бидат отстранети, процесот на преведување може успешно да се реализира. Со преведување на изворен код се добива датотека од *објектен код* (object code) која вообичаено ја има наставката .o или .obj (во нашиот пример sobiranje.obj).

Поврзување во извршен код

По преведувањето следува третата фаза – поврзување на објектни кодови во извршен код со помош на програмите поврзувачи (linkers). Објектниот код добиен со преведување треба да се поврзе со постоечките датотеки во кои се наоѓаат веќе преведени мали програми кои често се користат во програмите. Овие датотеки се нарекуваат *библиотеки* (libraries).

Ако кодот не може да се поврзе со потребните библиотеки, поврзувачот ќе јави порака за грешка и извршниот код нема да се креира. Овие грешки се нарекуваат *грешки при поврзување* (link-time errors). Програмерот мора да ги исправи овие грешки и повторно да се обиде да изврши и преведување и поврзување. Со успешно поврзување се добива извршен код, односно датотека на која ѝ се доделува наставката .exe. Извршна датотека може да се користи самостојно на кој било компјутер.



Сл. 4. 5 Шематски приказ на процес на изработка на програма

Тестирање на програма

Добивање на извршна датотека сè уште не е гаранција дека програмата ќе работи онака како што тоа е замислено. Затоа се изведува и последната фаза – тестирање на

програмата. Грешки откриени на овој начин се нарекуваат *грешки при изведување* (run-time errors) или *логички грешки*.

Овие грешки се последица на погрешно изработен алгоритам и компјутерот не може да ги открие. Затоа програмерот треба да ја тестира програмата за различни податоци и ваквите грешки (доколку ги има) да ги воочи и да ги поправи. По исправање на грешките целата постапка треба да се повтори од почеток: поправање на изворниот код → преведување → поврзување → тестирање.

Пр. 4. 7. Пример за синтаксна и за логичка грешка

Ако формулата за пресметување на периметар на правоаголник се запише на следниот начин:

L=2*(a+b

направена е синтаксна грешка – недостасува заграда на крајот од изразот. Оваа грешка преведувачот ќе ја открие и ќе ни јави порака.

Но, ако формулата се запише како:

L=2*(a-b)

направена е логичка грешка – формулата не е точна и нема да даде точен резултат. Во овој случај нема синтаксна грешка и преведувачот нема да даде предупредување.

Забелешка:

Ова е само пример од кој треба да воочиш разлика помеѓу синтаксна и логичка грешка. Повеќе за синтаксните и за логичките грешки и за нивното откривање ќе зборуваме подоцна.

Интерпретери

Понекогаш изворниот код се извршува без тој да се преведе во извршен код. За ова се користат други системски програми наречени *интерпретери*. Интерпретерот не ја преведува целата изворна програма и не прави извршна верзија, туку наредбите една по една ги преведува на машински јазик и веднаш ги извршува. На тој начин изворниот код мора да биде присутен во меморијата на компјутерот додека програмата се извршува. Тоа значи дека изворниот код е достапен на сите и авторот не е заштитен од евентуалната кражба. Од друга страна, грешките полесно се воочуваат и кодот може веднаш да се исправи, па овој тип на преведувачи се користи за проверка на пробни верзии на програмите.

Резиме

Луѓето кои пишуваат програми се нарекуваат *програмери*. Постапката на запишување на инструкции преку наредби на некој програмски јазик се нарекува *програмирање*.

Програмите се пишуваат во посебни вештачки јазици осмислени и конструирани за комуникација помеѓу човек и компјутер – *програмски јазици*. Програмските јазици ги делиме на нижи програмски јазици и на виши програмски јазици.

Нижите програмски јазици се машински ориентирани. Прво се појавиле машински јазици во кои програмите се пишувани со бинарни цифри, набргу се развиени нови програмски јазици наречени асемблери или симболички јазици. Програмите пишувани во асемблер се поблиски до човекот. Вишите програмски јазици можат да се извршат на компјутери со различни процесори и се доста слични со природниот јазик.

Правилата со кои се опишува начинот на конструирање на наредби се нарекуваат *синтакса* на јазикот.

Компјутерите ги разбираат и можат да ги извршат само програмите напишани на машински јазик. Програмата која ја извршува компјутерот се нарекува *извршна програма* или *извршен код*. Програмата напишана на некој од вишите програмски јазици се нарекува *изворна програма* или *изворен код*.

Процесот на изработка на програмата се состои од четири фази: 1. пишување на изворен код, 2. преведување на изворен код, 3. поврзување во извршен код и 4. тестирање на програма.

Прашања:

1. Како се нарекува постапка на изработка на програми?
2. Како се нарекуваат јазици во кои се пишуваат програми?
3. Како се делат програмските јазици?
4. Кои се нижи програмски јазици?
5. Наведи неколку виши програмски јазици!
6. Направи споредба помеѓу нижите и вишите програмски јазици?
7. Што е синтакса на програмски јазик?
8. Како се нарекува програма напишана на некој од вишите програмски јазици?
9. Како се нарекува програмата која ја извршува компјутерот?
10. Од кои фази се состои постапката за креирање на извршна верзија на програма?
11. Накратко опиши ги фазите во постапката за креирање на извршна верзија на програма!
12. Која е улога на програмите за преведување?
13. Каква датотека се добива по процесот на преведување?
14. Која е улога на програмите за поврзување?
15. Што се библиотеки?
16. Што се синтаксни, а што се логички грешки?
17. Што треба да се направи по корекција на грешките?
18. Најди синтаксна и логичка грешка во формулата за пресметување на средна вредност на два броја: sv = a - b) / 2.
19. Обиди се да направиш блок дијаграм за процесот на изработка на програма!
20. Што е интерпретер?
21. Која е разликата помеѓу преведување на програма со помош на преведувачи и поврзувачи и преведување на програма со помош на интерпретери?

4.2.3 Интегрирана околина за програмирање

Изворен програмски код може да се напише во било кој текст едитор, но постојат посебни програми во кои пишувањето на програмите е многу олеснето. Денес воглавно се користат програми во кои се вградени и поврзани: едитори за пишување на изворен код, програми за преведување (компајлирање), библиотеки на готови програми и програми за наоѓање на грешки (дебагирање). Ваквите програми се нарекуваат *интегрирани развојни околини* (IDE – integrated development environment) за програмски јазик.

За секој програмски јазик постојат неколку интегрирани развојни околини. Ние ќе програмираме во програмскиот јазик C++, а како интегрирана развојна околина ќе ја користиме програмата Code Blocks.

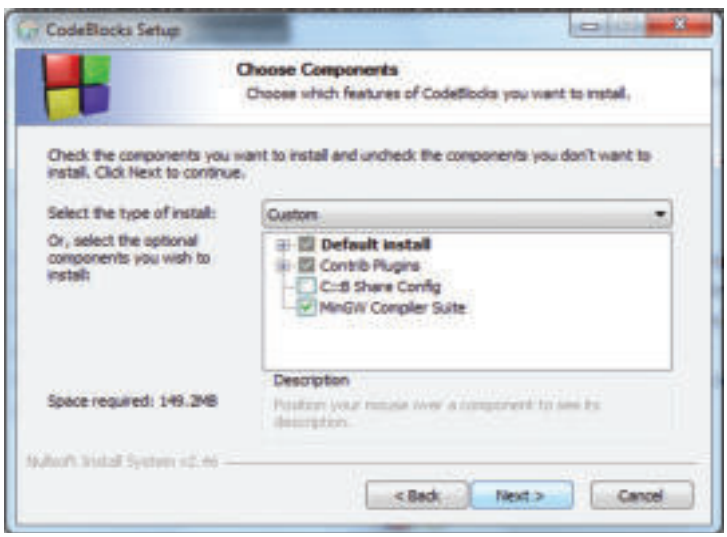
Програмските јазици C и C++ се едни од најпопуларните праграмски јазици за општа намена. Овие јазици имаат големи можности, делотворни се и прилагодливи.

Инсталирање на програмата Code Blocks

Code Blocks е бесплатна околина за C++ со отворен код во која се интегрирани едитор за текст, програми за преведување и поврзување, како и програми за откривање грешки. Значи, има сè што е потребно за пишување на изворна програма и за нејзино трансформирање во извршна програма.

Програмата Code Blocks може да се преземе преку линкот: http://www.codeblocks.org/downloads/binaries/. Во момент на пишувањето на учебников последната верзија на програмата за оперативниот систем Windows е: codeblocks-12.11mingw-setup.exe, а за оперативниот систем Linux: codeblocks-12.11-1-debian-dbg-i386.tar.bz2.

Кликни на еден од понудените линкови за преземање и преземи ја програмата. Потоа изврши инсталација. Инсталацијата се одвива на вообичаен начин. Сите опции остави ги како што се предложени.

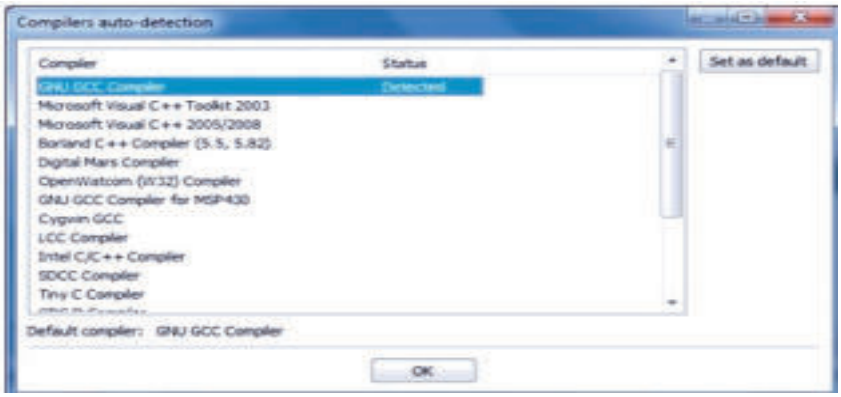


Сл. 4. 6 Инсталација на развојната околина Code Blocks (чекор 1)

Избери програмата веднаш да се стартува:

Сл. 4. 7 Инсталација на развојната околина Code Blocks (чекор 2)

На крај, означи ја првата опција и кликни на копчето *OK*:



Сл. 4. 8 Инсталација на развојната околина Code Blocks (чекор 3)

Прилагодување на работната околина

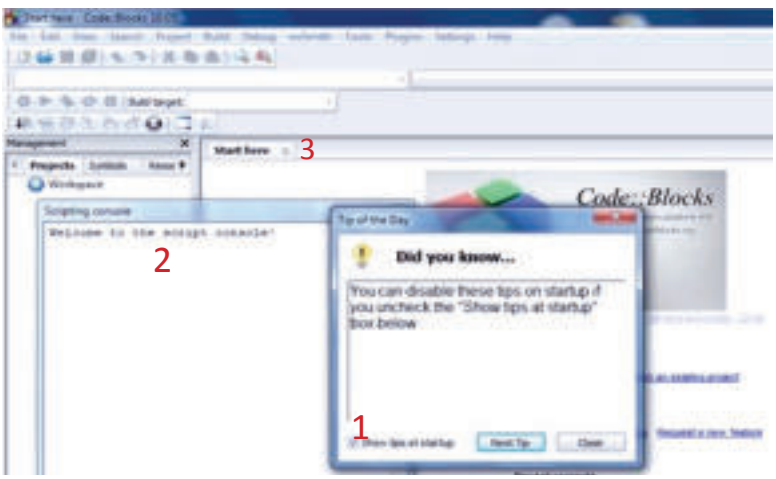
По стартување на програмата, ќе добиеш прозорец во кој, за прилагодување на работната околина:

1 Тргни го знакот за потврда од опцијата *Show tips at startup* и кликни на копчето *Close*;

2 Затвори го прозорецот *Scripting Console*;

3 Кликни на знакот *X* до копчето *Start Here*;

4 Во менито *View* во опцијата *Toolbars* остави го знакот за потврда само на *Compiler*.



Сл. 4. 9 Почетен прозорец на програмата Code Blocks

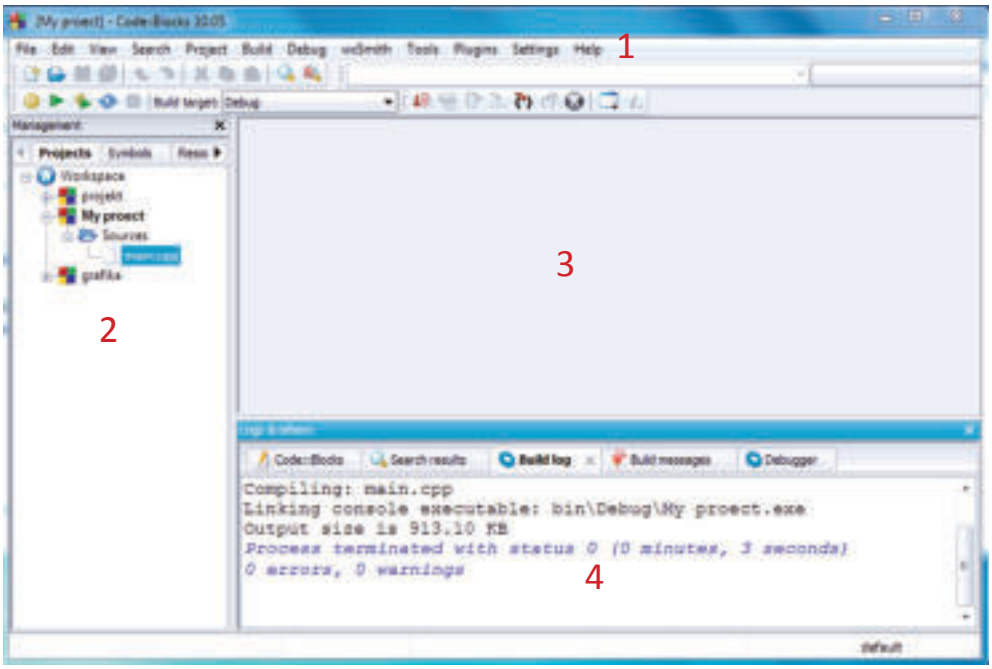
Сега работната околина опфаќа:

1 Мени со наредби,

2 Лентата Management,

3 Текст едитор за внесување на изворниот код,

4 Прозорец за испишување на пораки за грешки.



Сл. 4. 10 Прилагодена работна околина на програмата Code Blocks

Креирање, преведување и извршување на програма

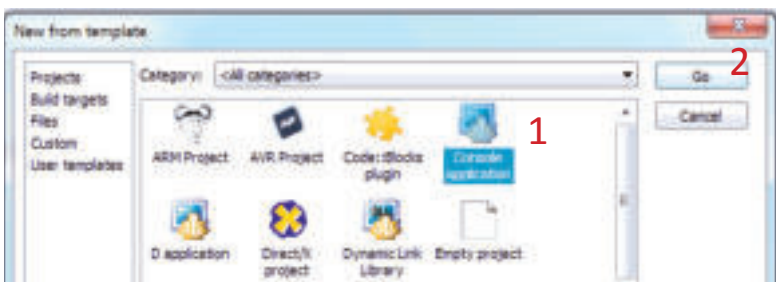
Програмите во C++ можат, но не мора да бидат дел од проект.

Креирање проект

Нов проект во CodeBlocks се креира со наредбата *File→New Project* по што се отвора прозорецот *New from template.*

1 Одбери *Console Application* и

2 кликни на копчето *Go*.



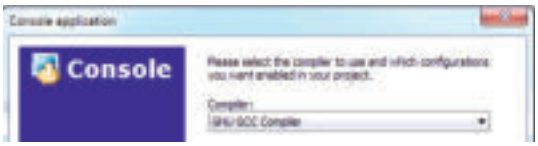
Сл. 4. 11 Креирање на нов проект (чекор 1)

Во следниот прозорец кликни на копчето *Next*, потоа одбери C++ и повторно кликни на копчето *Next*.

Сл. 4. 12 Креирање на нов проект (чекор 2)

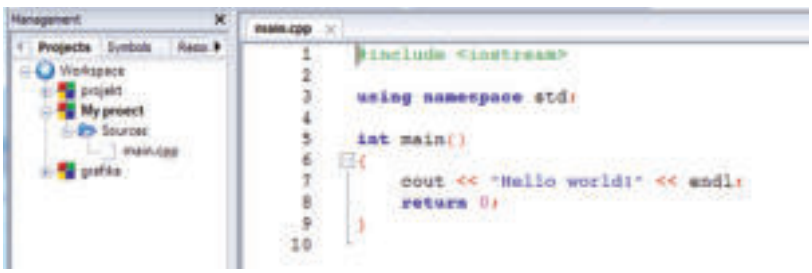
Внеси име на проектот во полето *Project Title*, потоа кликни на копчето *Browse* (означено со три точки) и во прозорецот *Browse for Folder* избери локација каде ќе го зачуваш проектот. Останатите полиња ќе се нагодат автоматски.

Кликни на копчето *Next*, провери дали полето *Compiler* ја содржи вредноста "GNU GCC Compiler", и кликни на копчето *Finish.*



Сл. 4. 13 Креирање на нов проект (чекор 3)

Кога ќе се креира нов проект тој содржи датотека main.cpp. Во левиот дел на екранот, отвори *Workspace,* потоа твојот проект, па *Sources* и отвори ја датотеката main.cpp (со кликнување двапати на левото копче од глушецот, или со кликнување на десното копче од глушецот и *Open main.cpp*). Тоа е програма која стандардно ја пишуваат сите кои започнуваат со учење на програмирање.



Сл. 4. 14 Датотеката main.cpp

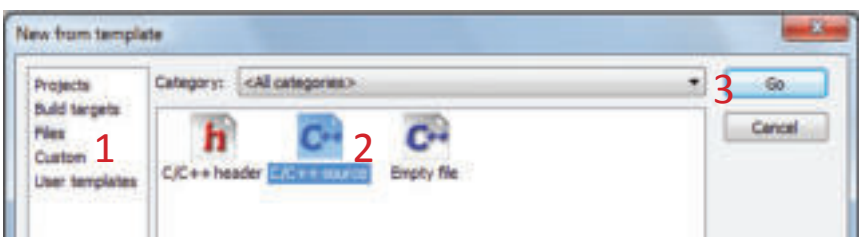
Креирање на нова датотека на изворен код

Нова датотека може да се креира во рамките на постоечки проект или самостојно. Еден од начините за креирање на нова датотека на изворен код е преку наредбата *File→New→File*. Во прозорецот *New from template*

1 избери *Files*,

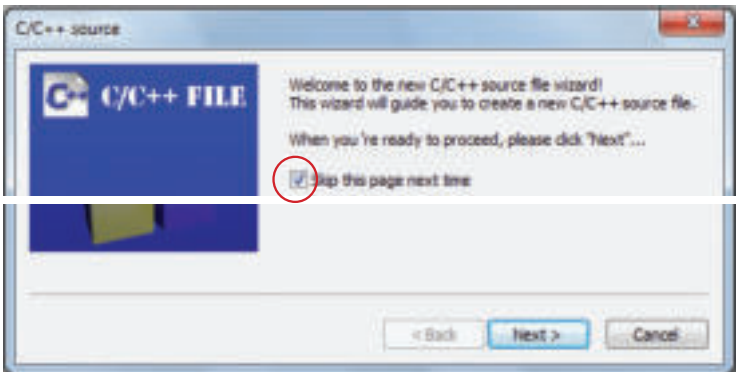
2 потоа избери го типот на датотека *C/C++ source*, и

3 кликни на копчето *Go*.



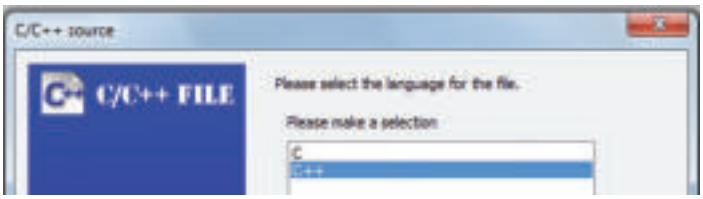
Сл. 4. 15 Креирање на нова датотека на изворен код (чекор 1)

Се покренува помошник за креирање на нова датотека. Доколку не сакаш овој прозорец и понатаму да се појавува потврди ја опцијата *Skip this page next time* и кликни на копчето *Next*.



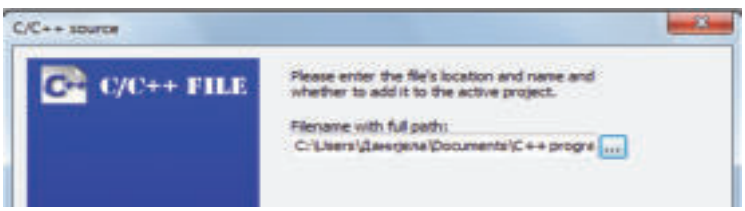
Сл. 4. 16 Креирање на нова датотека на изворен код (чекор 2)

Во следниот прозорец избери го јазикот *C++* и кликни на копчето *Next*.



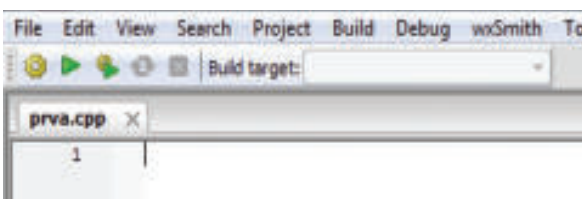
Сл. 4. 17 Креирање на нова датотека на изворен код (чекор 3)

Потоа кликни на копчето *Browse* (означено со три точки) по што ќе се појави прозорецот *Save as...* Избери локација каде ќе ја зачуваш новата датотека и дај ѝ име на датотеката. На крај кликни на копчето *Finish*.



Сл. 4. 18 Креирање на нова датотека на изворен код (чекор 4)

Со ова е креирана нова датотека со името *prva.cpp*:



Сл. 4. 19 Креирање на нова датотека на изворен код (чекор 5)

Совет:

Креирај посебен фолдер во кој ќе ги чуваш твоите програми напишани во јазикот C++.

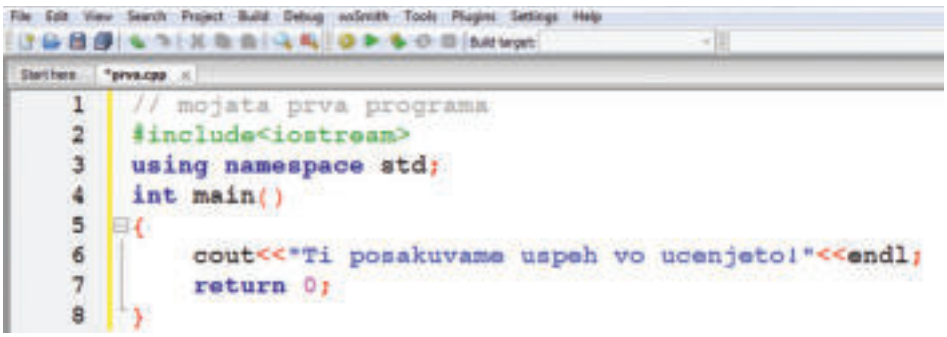
4.2.4 Преведување и извршување на програма

Кога е креирана датотека на изворниот код, може да се почне со пишување на програма. Програмата се пишува во едитор. Препиши ја следнава содржината на датотеката на изворниот код:

Овде е нула

```
// mojata prva programa
#include<iostream>
using namespace std;
int main()
{
    cout<<"Ti posakuvame uspeh vo ucenjeto!"<<endl;
    return 0;
}
```

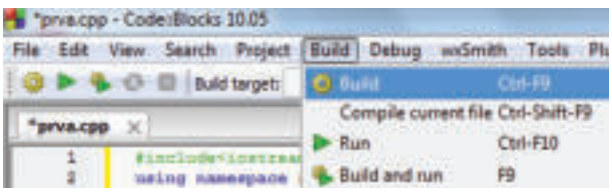
Не грижи се што не ја познаваш структурата на програмата и на поединиите наредби. Сега веројатно програмирањето ти изгледа застрашувачки, но набргу сè ќе биде појасно. Изворниот код ќе изгледа како на сликата:



Сл. 4. 20 Изворен код во едиторот на програмата Code Blocks

Линиите се означени со броеви и пред себе имаат жолта вертикална линија. Тоа значи дека тие линии сè уште не се преведени. Преведените линии пред себе имаат зелена вертикална линија. Ова е од корист кога се прават измени во програма.

Откако ќе се напише изворна датотека таа треба да се преведе и да се поврзе во извршен код. Тоа се постигнува со наредбата *Build→Build* (Кратенка *Ctrl + F9*).



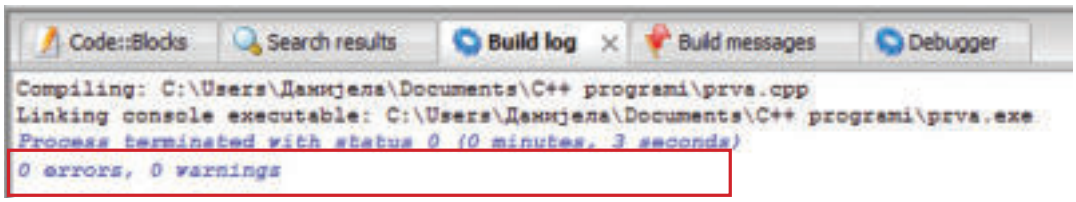
Сл. 4. 21 Наредба за градење на извршна верзија

Забелешка:

Фазите преведување и поврзување се одвиваат како една фаза која се нарекува преведување или компајлирање.

Ако постојат синтаксни грешки или предупредувања, тие ќе се испишат во посебна рамка (*Build log*) во долниот дел на прозорецот. Грешките треба да се исправат

и програмата да се преведе повторно. Кога преведувањето е успешно завршено, преведувачот дава порака како на следната слика:



Сл. 4. 22 Порака за успешно преведена програма

На крајот програмата треба да се изврши со наредбата *Build→Run* (кратенка *Ctrl + F10*).

Како последица на целиот процес, во папката има три датотеки:

Сл. 4. 23 Икони за изворна, за извршна и за објектна датотека во C++

Важно!

Програмата може да се изврши само ако пред тоа е преведена.

Извршната датотека се извршува во посебен прозорец:

Сл. 4. 24 Прозорец во кој се извршува програма

Нашата програма го испишува текстот „Ti posakuivame uspeh vo ucenjeto!“. Текстот „Press any key to continue“ го генерира самиот компјутер и со тоа не известува дека програмата е извршена и дека со притискање на било кое копче од тастатурата прозорецот ќе се затвори.

Забелешка:

За побрз пристап до наредбите можат да се користат копчињата Build, Run и Build and Run од лентата со алатки за преведување: 

Резиме

Програмите во кои се вградени и обединети програми за пишување на изворен код, програми за преведување, програми за поврзување и програми за наоѓање грешки се нарекуваат *интегрирани развојни околини*. Ние за програмскиот јазик C++ ќе ја користиме програмата Code Blocks. Кога е креирана датотека на изворниот код, може да се почнеме со пишување на програма. Програма се пишува во едиторот.

Откако ќе се напише, изворната датотека треба да се преведе и да се поврзе во извршен код. Кога преведувањето успешно е завршено, програмата треба да се изврши. Програмата се извршува во посебен прозорец. Како последица на целиот процес, во папката има три датотеки: изворна, објектна и извршна датотека.

Вештини што треба да ги усовршиш:

Да креираш нова датотека.
Да користиш едитор за пишување на изворен код.
Да преведеш изворна датотека во извршна датотека.
Да покренеш извршување на програма.

Прашања:

1. Што е интегрирана развојна околина?
2. За кои развојни околини за програмскиот јазик C++ си слушнал/а?
3. Инсталирај ја програмата Code Blocks и подеси ја работната околина?
4. Со која наредба се креира нова изворна датотека во Code Blocks?
5. Каде се пишува изворен код?
6. Со која наредба се преведува изворна датотека?
7. Кои датотеки се креираат по преведувањето?
8. Како знаеш дека програма е успешно преведена?
9. Со која наредба се извршува програма? Каде програмата се извршува?

Задача:

1. Напиши, преведи и изврши едноставна програма со која ќе се отпечати порака по твој избор! Колку и какви датотеки се креирани во твојата папка?

4.2.5 Извршување и изглед на готови пример програмски кодови

Погледни го кодот на програмата prva.cpp. Дали разбираш некои зборови и реченици? Дали ти е јасно што работи програмата? Дали препознаваш наредба која на компјутерот му кажува да испише соодветен текст на мониторот?

Да го разгледаме и анализираме изворниот код на програмата *prva.cpp*:

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"Ti posakuvame uspeh vo ucenjeto!"<<endl;
    return 0;
}
```

Ова е една од наједноставните програми, но сепак ги содржи основните делови кои секоја програма напишана во C++ мора да ги има.

Изворниот код започнува со следниве линии:

```
#include<iostream>
using namespace std;
int main()
{
```

а завршува со линиите:

```
    return 0;
}
```

Овие линии се пишуваат на почетокот и на крајот на секоја програма. Ќе анализираме линија по линија.

`#include <iostream>`

Линиите кои започнуваат со знакот „#“ преведувачот не ги преведува и тие се нарекуваат *претпроцесорски наредби*. Овие наредби на преведувачот му кажуваат пред преведувањето да вклучи и некоја додатна датотека. Наредбите во C++ се организирани во посебни датотеки, кои се нарекуваат *библиотеки*. Со изразот `#include` на преведувачот му се најавува од кои библиотеки ќе се користат наредби. Преведувачот тие библиотеки ќе ги вклучи во програмата пред да започне со преведувањето. Името на библиотека се става помеѓу знаците „<>“. Библиотеката `iostream` е стандардна библиотека која овозможува внесување на податоци преку тастатура и испишување на резултати преку монитор.

`using namespace std;`

Постојат многу библиотеки па може да се случи во две различни библиотеки да се најдат наредби со исто име а со различни намени. Сите елементи на стандардните C++ библиотеки се декларирани во просторот што се нарекува `namespace` (простор со имиња). Со оваа наредба на преведувачот му се кажува дека ќе се користат наредби од просторот std, што е кратенка од standard.

`int main ()`

Оваа линија означува дека тука започнува самата програма. Она што е пишувано порано не се смета за програма туку за упатства на преведувачот како да ја преведе програмата, на пр. кои библиотеки да користи при преведувањето.

`main` е име за главната функција која мора да ја има секоја C++ програма. Со оваа функција започнува извршување на сите прогрми во C++ . По зборот `main` стојат загради `()` што значи дека се работи за дефиниција на функција.

`{`

Голема отворена заграда означува почеток на дел од програма во кој се наоѓаат искази (наредби) на главната функција.

`return 0;`

Овој исказ е порака до оперативниот систем дека програмата успешно е извршена. По зборот `return` се става повратна вредност, во нашиот случај тоа е нула.

`}`

Голема затворена заграда се става на крајот на секоја програма и го означува крајот на главната функција.

`cout << “Ti posakuvame uspeh vo ucenjeto!“<<endl;`

Ова е единствената линија која во нашата програма произведува видлив ефект. Таа во C++ претставува *исказ* или *наредба*. Со овој исказ на мониторот се испишува пораката „Ti posakuvame uspeh vo ucenjeto!“.

Важно!

По секој изказ се става знакот точка и запирка „;“ што е знак дека исказот тука завршува. Претпроцесорските наредби не завршуваат со знакот точка и запирка. Исто така, и по линијата int main () не се става знакот точка и запирка.

Изглед на програма – вовлекување (индентација)

Во C++ нема строги правила по кои секој исказ мора да се пишува во нова линија. Така може да се напише и:

```
int main () {cout<<"Ti posakuvame uspeh vo ucenjeto"<<endl; return 0;}
```

За преведувачот оваа линија е исправна бидејќи за крај на исказот се смета знакот „;“, а не крајот на линијата. Исказите се пишуваат во нови линии за програмите да бидат попрегледни. Од тие причини овие искази се малку вовлечени во однос на почетокот и на крајот на функцијата. Ваквата техника на вовлекување на секвенци од искази кои сочинуваат логичка целина се нарекува *вовлекување* (*индентација*).

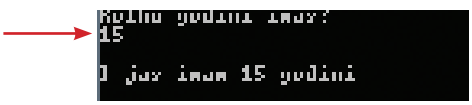
Зад. 4. 5. Следниот код напиши го така да биде полесен за читање. Испиши го на повеќе линии и користи индентација!

```
#include <iostream>
using namespace std;
int main ()
{int x;cout<<"Bravo";return 0;}
```

Пр. 4. 8. Препиши го кодот и зачувај ја датотеката со името *vtora.cpp*:

```
#include<iostream>
using namespace std;
int main()
{
    int god;
    cout<<"Kolku godini imas? ";
    cin>>god;
    cout<<"I jas imam "<<god<<" godini"<<endl;
    return 0;
}
```

Преведи ја и изврши ја програмата. На екранот прво ќе ја добиеш пораката „Kolku godini imas?“. Во новиот ред, на местото на покажувачот, внеси го бројот на твоите години и притисни го копчето *Enter*. Сега имаш ваков изглед на екранот (црвената стрелка покажува на текстот што корисникот го внесува):



Забележуваш дека во исказот `cout<<"Kolku godini imas?"<<endl;` е додаден изразот `endl` (скратено од end line) што предизвикува поминување на покажувачот во нов ред.

Исто така, се јавува и нов исказ:

`cin>>god;`

Овој исказ прифаќа податоци внесени преку тастатурата. Во овој случај тоа е бројот 15.

Забелешка:

Исказите cout и cin се декларирани во библиотеката iostream во рамки на просторот std. Тие се причина што мора да се користат првите две линии во програмата.

Ќе разгледаме уште еден исказ кој често се користи во C++ програмите

```
system ("PAUSE");
```

Со овој исказ на компјутерот му се укажува да застане и да чека додека не се притисне некое копче на тастатурата. За да ја разбереш оваа наредба изврши ја програмата *prva.exe* преку иконата од твојата папка; ќе забележиш дека прозорецот во кој се извршува програмата веднаш ќе се затвори пред да ја видиш пораката.

Овој исказ се пишува пред исказот `return 0;` и за негово користење мора да се вклучи библиотеката `cstdlib` преку претпроцесорската наредба `#include <cstdlib>`. Библиотеката cstdlib е библиотека на стандардни наредби.

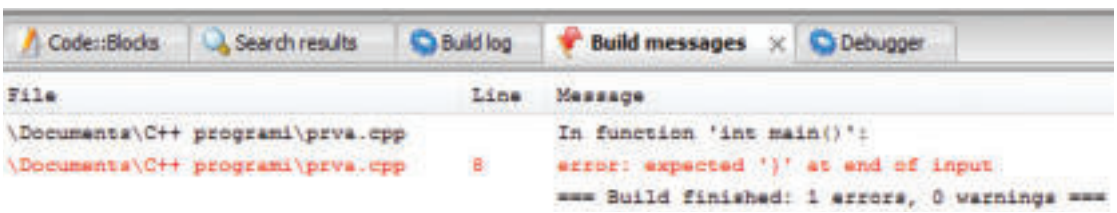
Пронаоѓање и исправање грешки – дебагирање

Во изработка на програми често се јавуваат грешки кои е потребно да се пронајдат и да се исправат. Компјутерските грешки се нарекуваат багови (buggs), па така и процесот на пронаоѓање и на исправање на грешки се нарекува дебагирање (debugging). Овој процес е неопходен во сите нови програми и често е потребно да се повтори повеќе пати. Современите програми се состојат од голем број наредби и често се случува грешките да се откријат дури кај крајните корисници. Од тие причини компаниите кои произведуваат софтвер издаваат пробни (бета) верзии на програмите пред програмата да ја пуштат во продажба.

Да разгледаме што ќе се случи ако во програмскиот код на програмата *prva.cpp* се направи некоја грешка. Напиши го кодот без последната голема заграда:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      cout<<"Ti posakuvame uspeh vo ucenjeto!"<<endl;
7      system("PAUSE");
8      return 0;
```

При преведувањето, преведувачот ќе воочи дека функцијата `main()` не е правилно затворена и во посебна рамка (Build messages) ќе испише порака за грешка:



Сл. 4. 25 Порака за грешка

Во оваа порака бројот 8 укажува дека грешката се јавила во линијата со реден број 8. Пораката по зборот „error“ ни кажува за каков вид на грешка се работи.

Забелешка:

Во пораки за синтаксна грешка бројот на линија укажува на линија во која преведувачот ја забележал грешката, често тоа е линија по или пред исказот во кој грешката е направена.

Зад. 4. 6. Во програмата *prva.cpp* избриши ја отворената голема заграда и набљудувај што ќе се случи. Истото направи го со малите загради и со наводниците.

Синтаксните грешки можат да ги откријат програмите за преведување за што ќе не известат. Додека не се отстранат сите синтаксни грешки програмата нема да се преведе.

Логичките грешки настануваат како последица на погрешно изработен алгоритам и се причина програмата да не ја извршува задачата точно. Програмерот мора самиот да открие постоење на логички грешки. Компјутерот тука не може да му помогне, бидејќи тој „не знае“ која задача програмерот сака да ја реши.

Забелешка:

За откривање на логичките грешки се користат програми наречени дебагери кои овозможуваат извршување на програмата линија по линија и стопирање на нејзиното извршување на одредено „сомнително“ место.

Пр. 4. 9. Ако програмерот во програмскиот јазик C++ исказот за пресметување на плоштина на круг го напише во следната форма:

```
P=r+r*3.14
```

тој ќе ја добие следната порака за синтаксна грешка: „In function ’int main()’ error: expected ’;’ before...“, бидејќи на крајот на наредбата не го ставил знакот точка и запирка.

Ако програмерот ја исправи грешката и напише:

```
P=r+r*3.14;
```

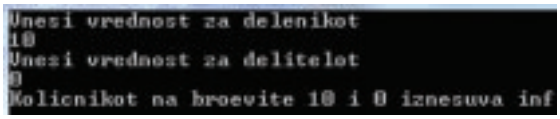
тој нема да добие порака за грешка бидејќи наредбата синтаксно е исправна. Меѓутоа, со оваа наредба плоштината на круг нема точно да се пресмета (формулата за пресметување на плоштина на круг е P=r*r*3.14).

Логичка исправност на програмите се обезбедува со проверки и со тестирање. Програмата воглавно се тестира за влезните податоци за кои однапред е познат резултатот или за кои резултатот може лесно да се провери на друг начин (рачно пресметување). При тоа треба да се води сметка програмата да се тестира за различни податоци.

Пр. 4. 10. Со следнава програма се внесуваат два броја и се пресметува нивниот количник:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a,b;
    cout<<"Vnesi vrednost za delenikot"<<endl;
    cin>>a;
    cout<<"Vnesi vrednost za delitelot"<<endl;
    cin>>b;
    cout<<"Kolicnikot na broevite ";
    cout<<a<<" i "<<b<<" iznesuva "<<a/b<<endl;
    system("PAUSE");
    return 0;
}
```

Преведи ја и изврши ја програмата. За деленикот внеси вредност 10, а за делителот внеси вредност 0. Програмата нема да даде точен резултат.



Причина за тоа е што делењето со 0 не е дефинирано и програмата не знае како да се однесува во тој случај. Програмерот направил превид во алгоритамот и не го предвидел овој случај.

Примери за логички грешки се: ако за должината на страна на геометриска фигура се внесе 0 или негативен број, ако за страните на триаголник се внесат должини со кои не може да се формира триаголник, ако за хипотенуза се внесе помала вредност отколку за катета и сл. Ваквите и слични ситуации програмерот мора да ги земе предвид.

Резиме

Линиите кои започнуваат со знакот „#“ преведувачот не ги преведува и тие се нарекуваат претпроцесорски наредби. Со изразот *#include* на преведувачот му се најавува од кои библиотеки ќе се користат наредби. Библиотека *iostream* е стандардна библиотека која овозможува внесување на податоци преку тастатура и испишување на резултати преку монитор. Со исказот *using namespace std;* на преведувачот му се кажува дека ќе се користат наредби од просоторот *std*, што е кратенка од *standard*.

main е име на главната функција која мора да ја има секоја C++ програма. Искази кои ја сочинуваат функцијата *main* се стават помеѓу големите загради „*{*“ и „*}*“. Секој исказ завршува со знакот точка и запирка „*;*“. Исказот r*eturn 0;* е порака до оперативниот систем дека програмата успешно е извршена.

Техниката на вовлекување на секвенци од искази кои сочинуваат логичка целина се нарекува *индентација*.

Процесот на пронаоѓање и на исправање на грешки се нарекува *дебагирање*. Кога преведувачот ќе воочи синтаксна грешка, тој во посебна рамка (*Build messages*) ќе испише порака за видот на грешката и за линијата во која грешката е воочена. Логичките грешки ги открива програмерот преку тестирање на програмата, во што можат да му помагаат посебни програми наречени *дебагери*.

Вештини што треба да ги усовршиш:

Да препознаваш структура на програма.

Да употребуваш индентација.

Да внесеш податоци преку тастатурата.

Да препознаваш пораки за синтаксни грешки.

Прашања:

1. Со кои линии започнува, а со кои линии завршува програмскиот код во C++?
2. Како се нарекуваат линиите кои започнуваат со знакот #? Како преведувачот се однесува кон овие линии?
3. За што се користи изразот `#include`?
4. Како се нарекува стандардна библиотека која овозможува внесување на податоци и прикажување резултати?

5. Која е улогата на исказот `using namespace std;`?
6. Како се нарекува главната функција во C++?
7. Како се одвојуваат исказите еден од друг?
8. Како оперативниот систем ќе знае дека програмата успешно завршила со работа?
9. За што се користи исказот `system ("PAUSE");`? Дали е задолжително овој исказ да се користи во програмите?
10. Дали мора секој исказ во програмата да се пишува во нова линија?
11. Што е индентација?
12. Зошто програмерите ја користат техниката на индентација?
13. Што е дебагирање?

Задачи:

1. Пополни ги празните места во следниот код:

```
#include <_________>
_________ <cstdlib>
using _____________ std;
int _____()
{
      cout<<"Zdravo!"___endl;
      ____<<"Kako si?"<<endl;
      system("______");
      return __;
}
```

2. Уреди го следниот код на начин тој да биде прегледен и лесен за читање:

```
      #include <iostream> #include <cstdlib>
using namespace std;
int main() {int god; cout<<"Hello
      world!"<<endl;
cout<<"I'm C++"<<endl;
  system("PAUSE"); return 0; }
```

3. Дали во следната програма е направена синтаксна или логичка грешка? Најди ја грешката!

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
      cout<<"Za 100 evra mozes da dobies ";
      cout<<100/61.5<<" denari!"<<endl;
      system("PAUSE");
      return 0;
}
```

4. Најди ги грешките во следнава програма:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
```

```
{
        Cout<<"Kako si?"<<endl;
        cout<<"Mislis deka programiranjeto e tesko?">>endl;
        cout<<"Ne grizi se!;
        cout"Nabrgu ke razberes mnogu poveke";
        system(PAUSE);
        return(0)
}
```

4.3 Програма со редоследна структура

4.3.1 Основни елементи на програмскиот јазик

Градбени делови на програмскиот јазик C++

Програмските јазици се состојат од точно одредена група зборови кои програмерот според одредени правила ги подредува во искази кои му пренесуваат информација на компјутерот.

Програмските јазици, како и природните јазици, имаат своја *азбука*. Азбуката на јазикот C++ е составена од:

- големите и малите букви на англиската абецеда:

  |A | B| C | D| E| F|G|H|I|J|K|L|M|N|0| P|Q| R|S|T|U|V|W|X|Y|Z|

  |a | b | c | d | e| f|g|h| i| j| k|l|m|n|o|p|q|r|s| t|u| v|w|x|y|z|
- арапските цифри:

  |0 |1 |2|3|4|5|6|7|8|9|
- специјалните знаци:

  | + | - | * | / |= | (|) | { \ } \ [|] \ < | > | ‘ | “ | ! | # | \ | % | & | | | | _ |

  | ^ | ¸ | ~ \ , \ ; | : | ?|
- и знакот за празно место | |.

Често се користат и сложени симболи, односно двознаци и тризнаци, на пр.:

|+= | -= | *= | /= | %= | << | >> | ++ | == | /* | */ | && |!= |<<= |>>= |и други.

Со овие знаци се пишуваат сите конструкции од кои програмерот според точно одредени правилата гради искази (наредби). Тие конструкции претставуваат *градбени делови на програмскиот јазик*. Тоа се:

- резервирани зборови
- идентификатори (имиња на величини, функции, операции и сл.)
- оператори (математички, логички, релациони, за доделување)
- интерпункциски знаци (точки, наводници, загради и сл.)
- коментари (произволен текст со кој се објаснува текот на програмата).

Резервирани зборови

Некои од зборовите кои се користат во програмите однапред се дефинирани и преведувачот точно знае што тие значат. Овие зборови се нарекуваат *клучни зборови* и нивното значење не може да се менува. Некои од клучните зборови во јазикот C++ се: delete, int, return, if, then, else, do, while, true, false.

Постојат и предефинирани зборови чие значење исто така е однапред дефинирано али може да се смени (иако е најдобро тоа да не се прави). И едните и другите се *резервирани зборови* и тие не можат да се користат за имиња кои се формираат за да се опишат величини (променливи и константи), функции и слично. Резервираните зборови се пишуваат со мали букви.

Идентификатори

Имињата што ги формира програмерот се нарекуваат кориснички дефинирани зборови или *идентификатори*.

Постојат некои правила при формирање на идентификаторите:

- име се состои од еден или од повеќе знаци кои можат да бидат букви од англиската абецеда (a - z, A - Z), цифри (0 - 9) и знакот долна црта (_) (името не смее да содржи други специјални знаци ниту знакот за празно место),
- првиот знак во името може да биде буква или долната црта,
- се прави разлика помеѓу мали и големи букви (на пр. Zbir, zbir и ZBIR се три различни идентификатори).
- резервираните зборови не смеат да се користат како идентификатори.

Пр. 4. 11. Правилно запишани имиња во C++ се:

```
Broj,
BrojVoDnevnik,
reden_broj,
plostina1,
Ime1_1,
Sobiraj,
```

Пр. 4. 12. Неправилно запишани имиња во C++ се:

`ime i prezime`	содржи празно место
`100procenti`	започнува со цифрата 1
`sto%`	содржи специјален знак %
`ученик`	содржи букви од кирилицата
`return`	резервиран збор

Совет:

Иако не е пропишано како правило, добра пракса е да се користат имиња кои полесно ќе ги запомниш и ќе имаат некое значење. На пример, подобро е да користиш имиња како што се paralelka, volumen_na_prizma, MestoNaRagjanje и слично наместо a1, a2, a3, xy итн. Ова е особено значајно за снаоѓање во програмите. Ако некој друг, или самиот автор по подолго време, ја чита програмата, полесно ќе разбере за што во неа се работи.

Оператори

Операторите (+, -, *, /, =, %, ++, --, &&, ||, +=,*= и др.) се користат за означување на аритметички, на логички и на други операции кои се изведуваат врз величини (променливи и константи).

Интерпункциски знаци

Интерпункциските знаци (во кои спаѓа и знакот празно место) се користат за раздвојување на елементите на јазикот. На пример, со знакот точка и запирка се разделуваат искази.

Коментари

Програмерите пишуваат коментари за да објаснат што се работи во програмата. На тој начин други програмери полесно се снаоѓаат кога ја читаат програмата, а неретко коментарите им се од корист и на нив самите кога програмата ќе ја читаат по подолго време.

До сега се сретнавме со коментарот:

```
// mojata prva programa
```

Кога преведувачот ќе дојде до двојната коса црта, тој ќе го занемари текстот што следува сè до крајот на таа линија и преведувањето ќе го продолжи во следната линија. По знакот // може да се напише коментар само во една линија. Подолгите коментари (на пр. во заглавието на програмата може да се напише упатство за користење, лиценца и слично) во C++ се пишуваат помеѓу знаците /* и */. На пример:

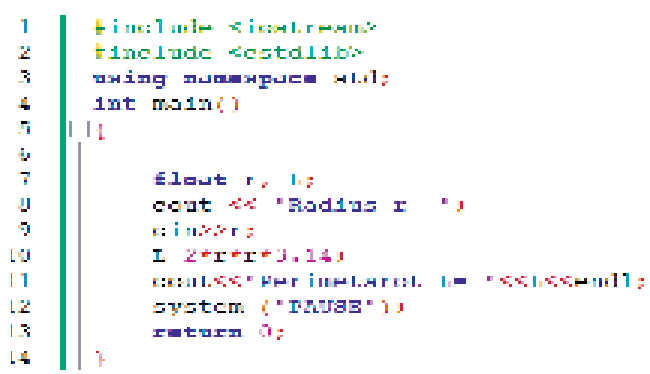
```
/*   So programava se presmetuva
     perimetarot i plostinata
     na pravoagolen triagolnik */
```

Основни елементи на програмскиот јазик C++

Во табелава се дадени основните елементи на програмскиот јазик C++:

Типови податоци	`int, float, char`
Константи	`0, 123.6, "Hello"`
Променливи	`i, sum`
Декларации	`int x; float a; int fun(int);`
Изрази	`sum + i`
Искази (наредби)	`sum = sum + i; while, for, if-else, switch, break`
Функции	`main(), Hello()`
Модули	`datoteka`

На следната слика е даден пример на програма со која се пресметува плоштина на круг. Веќе кажавме дека програмата Code Blocks различни елементи на програмата ги прикажува со различни бои.



Сл. 4. 26 Изглед на C++ програма

Претпроцесорските наредби се прикажани со зелената боја.

Операторите (<<, >>, *) се прикажани со црвената боја, идентификаторите (main, P, r) и резервираните зборови (cout, cin) се прикажани со црната боја, константите (3.14, 0) се прикажани со виолетовата боја.

Декларациите, за кои подоцна ќе зборуваме, (int, float) се прикажана со темно сината боја.

Функции

Во повеќето програми постојат делови од кои сочинуваат една целина и кои повеќепати се повторуваат. Пишување на тие делови повеќе пати е заморно, па затоа тие се издвојуваат, им се дава име и по потреба се повикуваат во програмата. Деловите од програми кои сочинуваат логичка целина и извршуваат точно одредени задачи се нарекуваат *потпрограми*. Во C++ сите потпрограми се претставени како функции.

Во програмите можат да се користат веќе постоечки функции кои се зачувани во библиотеките или можат да се креираат сопствени функции кои подоцна ќе се користат. Ова се посложени задачи со кои нема да се запознаеш на ова ниво на програмирање. Ќе ја користиш само функцијата main()за која веќе зборувавме.

Функцијата main ()

Тоа е главна функција на секоја C++ програма и означува дел на програма кој треба да се изврши. Таа содржи секвенца од искази (еден или повеќе искази) кои се пишуваат помеѓу големите загради { и }. Секвенците од искази се пишуваат вовлечено заради прегледност.

Структура на програма во C++

Се запознавме со основните елементи кои сочинуваат структура на програма во програмскиот јазик C++. Структурата на една програма изгледа вака:

```
/*
 коментари во заглавието
*/
претпроцесорски наредби
... декларации на променливи
int main()
{
 .... искази (извршни наредби)....
 return 0;
}
```

Резиме

Програмските јазици се состојат од точно одредена група зборови кои програмерот според одредени правила ги подредува во искази.

Градбените делови на програмскиот јазик се: *резервирани зборови, идентификатори , оператори, интерпункциски знаци* и *коментари*.

Резервираните зборови не можеме да ги користиме за имиња кои ние ќе ги формираме за да опишеме величини (променливи и константи), функции и слично. Имињата што ги формира програмерот се нарекуваат кориснички дефинирани зборови или *идентификатори*. Постојат правила при формирање на идентификаторите.

Основните елементи на програмскиот јазик C++ се: типови и класи, константи, променливи, декларации, изрази, искази, функции и модули.

Прашања:

1. Од кои знаци е составена азбуката на програмскиот јазик C++?
2. Наведи ги градбените делови на програмскиот јазик C++!
3. Што се резервирани зборови?
4. Зошто резервираните зборови не можат да се користат за формирање на корисничките имиња?
5. Што се идентификатори?
6. Кои правила мора да се почитуваат при формирање на идентификаторите?
7. Кои од наведените имиња не се идентификатори и зошто?

`tocno?`	`10SoKromid`	`Vozrast`	`noviot_avtomobil`
`a+b`	`AAa`	`x[1]`	`programski jazik`

8. Што се коментари?
9. Дали преведувачот ги преведува коментарите? Зошто?
10. Како се пишуваат коментари во C++?
11. Кои се основни елементи на програмскиот јазик C++?
12. Која функција мора да ја има секоја програма во програмскиот јазик C++?
13. Која е улогата на наредбата `include`?
14. Колку `main` функции може да има во програма?

4.3.2 Искази. Исказ за приказ на екран

Во програмата prva.cpp го видовме исказот

```
cout << "Ti posakuvame uspeh vo ucenjeto!";
```

со кој текстот „Ti posakuvame uspeh vo ucenjeto!“ се прикажува на екранот.

Да ги анализираме составните делови на овој исказ:

- `cout` – израз која насочува податоци кон излезот, најчесто тоа е монитор,
- `<<` – операторот на испишување кој пораката ”Ti posakuvame uspeh vo ucenjeto!” ја праќа кон излезот,
- `" "` – сè што е ставено во наводници е порака која се прикажува на екранот,
- `;` – означува крај на исказ (не на линија).

Со користење на исказот за приказ, на мониторот може да се прикажат (отпечатат) какви било податоци. Мониторот е стандардна излезна единица.

Примери за примена на исказот за приказ:

`cout<<25;`	ќе се прикаже бројот 25
`cout<<"a";`	ќе се прикаже вредност буквата a
`cout<<a;`	ќе се прикаже вредност на променливата a
`cout<<2*a;`	ќе се прикаже двојната вредност на променливата a
`cout<<"Tekst";`	ќе се прикаже текстот „Tekst“

Пр. 4. 13. Изврши ја следната програма!

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"Zdravo!";
    cout<<"Jas sum C++ programa.";
    system("PAUSE");
    return 0;
}
```

```
Zdravo!Jas sum C++ programa.Press any key to continue . . .
```

Забележуваш дека речениците „Zdravo!“ и „Jas sum C++ programa!“ се прикажани во една линија и се прилепени една со друга. Како ќе ги разделиш со едно празно место?

По испишување на податоци наведени во исказот `cout`, покажувачот останува во истата линија и следното испишување на податоци продолжува тука.

Пр. 4. 14. За испишување во нова линија, се користи изразот `endl` (кратенка од end line) со кој покажувачот се поместува за една линија подолу.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"Zdravo!"<<endl<<"Jas sum C++ programa.";
    system("PAUSE");
    return 0;
}
```

```
Zdravo!
Jas sum C++ programa.Press any key to continue . . .
```

Во исказот `cout<<”Zdravo!”<<endl<<”Jas sum C++ programa.”;` се наведени повеќе оператори на испишување. Во тој случај податоците се испишуваат од лево кон десно.

Во општ случај исказот cout се користи на следниот начин:

```
cout<<podatoci[<<podatoci];
```

при што податоците можат да бидат константи, променливи, изрази или нивна комбинација.

Забелешка:

Деловите на искази кои не се задолжителни ќе ги ставаме во средни загради.

Техника на редоследно извршување

Во главната функција има два искази за приказ на екран:

```
cout<<"Zdravo!";
cout<<"Jas sum C++ programa.";
```

Кој исказ ќе се изврши прв? Што ќе се случи ако на исказите им се заменат местата?

Пр. 4. 15. Програмата од примерот 5.14 може да се напише и вака:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"Zdravo!"<<endl;
    cout<<"Jas sum C++ ";
    cout<<"programa.";
    system("PAUSE");
    return 0;
}
```

```
Zdravo!
Jas sum C++ programa.Press any key to continue . . .
```

Во овој пример има повеќе искази за печатење на екран наредени еден по друг. Сите овие искази ќе се извршат точно по оној редослед по кој се напишани.

Вака напишани искази се нарекуваат *секвенца од искази*. Пишувањето на искази во програмскиот код кои се извршуваат сите и редоследно се вика *техника на редоследно извршување*.

Пр. 4. 16. Отвори нова датотека и дај ѝ име C++. Препиши го кодот:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"   **** "<<endl;
    cout<<"  ******        **        **"<<endl;
    cout<<" ***   **       **        **"<<endl;
    cout<<" **              **        **"<<endl;
    cout<<" **          ********  ********"<<endl;
    cout<<" **          ********  ********"<<endl;
    cout<<" **              **        **"<<endl;
    cout<<" ***   **        **        **"<<endl;
    cout<<"  ******         **        **"<<endl;
    cout<<"   ****"<<endl;
    system("PAUSE");
    return 0;
}
```

Преведи ја и изврши ја програмата!

Зад. 4. 7. Напиши и изврши програма со која на екранот ќе се прикаже твоето име напишано со помош на ѕвездички!

Резиме

Исказ или *наредба* одредува кои дејства е потребно да се извршат за да се добие резултат. Секој исказ мора да заврши со знакот точка и запирка.

За приказ на екран се користи исказот cout. Во општ случај исказот *cout* се користи на следниот начин: *cout<<podatoci[<<podatoci];* при што податоците можат да бидат константи, променливи, изрази или нивна комбинација.

cout насочува податоци кон излез, најчесто тоа е монитор, „<<“ е оператор на испишување кој пораките ставени во наводници ги праќа кон излезот.

Искази кои се извршуваат точно по оној редослед по кој се напишани претставуваат *секвенца од искази*. Пишувањето искази во програмскиот код кои се извршуваат сите и редоследно се вика *техника на редоследно извршување*.

Вештини што треба да ги усовршиш:

Да познаваш структура на исказот за прикажување на екран (cout).

Правилно да го употребуваш исказот cout и операторот <<.

Да го користиш изразот endl во исказот за прикажување на екран.

Да напишеш едноставна програма со користење на исказите cout и техника со напластени искази.

Прашања:

1. Што е исказ?
2. Кој исказ се користи за приказ на екран?
3. Во исказот за приказ на екранот, што означува операторот << ?
4. Во исказот за приказ на екранот, кој израз се користи за да покажувачот помине во нов ред?
5. Што се означува со изразот напластени искази?
6. Како се нарекува техниката на пишување искази кои се извршуваат сите и редоследно?
7. а) Што ќе се прикаже на екранот со следниов исказ?

```
cout<<”J”<<endl<<”A”<<endl<<”N”<<endl<<”A”<<endl;
```

б) Што ќе се прикаже на екранот со следнава секвенца од искази?

```
cout<<”J”<<endl;
cout<<”A”<<endl;
cout<<”N”<<endl;
cout<<”A”<<endl;
```

Задачи:

1. Напиши програма со која ќе го добиеш следниов приказ на екранот:

   Zdravo!

   Kakov prekrasen den!

   Odam na prosetka.

2. Напиши програма со која ќе се отпечати твоето име на следниов начин:

```
I
M
E
```

3. Напиши програма со која ќе се отпечати твоето име на следниов начин:

```
I
 M
  E
```

4. Напиши програма со која на мониторот ќе се прикаже слика по твој избор со помош на ѕвездички!

4.4 Променливи и искази за доделување

4.4.1 Аритметички операции и изрази

Пр. 4. 17. Напиши ја и изврши ја следнава програма:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      cout<<"5+2"<<endl;
7      system("PAUSE");
8      return 0;
9  }
```

```
5+2
Press any key to continue . . .
```

На екранот се прикажува „5+2“. Овде изразот „5+2” е низа од знаци која се прикажува на истиот начин како што таа е напишана во наводниците.

Напиши ја истата програма, но сега тргни ги наводниците!

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      cout<<5+2<<endl;
7      system("PAUSE");
8      return 0;
9  }
```

```
7
Press any key to continue . . .
```

Програмата ги собира броевите 5 и 2 и резултатот, кој во случајов е 7, го прикажува на екранот. Изразот 5+2 претставува аритметички израз кој има своја бројна вредност.

Зад. 4. 8. Напиши програма со која на екранот ќе се прикаже 5 + 2 = 7!

Аритметички израз е бројна вредност или запис од две или на повеќе бројни вредности и математички оператор(и) меѓу нив. *Оператори* во програмскиот јазик C++ се: плус (+) за операцијата собирање, минус (-) за операцијата одземање, ѕвездичка (*)

за операцијата множење, коса црта (/) за операцијата делење и модул (%) за остаток при делење на два цели броја. Ова се само некои од операторите кои се користат во програмскиот јазик C++.

Аритметичките изрази се градат на ист начин како и во математиката. Примери за аритметичките изрази се: `2*3+8`, `7-2`, `(15-5)*100`, `2/a`, `a-b`. Овде броевите како што се 2, 3, 100 итн. се константи, додека `a` и `b` се променливи.

Променливи и константи

Со програмите се обработуваат податоци и тие претставуваат *величини* кои можат да бидат:

- константи – величини кои не ја менуваат својата вредност,
- променливи – величини кои ја менуваат својата вредност.

Доделување на вредност на променлива. Оператор за доделување

Во компјутерот за секоја променлива се предвидува и се резервира место во меморијата. Секое резервирано место има своја адреса која е тешка за памтење, па од тие причини на променливите им се доделуваат *симболички имиња*. При тоа, мора да се води сметка да се почитуваат правилата кои важат за сите идентификатори.

Во математиката и во другите науки се среќаваш со многу променливи, на пр. a, b, c – страни на триаголник, P – плоштина, t – температура итн. На сите овие променливи им се доделуваат различни вредности. На пример, на температурата може да ѝ се додели вредност -15, али и 23, 17, 32 итн.

На променливите им се доделува вредност со помош на *оператор за доделување*. Операторот за доделување е знакот еднакво (=).

Важно!

Знакот еднакво (=) овде не означува еднаквост како во математиката. Со оператор за доделување на ***променливата од левата страна*** *на операторот ѝ се* ***доделува вредност на изразот од десната страна*** *на операторот. Од тие причини променливата на која ѝ се доделува вредност мора да биде од левата страна на операторот за доделување. Неточно е на пр. 5 = a или a+b = c;*

Примери за доделување вредности на променливи се дадени во следнава табела:

`x = 5;`	на променливата `x` ѝ се доделува вредност 5. Ова всушност, значи дека во мемориската локација која е означена со симболичко име `x` ќе се зачува бројот 5 или дека бројот 5 е содржина на променливата `x`.
`a = 20;`	на променливата `a` ѝ се доделува вредност 20.
`a = b*3;`	помножи ги содржината на променливата `b` и бројот 3 и производот зачувај го во променливата `a`.
`x = x+3;`	на содржината на променливата x додај број 3 и збирот зачувај го во променливата `x`.
`x=c=b=5;`	на променливата `b` ѝ се доделува вредност 5, на променливата `c` ѝ се доделува содржината на променливата `b`, на променливата `x` ѝ се доделува содржината на променливата `c`. Конечен резултат на овој исказ е: содржината на сите три променливи е иста и е еднаква на бројот 5.

Да ја разгледаме следнава секвенца од искази:

Исказ	Содржина на променливата		
	a	b	zbir
a = 3;	3	?	?
b = 7;	3	7	?
a = a+1;	4	7	?
zbir = a+b;	4	7	11

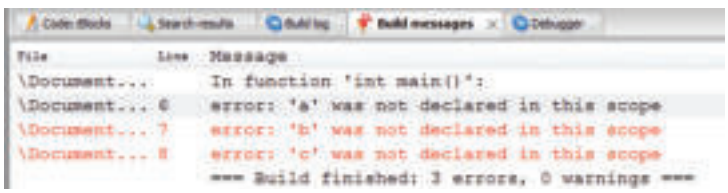
По извршување на оваа секвенца од искази променливата a ќе има вредност 4, променливата b ќе има вредност 7 и променливата zbir ќе има вредност 11.

Посебно внимание ќе му обрнеме на исказот a=a+1; Во математиката ваков исказ не е точен, но во програмскиот јазик C++ знакот = нема исто значење како во математиката. Пред овој исказ, на променливата a ѝ е доделена вредност 3 (со исказот a=3;). Со исказот a=a+1; вредноста на променливата a (бројот 3) се собира со 1 и збирот (бројот 4) се доделува на променливата a. Тоа значи дека во мемориската локација која е означена како променлива a претходната содржина (бројот 3) ќе се избрише, а на нејзиното место ќе се зачува новата содржина (бројот 4).

Сè уште не можеме да напишеме програма во која ќе се користат променливи. Обиди се да ја извршиш следнава програма:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      a = 2;
7      b = 5;
8      c = a+b;
9      cout<<c<<endl;
10     system("PAUSE");
11     return 0;
12 }
```

Преведувачот ќе јави грешки дека променливите кои се користат во програмата не се декларирани:



Очигледно е дека преведувачот бара да се декларираат (најават) променливи кои ќе се користат во програмата. За ова да може да се направи, прво ќе се запознаеме со поимот тип на променлива.

4.4.2 Типови на променливи

Компјутерот мора однапред да знае какви податоци ќе се чуваат во променливите. Причината е едноставна – потребно е повеќе место за зачувување, на пр., на една реченица отколку за зачувување на еден знак. Значи, компјутерот треба однапред да знае колкав простор во меморијата ќе резервира за секоја променлива.

Освен тоа, компјутерот не се однесува исто со сите податоци. На пример, со броевите може да извршува аритметичките операции, додека со знаци тоа не може да го прави. Од овие причини за секоја променлива мора да се знае *тип на податокот* кој во неа ќе биде зачуван. Се разликуваат основни и други типови податоци. Основните типови податоци се:

- *цели броеви* – броеви кои немаат децимален дел,
- *реални броеви* – броеви кои имаат цел и децимален дел,
- *знаци* – кој било знак од азбуката на C++,
- *логички податоци* – точно или неточно, односно 0 или 1.

Типот на променлива е одреден со типот на податокот кој во неа може да се зачува. На секој променлива, освен симболичко име, мора да ѝ се додели и *ознака за тип* кој ни кажува колку простор во меморијата треба да се резервира, колкав е опсегот на дозволените вредности на податокот, кои операции се можни со тој податок и слично.

Во табелата се дадени дел од основните типови податоци и нивните ознаки, како и вредностите кои можат да се доделат на променлива од соодветен тип:

Ознака	Опис	Големина	Опсег на вредности
`char`	знаковна променлива	1 бајт	256 знаци
`int`	целобројна променлива	4 бајти	0 - 4294967295 или од -2147483648 до 2147483647
`bool`	логичка променлива	1 бајт	true или false
`float`	реална променлива со обична прецизност	4 бајти	со точност до 7 децимални места
`double`	реална променлива со двојна прецизност	8 бајти	со точност до 15 децимални места

Целобројните типови можат да бидат само позитивни или и позитивни и негативни. Ако во некоја целобројна променлива треба да се зачуваат само позитивни вредности, тогаш пред ознаката `int` се става ознаката `unsigned`. Постои и ознака `signed`, но таа се подразбира, па не мора да се пишува.

Ќе спомнеме уште еден тип податоци кој не се вбројува во основните типови, но во многу работи така се однесува така. Тоа е низа од знаци (карактери) или *стринг* со која веќе се запозна (“Dobar den”, “C++” итн.). За сега е доволно да знаеш дека низа од знаци се пишува помеѓу наводници. Ознака за овој тип податоци е string.

За љубопитните:

Комплетната табела со сите основни типови податоци можеш да ја погледнеш во додатокот А, на крајот на учебникот.

Декларирање на променливи

Секоја променлива која ќе се користи во програма мора да се најави, односно да се *декларира*. При тоа треба да се има предвид следниве карактеристики:

- секоја променлива има свое име
- секоја променлива има тип
- секоја променлива содржи вредност која ѝ се доделува најчесто со операторот за доделување.

Кога се декларира некоја променлива, прво се наведува ознака за тип, а потоа име на променливата:

```
tip imenapromenliva [imenapromenliva];
```

По еден тип може да се наведат и повеќе променливи одвоени со запирка. Така со исказот `int i, j;` се најавува дека ќе се користат две целобројни променливи, нивните имиња се `i` и `j`, на нив може да им се додели вредност која ќе биде цел број.

Примери на декларирање на променливи:

`int a;`	декларација на целобројна променлива a
`float broj;`	декларација на реална променлива broj
`char bukva;`	декларација на знаковна променлива bukva
`int b,c,d;`	декларација на 3 целобројни променливи b, c и d

Пр. 4. 18. Сега имаме сè што е потребно за да се напише програма со која ќе се соберат вредности на две променливи. Пред да се користат променливи, тие треба да се декларираат:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b, c; // deklariranje na promenlivi
    a = 2;
    b = 5;
    c = a+b;
    cout<<c<<endl;
    system("PAUSE");
    return 0;
}
```

```
7
Press any key to continue . . .
```

Наместо исказот `cout<<c<<endl;`

подобро е да се напише `cout<<"c="<<c<<endl;`

со што на екранот ќе се отпечати „c=7“. На тој начин корисникот знае на што се однесува резултатот што е прикажан на екранот.

Истиот излез ќе се прикаже и ако се напише `cout<<"c="<<a+b<<endl;` Во овој случај може да се изостави исказот `c=a+b;`. Зошто?

Пр. 4. 19. Со следната програма ќе се отпечати „c“ наместо 7. Зошто?

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b, c;
    a = 2;
    b = 5;
    c = a+b;
    cout<<"c"<<endl;
    system("PAUSE");
    return 0;
}
```

```
c
Press any key to continue . . .
```

Треба да се внимава кога се доделуваат вредности на променливите. Преведувачот нема да не извести ако на некоја променлива ѝ се доделува податок од несоодветен тип, програмата ќе се преведе но нема да даде точен резултат. Да го погледнеме следниот пример.

Пр. 4. 20. На целобројната променлива a ѝ се доделува реален број:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b, c;
    a = 2.5;
    b = 5;
    c = a+b;
    cout<<"c = "<<c<<endl;
    system("PAUSE");
    return 0;
}
```

```
c = 7
Press any key to continue . . .
```

Во овој пример со изразот `a=2.5;` на променливата a нема да ѝ се додели реален број 2.5. Таа е декларирана како целобројна променлива па во неа се зачувува само целиот дел од доделената вредност (2).

Зад. 4. 9. Во претходниот пример на променливата a додели ѝ податок од тип знак, на пример `a='2';` (знаците се пишуваат помеѓу апострофи). Преведи ја и изврши ја програмата! Што се случува?

Иницијализација на променливи

Пред да се започне со користење со користење на некоја променлива, неопходно е таа да има некоја вредност. Доделување на почетната вредност на променливите се нарекува *иницијализација на променливи.*

На променливите може да им се доделат вредност во истиот исказ со кој тие се декларираат. Така наместо:

```
int a;
a = 5;
```

може да се напише

```
int a = 5;
```

Пр. 4. 21. Плоштина на правоаголникот со страните a = 8 и b = 3:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    // inicijalizacija na promenliv
    float a = 8;
    float b = 3;
    float P;
    // dodeluvanje na vrednost na P
    P = a*b;
    cout<<"P = "<<P<<endl;
    system("PAUSE");
    return 0;
}
```

```
P = 24
Press any key to continue
```

Пр. 4. 22. Замена на вредности на две променливи:

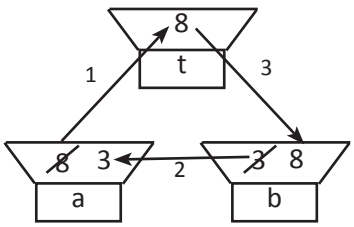
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    // inicijalizacija na promenlivi
    int a = 8;
    int b = 3;
    int t;
    // dodeluvanje na vrednosti
    t = a;
    a = b;
    b = t;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl;
    system("PAUSE");
    return 0;
}
```

```
a = 3
b = 8
Press any key to continue
```

Замена на вредности на две променливи често се користи во програмите.

Променливата `t` се користи како привремена променлива во која ќе се зачува вредноста на променливата `a` пред да ѝ се додели нова вредност со исказот `a=b;`. Со исказот `b=t;` на променливата `b` ѝ се доделува вредноста од привремената променлива `t` што всушност е претходната вредност на променливата `a`.

	a	b	t
`int a = 8;`	8	?	?
`int b = 3;`	8	3	?
`t = a;`	8	3	8
`a = b;`	3	3	8
`b = t;`	3	8	8



Сл. 4. 27 Графички приказ на замена на вредности на променливи

Важно!

Ако на променливите не им се додели некоја вредност тие сепак ќе имаат некоја случајна вредност која преведувачот автоматски ќе и додели. Линиите a = 5; и b = 2; стави ги како коментари, преведи ја и изврши ја програмата. Ќе видиш дека програмата ќе работи поинаку, односно ќе даде некој произволен резултат. ***Секогаш доделувај вредности на променливи!***

Константи

Освен променливите, во C++ се користат и константи. Со константите во програмите се изразуваат одредени вредности. Како и променливите, и константите се разликуваат според типот на податок со кој нејзината вредност е изразена. Така има:

- *целобројни константи* (15, 100, -50)
- *реални константи* (2.34, 105.5, -12.2)
- *знаковни константи* (‘a’, ‘M’, ‘+’, ‘$’)
- *логички константи* (true/false, 1/0)
- *стринговни константи* (“Dobar den”, “C++”).

Забележуваш дека знаковните константи се запишуваат помеѓу апострофи, додека стринговните константи се запишуваат помеѓу наводници.

И константите можат да се декларираат на сличен начин како и променливите. Единствената разлика е во тоа што се користи клучниот збор const. На пр.:

```
const float DDV = 0.18;
const float BRZINA_NA_SVETLINATA = 2.997925e8;
const char ZNAK='$';
```

Реалната константа DDV е прикажана со децимална точка, а константата `BRZINA_NA_SVETLINATA`, која исто така е реална, е прикажана со т. н. научно означување. Во овој начин на означување се користи малата или големата буква „e“ со која се означува експонентот на бројот. `2.997925e8` е претставен бројот $2.997925 \cdot 10^8$.

Забелешка:

Програмерите имињата на константите вообичаено ги пишуваат со големи букви иако такво синтаксно правило во јазикот C++ не постои.

Пр. 4. 23. Следнава програма пресметува плоштината на круг со радиусот r = 2.5:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      const float PI = 3.14;
7      float r = 2.5;
8      float P;
9      P = PI*r*r;
10     cout<<"P = "<<P<<endl;
11     system ("PAUSE");
12     return 0;
13 }
```

```
P = 19.625
Press any key to continue . . .
```

Во програмава се користи реалната константата `PI` на која ѝ се доделува вредност 3.14.

Резиме

Аритметички израз е бројна вредност или запис од две или повеќе бројни вредности и математички оператор(и) меѓу нив. Операторите во програмскиот јазик C++ се: +, -, *, / и %. Секоја променлива или константа која ќе се користи во програма мора да се најави, односно *декларира*. Под ова се подразбира одредување на името и на типот на променливата.

На променливите им се доделува вредност со помош на операторот за доделување (=). Ако на променливите не им доделиме вредност, тие сепак ќе имаат некоја случајна вредност која преведувачот автоматски ќе ѝ ја додели.

Основни типови и ознаки на променливите и на константите се: целобројни, реални, знаковни и логички.

Доделување почетна вредност на променлива се нарекува *иницијализација на променлива*.

Вештини што треба да ги усовршиш:

Да пишуваш и да користиш аритметички изрази во C++.

Да доделуваш вредности на променливи со помош на операторот =.

Да препознаваш типови на променливи и правилно да ги декларираш.
Да напишеш едноставна програма со доделување вредност на променлива преку едноставни математички изрази.

Прашања:

1. Што е аритметички израз? Напиши неколку аритметички изрази?
2. Кои величини се користат во програмските јазици?
3. Направи разлика помеѓу константи и променливи?
4. Што се подразбира под декларација на променлива?
5. Дали е можно повеќе променливи да се декларираат со еден исказ? Ако е, наведи пример!
6. Која од наведените декларации е исправна?
   а) `int n=-10;`
   б) `unsigned int i =-10;`
   в) `signed int = 3.8;`
7. Наведи примери за секој од типовите на константи!
8. Што ќе се прикаже на екранот со следниот програмски код?
```
char a,b,c;
a='b';
b='c';
c=a;
cout<<a<<b<<c;
```
9. Што ќе се прикаже на екранот со следниот програмски код?
```
bool x=2>1;
cout<<x;
```

Задачи:

1. Ако во програма се користат променливите int kolicina=24 и float cena=12.5, со наредбата cout испиши ги:
   а. нивните имиња
   б. нивните вредности
   в. извештај во форма „kolicina = x, cena = y“, така што наместо x и y се испишуваат вредностите на променливите kolicina и cena.
2. Напиши програма за пресметување на плоштина на правоаголник со страните a=8.5 и b = 3.2! Излезот да се прикаже на екранот во следнава форма:
   `Plostinata na pravoagolnikot so stranite a=8.5 i b=3.2 iznesuva 23.4.`
3. Напиши програма за пресметување на плоштина на круг со радиус r = 2.5! Излезот да се прикаже на екранот во следнава форма:
   `Plostinata na krugot so radiusot r = 2.5 iznesuva 19.625.`
4. Напиши програма со која ќе пресметаш плоштина и периметар на квадрат со страна a = 8.5!
5. Напиши програма со која ќе се најде аголот на врвот на рамностран триаголник чиј агол на основата е 30 степени!
6. Напиши програма со која ќе најдеш решение на линеарната равенка чии параметри се a=3, b=-18!

4.5 Дополнителни специфики на јазикот

4.5.1 Исказ за внесување на податоци во програма

Програмите кои прикажуваат излез како информација која е непроменлива и е независна од корисникот се неинтерактивни компјутерски програми. Во таквите програми на променливите им се доделува некоја вредност во самата програма. За на променливите да им се доделат поинакви вредности, мора да се измени програмата, повторно да се преведе и повторно да се изврши. Ова не е она што се очекува од програмите. Програмите треба да бидат интерактивни каде корисникот ќе може да внесе вредности на променливи според неговите потреби и да се прикажат резултати зависно од влезните податоци на корисникот. Секако дека во програмските јазици тоа е овозможено.

Во C++ преземањето на податоци се извршува со операторот >> во комбинација со исказот `cin`. На пример, со исказот

```
cin>>broj;
```

се прифаќа вредност која корисникот ја внесува преку тастатурата и ѝ се доделува на променливата `broj`.

Да ги анализираме составните делови на овој исказ:

- `cin` – израз кој „извлекува“ податоци од влезот, стандарден влез е тастатурата,
- `>>` – оператор за читање кој презема податок од влезот и го праќа во соодветна променлива,
- `broj` – име на променлива на која ѝ се доделува вредност прифатена од тастатурата,
- `;` – означува крај на исказ (не на линија).

Примена на исказот за внесување на податоци:

`cin>>x;`	Програмата очекува внесување на податок кој ќе се зачува во променливата x.
`cin>>a>>b;`	Можно е со еден исказ за внесување да се внесат повеќе податоци кои во променливите се зачувуваат по редоследот од лево кон десно.

Корисникот податоците ги внесува преку тастатурата. Исказот `cin` ја чита внесената вредност и ја доделува на променливата чие име е наведено како параметар во исказот.

По внесувањето на податоците, корисникот мора да го притисне копчето Enter, инаку програмата нема да ги преземе податоците и нема да продолжи со работа.

`cin>>a>>b;` е исто што и

```
cin>>a;
cin>>b;
```

Во двата случаи корисникот треба да внесе два податока, првиот за променливата `a` и вториот за променливата `b`. При тоа, по внесување на секој податок може да го притисне копчето Enter, но податоците може да ги одвои и со притискање на копчињата Space или Tab.

Во општ случај исказот `cin` го користиме на следниот начин:

```
cin>>promenliva[>>promenliva];
```

Пр. 4. 24. Програмата за собирање на два броја е преуредена така што корисникот ќе внесува податоци за променливите a и b.

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int a, b, c; //deklariranje na promenlivi
7      cin>>a>>b;   //vnesuvanje na promenlivi
8      c = a+b;     //dodeluvanje na vrednost
9      cout<<"c = "<<c<<endl;
10     system ("PAUSE");
11     return 0;
12 }
13
```

внесува корисник
внесува корисник

```
4
5
c = 11
Press any key to continue
```

4.5.2 Техника за објаснувања за податоците кои се очекуваат од корисникот

Програмата од претходниот пример нема грешки, сепак нешто недостасува. Кога ќе ја покренеш програмата добиваш празен екран, програмата чека да внесеш некој податок. Програмата сам си ја напишал и знаеш што треба да направиш, но што ако некој друг ја користи програмата?

Проблемот може да се случи и ако корисникот внесе погрешен податок. Изврши ја програмата и за променливата a внеси некој реален број. Програмата ќе даде неточен резултат. Од овие причини на корисникот треба да му се даде појаснување во вид на порака што од него се очекува. Пред секој `cin` исказ треба да се напише и `cout` исказ со кој на корисникот ќе му се прикаже порака каков податок од него се очекува.

Значи наместо

```
cin>>a;
```

се пишува

```
cout<<"Vnesi cel broj: "<<endl;
cin>>a;
```

Пр. 4. 25. Програмата во која се користат пораки за објаснување какви податоци се очекуваат од корисникот:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      // deklariranje na promenlivi
7      int a, b, c;
8      //vnesuvanje na promenlivi
9      cout<<"Vnesi cel broj: ";
10     cin>>a;
11     cout<<"Vnesi uste eden cel broj: ";
12     cin>>b;
13     c = a+b; // dodeluvanje na vrednost
14     cout<<"Zbirot na broevite ";
15     cout<<a<<" i "<<b<<" iznesuva "<<c<<endl;
16     system ("PAUSE");
17     return 0;
18 }
```

```
Vnesi cel broj: 6
Vnesi uste eden cel broj: 4
Zbirot na broevite 6 i 4 iznesuva 10
Press any key to continue . . .
```

Пр. 4. 26. Програма за пресметување на претходник и следбеник на цел број.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a;
    cout<<"Vnesi cel broj: ";
    cin>>a;
    cout<<"Prethodnik na brojot "<<a<<" e "<<a-1<<endl;
    cout<<"Sledbenik na brojot "<<a<<" e "<<a+1<<endl;
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 8
Prethodnik na brojot 8 e 7
Sledbenik na brojot 8 e 9
Press any key to continue . . .
```

Резиме

Програмите каде корисникот може да внесе вредности на променливи и со кои се прикажуваат резултати зависно од влезните податоци на корисникот се нарекуваат *интерактивни програми.*

Во C++ превземањето на податоци се извршува со операторот „>>“ во комбинација со *cin*. Во општ случај *cin* се користи на следниот начин: *cin>>promenliva[>>promenliva];*

Пред секој cin исказ треба да се напише и cout исказ со кој на корисникот му се прикажува порака каков податок од него се очекува.

Вештини што треба да ги усовршиш:

Да го користиш исказот за внесување на податоци во програма (cin).

Правилно да го користиш операторот >> во исказот за внесување на податоци.

Да користиш техника со која на корисникот ќе му објасниш што од него се очекува.

Да напишеш едноставна програма со исказот за внесување на податоци во програма.

Прашања:

1. Кои програми се нарекуваат интерактивни програми?
2. Со кој исказ се овозможува внесување на вредности за променливи?
3. Како корисникот внесува вредности во програма?
4. Напиши исказ со кој преку тастатурата се внесува реален број и се придружува на променливата t!
5. Зошто е потребно на корисникот да му се прикаже порака пред тој да внесе некоја вредност?
6. Кои вредности ќе им се доделат на променливите a и b со исказот cin>>a>>b; ако корисникот по ред ги внесе следниве вредности: -1, 1?
7. Што ќе се прикаже на мониторот по извршување на следниот код:

```
int i,j,k;
cin>>j>>k>>i;
cout<<i<<", "<<j<<", "<<k;
```

ако корисникот по ред ги внесе следните вредности: 1, 2, 3?

Задачи:

1. Напиши програма со која ќе се пресметаат плоштината и периметарот на правоаголник! Овозможи корисникот да внесе вредности на страните на правоаголникот.

Пример 1: Vnesi gi stranite na pravoagolnikot: 5 3 Plostinata na pravoagolnikot e 15, a negoviot perimetar e 16	Пример 2: Vnesi gi stranite na pravoagolnikot: 4.5 2.8 Plostinata na pravoagolnikot e 12.6, a negoviot perimetar e 14.6

2. Напиши програма со која ќе се пресметаат плоштината и периметарот на квадрат! Овозможи корисникот да внесе вредност на страната на квадратот.

Пример 1: Vnesi ja stranata na kvadratot: 5 Plostinata na kvadratot e 25, a negoviot perimetar e 20	Пример 2: Vnesi ja stranata na kvadratot: 7.2 Plostinata na kvadratot e 51.84, a negoviot perimetar e 28.4

3. Напиши програма со која ќе се пресметаат плоштината и периметарот на круг со даден радиус!

Пример 1: Vnesi go radiusot r= 5 Plostinata na krugot e 78.5, a negoviot perimetar e 31.4	Пример 2: Vnesi go radiusot r= 3.2 Plostinata na krugot e 32.1536, a negoviot perimetar e 20.096

4. Напиши програма која ѝ помага на касиерката да пресмета кусур, така што од неа бара да внесе цена на артикал, количина на артикл и износ што го зема од купувачот!

Пример: Cena: 100 Kolicina: 4.5 Iznos: 1000 Kusur: 550

4.5.3 Дополнителни спецификации на јазикот C++ (прв дел)

Операцијата делење со целобројни и со реални променливи

Да ја разгледаме следнава програма:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout<<"Vnesi dva celi broja!"<<endl;
    cin >>a>>b;
    cout<<"Zbirot na broevite "<<a<<" i "<<b<<" iznesuva "<<a+b<<endl;
    cout<<"Razlikata na broevite "<<a<<" i "<<b<<" iznesuva "<<a-b<<endl;
    cout<<"Proizvodot na broevite "<<a<<" i "<<b<<" iznesuva "<<a*b<<endl;
    cout<<"Kolicnikot na broevite "<<a<<" i "<<b<<" iznesuva "<<a/b<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dva celi broja!
5       2
Zbirot na broevite 5 i 2 iznesuva 7
Razlikata na broevite 5 i 2 iznesuva 3
Proizvodot na broevite 5 i 2 iznesuva 10
Kolicnikot na broevite 5 i 2 iznesuva 2
Press any key to continue . . .
```

Гледаме дека операторите работат главно тоа што од нив се очекува, освен операторот за делење. Наместо очекуваниот резултат 2.5, програмата дава резултат 2. Ова се случува затоа што променливите `a` и `b` се декларирани како целобројни променливи па програмата како резултат враќа цел број (го занемарува децималниот дел на резултатот).

Истото ќе се случи и ако резултатот се зачува во нова променлива која ќе се декларира како реална променлива. Со секвенцата од искази:

```
int a = 5;
int b = 2;
float kolicnik = a/b;
cout<<kolicnik;
```

на екранот ќе се прикаже 2 иако променливата `kolicnik` е декларирана како реална променлива.

Зад. 4. 10. Во математиката изразот 1/3*3 има вредност 1. Напиши програма со која ќе провериш која вредност ја има овој израз во C++! Објасни!

Ваков начин на делење не е грешка на програмскиот јазик C++. Вака дефинираната операција на делење на целобројните константи и променливи се нарекуваат *целобројно делење,* а резултатот се нарекува *целобројен количник.* На пр.: `10/2=5, 10/3=3, 100/75=1, 75/100=0` итн.

За целобројните константи и променливи е дефинирана уште една операција – *остаток при целобројното делење*, за која се користи операторот „%“ (modul). Така `5%2 = 1`, бидејќи остатокот на делењето на броевите 5 и 2 е 1, `10%2=0, 10%3=1, 100%75=25, 75%100=75` итн.

Зад. 4. 11. Напиши програма со која се пресметува количникот и остатокот на делење на два цели броја!

Пр. 4. 27. Програма со која се наоѓаат цифри на даден двоцифрен број:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Vnesi dvocifren broj!"<<endl;
    cin>>n;
    cout<<"Prvata cifra na brojot ";
    cout <<n<<" e "<<n/10<<endl;
    cout<<"Vtorata cifra na brojot ";
    cout <<n<<" e "<<n%10<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dvocifren broj!
38
Prvata cifra na brojot 38 e 3
Vtorata cifra na brojot 38 e 8
Press any key to continue . . .
```

Зад. 4. 12. Напиши програма со која ќе го најдеш збирот на цифрите на даден двоцифрен број!

Претворање на типови податоци

Видовме дека ако и деленикот и делителот се цели броеви, тогаш и количникот ќе биде цел број без разлика дали броевите се деливи или не. На пример, со исказот

```
float kolicnik = 5/2;
```

променливата `kolicnik` ќе добие вредност 2 (цел број).

Секој цел број може да се запише како реален број, на пр. бројот 2 може да се запише како 2.0 или скратено 2. Ако претходниот исказ се запише како

```
float kolicnik = 5./2; или float kolicnik = 5/2.;
```

променливата `kolicnik` ќе добие вредност 2.5 (реален број)

Доволно е децималната точка да се стави само после еден операнд (величина врз која се извршуваат операции).

Во C++ е дозволено претворање на типови податоци, тоа се одвива според одредени правила:

- Ако операндите се од ист тип, од тој тип е и резултатот.
- Ако операндите се од различни типови, двата операнди и резултатот се претвораат во посложениот тип. Така, ако еден операнд е цел број, а друг реален, двата операнди ќе се претворат во реален број, а таков ќе биде и резултатот.

Примери на претварање разни типови податоци:

`int a,b;`	резултатот на изразот `a/b` ќе биде цел број
`int a;` `float b;`	резултатот на изразот `a/b` ќе биде реален број
`int a,b;` `int c;`	резултатот на изразот `a*b/c` ќе биде цел број
`int a,b;` `float c;`	резултатот на изразот `a*b/c` ќе биде реален број

Пр. 4. 28. Програма која пресметува дадени метри во километри.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int m;
    cout<<"Vnesi broj na metri!"<<endl;
    cin>>m;
    cout<<m<<" metri e "<<m/1000.<<" km "<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi broj na metri!
100
100 metri e 0.1 km
Press any key to continue . . .
```

Се поставува прашање како да се постапи за да се изврши делење на две целобројни променливи и да се добие резултат со децимални места. Решението е во користење на *операторот за доделување на тип* со кој променливите привремено се однесуваат како променливи од друг тип. Така целобројната променлива a во изразот `(float) a` се однесува како реална променлива.

Со секвенцата од наредби:

```
int a = 5;
int b = 2;
float kolicnik = (float)a/b;    //ili float kolicnik = a/(float)b;
```

на променливата kolicnik ѝ се доделува вредност 2.5.

Со употреба на операторот `float`, вредностите на променливите a и b се претвораат во реални броеви пред операцијата делење. Самите променливи a и b не го менуваат својот тип така што со повторното користење на исказот

```
kolicnik = a / b;
```

на променливата `kolicnik` ќе ѝ се додели вредност 2.

Редослед на операции

Во математиката постои утврден редослед на извршување операции: операциите се извршуваат од лево кон десно ако сите операции се од ист приоритет. Овој редослед се менува доколку има операции кои не се од ист приоритет. Операциите множење, делење и остаток при делењето имаат повисок приоритет што значи дека прво ќе се извршат овие операции па дури тогаш ќе се извршат операциите собирање и одземање. Заградите го менуваат приоритетот на начин што прво се пресметува изразот во заградите:

`6/3+3=5` `6/(3+3)=1`

Истиот редослед е усвоен и во програмскиот јазик C++. Така со исказот `a=5+7/2` променливата a ќе добие вредност 8, затоа што операцијата целобројно делење има предност.

За `a=12, b=3, c=4` вредноста на изразот `a/b*c` ќе биде `12/3*4=4*4=16`, додека вредноста на изразот `a/(b*c)` ќе биде `12/(3*4) = 12/12 = 1`.

Со кој од овие два изрази е претставен математичкиот израз $\frac{a}{bc}$?

Совет:

Ако не си сигурен/на која операција прва ќе се изврши, тоа што сакаш да се изврши прво стави го во загради. Наместо a = 5+7/2 можеш да напишеш и a = 5+(7/2).

Скратување на изрази. Оператори +=, -=, *=, /=, %=, ++, --

Често се јавува потреба од промена на вредноста на една променлива, за што се користат изрази во кои се врши некоја математичка операција врз истата таа променлива. На пр. со изразот `a=a-1;` вредноста на променливата a се намалува за 1, со изразот `s=s+a;` вредноста на променливата s се зголемува за вредноста на променливата a.

C++ поддржува пет оператори (по еден оператор за секоја аритметичка операција: `+=, -=, *=, /=, %=`) кои овозможуваат скратено пишување на ваквите изрази. Така, наместо изразот `a=a+1;` може да се напише `a+=1;`, наместо `s=s+a;` може да се напише `s+=a;`, наместо `a=a*b;` може да се напише `a*=b;` итн. Следнава табела ги содржи сите потребни информации за овие оператори:

Примери за скратување на изрази:

Скратен израз	Еквивалентно со
`a*=5`	`a=a*5`
`a/=5-b`	`a=a/(5-b)`
`a*=(b-c)+5`	`a=a*((b-c)+5))`

Пр. 4. 29. Демонстрација за скратување на изрази:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a=12;
    cout<<"a="<<a<<endl;
    a+=5;
    cout<<"po izrazot a+=5, a = "<<a<<endl;
    a-=5;
    cout<<"po izrazot a-=5, a = "<<a<<endl;
    a*=5;
    cout<<"po izrazot a*=5, a = "<<a<<endl;
    a/=5;
    cout<<"po izrazot a/=5, a = "<<a<<endl;
    a%=5;
    cout<<"po izrazot a%=5, a = "<<a<<endl;
    system("PAUSE");
    return 0;
}
```

```
a=12
po izrazot a+=5, a = 17
po izrazot a-=5, a = 12
po izrazot a*=5, a = 60
po izrazot a/=5, a = 12
po izrazot a%=5, a = 2
Press any key to continue . . .
```

Операциите *зголемување за 1* и *намалување за 1* можат да се напишат уште пократко. Во C++ постојат оператори со кои се дефинирани овие операции:

- ++ – зголемување за 1 (increment)
- -- – намалување за 1 (decrement)

Изразите:

`a=a+1;` `a+=1;` и `a++;`

имаат иста функција, сите ја зголемуваат вредноста на променливата a за 1.

За љубопитните:

C++ го добил името токму по операторот ++.

Всушност, овие два оператора (++ и --) можат да се користат на два начина: како префикс или како постфикс, зависно од тоа дали операторот се наоѓа пред променливата (++a) или по променливата (a++). И во двата случаи вредноста на променливата се зголемува, односно намалува за 1, но кога резултатот од изразите се користи во некои искази, важно е да се обрне внимание на положбата на операторот во однос на променливата. Операторот како префикс прво делува на променливата од неговата десна страна и ја менува нејзината вредност, потоа враќа нова (променета) вредност. Операторот како постфикс прво ја враќа постоечката вредност, потоа делува на променливата од неговата лева страна и ја менува нејзината вредност.

Израз	Значење
a++	ја враќа вредноста која се содржи во променливата a, па ја зголемува вредноста на a за 1
++a	ја зголемува вредноста на a за 1, па ја враќа вредноста која се содржи во променливата a
a--	ја враќа вредноста која се содржи во променливата a, па ја намалува вредноста на a за 1
--a	ја намалува вредноста на променливата a за 1, па ја враќа вредноста која се содржи во променливата a

Пр. 4. 30. Демонстрација на операцијата a++:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int x,a;
    cout<<"Vnesi cel broj: ";
    cin>>a;
    x=a++;
    cout<<"Po izvrsuvanje na operacijata a++:"<<endl
    cout<<"x = "<<x<<endl;
    cout<<"a = "<<a<<endl;
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 5
Po izvrsuvanje na operacijata a++:
x = 5
a = 6
Press any key to continue . . .
```

Пр. 4. 31. Демонстрација на операцијата ++a:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int x,a;
    cout<<"Vnesi cel broj: ";
    cin>>a;
    x=++a;
    cout<<"Po izvrsuvanje na operacijata ++a:"<<endl;
    cout<<"x = "<<x<<endl;
    cout<<"a = "<<a<<endl;
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 5
Po izvrsuvanje na operacijata ++a:
x = 6
a = 6
Press any key to continue . . .
```

Резиме

За целобројните константи и променливи се дефинирани операциите целобројно делење (/) и остаток при целобројното делење (%).

C++ подржува пет оператори (по еден оператор за секоја аритметичка операција: +=, -=, *=, /=, %=) кои овозможуваат скратено пишување на изразите со кои се врши промена на вредност на некоја променлива.

Операциите зголемување за 1 и намалување за 1 можат да се напишат уште пократко со користење на операторите ++ и -- соодветно. Овие два оператори (++ и --) можат да се користат на два начини: како префикс или како постфикс, зависно од тоа дали операторот се наоѓа пред променливата (++a) или по променливата (a++).

Вештини што треба да ги усовршиш:

Правилно да ја користиш операцијата делење со цели и со реални броеви.

Да користиш операција модул (%) со цели броеви.

Да користиш скратено запишување на изрази со операции.

Да напишеш програма со до сега изучените техники.

Прашања:

1. Пресметај!

   а) `385/100=___` б) `385%10=___`

   в) `385/10%10=____` г) `385%100/10=___`

2. Што ќе се испише на екранот по следната секвенца од наредби?

   а)
   ```
   int a = 4;
   int b = 5;
   cout<<(a+b)/2;
   ```

   б)
   ```
   int a = 4;
   int b = 5;
   cout<<(a+b)/2.;
   ```

3. Следните аритметички изрази напиши ги на програмскиот јазик C++!

   $2(a+b)$ $\frac{1}{3}x$ $\frac{a+b}{2}$

4. Колкава е вредноста на изразот `a*b/c` за целобројните променливи `a,b,c`:

   а) `a=9, b=4, c= 6?` б) `a=10, b=5, c= 4?`

5. За целобројните променливи `a,b,c`, каде `a=8, b=12, c=4, d=6` пресметај

   а) `a*b/c*d=____` б) `a*b/(c*d)=____` в) `(a*b)/(c*d)=_____`

6. Следниве изрази претстави ги во скратена форма!

   а) `a = a-9;` б) `x = x/(y+1);` в) `a=a*((b-c)+5));`

7. Колкава ќе биде вредноста на променливите a и b по извршување на следниве секвенци од искази?

   а)
   ```
   a = 1;
   b = ++a + 5;
   ```

   б)
   ```
   a = 1;
   b = a++ + 5;
   ```

Задачи:

1. Напиши програма со која ќе пресметаш средна вредност на два цели броја!

Пример 1:	Пример 2:
Vnesi dva celi broja: 6 8 Srednata vrednost na 6 i 8 e 7	Vnesi dva celi broja: 7 8 Srednata vrednost na 7 i 8 e 7.5

2. Напиши програма со која ќе ги најдеш цифрите на даден трицифрен број! Помош: погледни го прашањето 1!

Пример 1:	Пример 2:
Vnesi tricifren broj: 485 Na brojot 485 prvata cifra e 4, vtorata cifra e 8, tretata cifra e 5	Vnesi tricifren broj: 326 Na brojot 326 prvata cifra e 3, vtorata cifra e 2, tretata cifra e 6

3. Напиши програма со која време дадено во секунди ќе се прикаже во часови, минути и секунди!

Пример 1:	Пример 2:
Vnesi vreme vo sekundi: 1000 1000 sekundi se 0 casovi, 16 minuti i 40 sekundi	Vnesi vreme vo sekundi: 20000 20000 sekundi se 5 casovi, 33 minuti i 20 sekundi

4.5.4 Дополнителни спецификации на јазикот C++ (втор дел)

Константи и променливи од типот char

Знаковните константи од типот char се пишуваат помеѓу апострофи, на пр. `'a'`, `'A'`, `'+'`, `'4'` итн. Внимавај, '4' и 4 не се исти податоци! Зошто?

За чување на знаците во меморијата е предвиден 1 бајт, што значи дека може да се зачуваат 2^8=256 различни знаци. Сите знаци се претставени во ASCII табела по точно определен редослед (од 0 до 255) и секој знак има свој реден број. Така буквата `'A'` има реден број 65, буквата `'a'` има реден број 97 итн.

Препорака:

ASCII табелата погледни ја на веб страната http://www.asciitable.com/.

Во знаковна променлива може да се зачува:

- еден знак запишан помеѓу апострофи или
- ASCII вредност на тој знак (декадна вредност).

Примери на доделување вредност на знаковна променлива:

```
char bukva1='A';
char bukva2=65;
```

Во двете променливи ќе се зачува иста вредност, знакот `'A'`, и тоа во форма на број кој претставува ASCII вредност на знакот `'A'`.

Знаковните константи и променливи можат да се споредуваат:

`cout<<'a'<'b'`	Со овој исказ на екранот ќе се прикаже 1 (точно).
`cout<<'a'<'B'`	Со овој исказ на екранот ќе се прикаже 0 (неточно).

Исто така, на знаковните константи и променливи можат да се применат и математичките операции што ќе го видиме во следниот пример. Всушност, математичките операции се извршуваат врз декадните вредности на знакот во ASCII табелата.

Пр. 4. 32. Програмата ги печати претходниот и следниот знак на дадениот знак:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      char znak;
7      cout<<"Vnesi eden znak: ";
8      cin >>znak;
9      cout<<"Pred znakot "<<znak;
10     cout<<" e znakot "<<(char)(znak-1)<<endl;
11     cout<<"Po znakot "<<znak;
12     cout<<" e znakot "<<(char)(znak+1)<<endl;
13     system ("PAUSE");
14     return 0;
15 }
```

```
Vnesi eden znak: d
Pred znakot d e znakot c
Po znakot d e znakot e
Press any key to continue . . .
```

Забелешка:

Обрни внимание на употреба на операторот за доделување на тип (char) пред изразот znak - 1! Зошто мора да се користи овој оператор?

Константи и променливи од типот string

Знаковните константи најчесто се користат во низи од знаци или стрингови. Стринговите се пишуваат помеѓу наводници и до сега ги користевме за испишување на пораки на екран. Стринг со исказот `cin` може да се внесе преку тастатурата и да се зачува во некоја променлива (кога се внесува стринг преку тастатурата, тогаш тој не се запишува помеѓу наводници).

Пр. 4. 33. Програма со која се чита стринг и се зачувува во стрингова променлива:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    string ime;
    cout<<"Kako se vikas?"<<endl;
    cin>>ime;
    cout<<endl<<ime<<", dobar den!"<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Kako se vikas?
Petar

Petar, dobar den!
Press any key to continue . . .
```

Исказот `cin` не е погоден за прифаќање на стрингови кои се состојат од повеќе зборови затоа што тој чита податок до првиот знак бланко (празно место), па од стрингот може да вчита само еден збор. Зошто? Во таквите ситуации се користи функција getline.

Пр. 4. 34. Внесување на стрингови од повеќе зборови:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    string ImePrezime;
    cout<<"Kako se vikas?"<<endl;
    getline (cin,ImePrezime);
    cout<<"Zdravo, "<<ImePrezime<<"!"<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Kako se vikas?
Petar Markovski
Zdravo, Petar Markovski!
Press any key to continue . . .
```

Форматирано печатење

Веќе се запознавме со изразот endl со кој покажувачот поминува во нов ред. Сега ќе се запознаеме со уште некои техники за форматирано печатење.

За поминување во нов ред, во исказот cout се вклучува специјален знак \n.

Пр. 4. 35. Со исказот:

```
cout<<"Zdravo!\n"<<"Kako se vikas?";
```

на екранот ќе се прикаже:

```
Zdravo!
Kako se vikas?
```

За податоците да се прикажат табеларно, се користи специјален знак \t.

Пр. 4. 36. Со следниот код:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    cout<<"Red.br."<<"\tZadaca1"<<"\tZadaca2"<<"\tZadaca3"<<"\n";
    cout<<1<<"\t20"<<"\t15"<<"\t12"<<"\n";
    cout<<2<<"\t17"<<"\t25"<<"\t10"<<"\n";
    system ("PAUSE");
    return 0;
}
```

на екранот ќе се прикаже:

```
Red.br. Zadaca1 Zadaca2 Zadaca3
1       20      15      12
2       17      25      10
Press any key to continue . . .
```

Резиме

Знаковните константи од типот char се пишуваат помеѓу апострофи, на пр. `'a'`, `'A'`, `'+'` итн. Знаковните константи најчесто се користат во низи од знаци или стрингови. Исказот `cin` не е погоден за прифаќање на стрингови кои се состојат од повеќе зборови затоа што тој чита податок до првиот знак бланко, па се користи функцијата *getline*.

За форматирано печатење се користат специјалните знаци: \n за преминување во нов ред и \t за печатење во колони.

Вештини што треба да ги усовршиш:

Да користиш знаковни и стрингови променливи и константи.

Да користиш форматирано печатење.

Да напишеш програма со до сега изучените техники.

Прашања:

1. Ако буквата `'A'` во ASCII код има реден број 65, кој реден број го има буквата `'D'`?
2. Со кој тип мора да се декларира променливата x, ако сакаме да ѝ доделиме вредност '2'?
3. Објасни за што се користат специјалните знаци \n и \t!
4. Дали има разлика помеѓу следниве искази?

```
cout<<"Dobar den<<endl;   и   cout<<"Dobar den\n";
```

Задачи:

1. Напиши програма со која ќе се внесе име и презиме, а ќе се отпечати презиме па име! Помош: користи две стрингови променливи и исказот cout.

Пример 1:	Пример 2:
Kako se vikas? Mite Pavlov	Kako se vikas? Jana Kostovska
Zdravo, Pavlov Mite	Zdravo, Kostovska Jana

2. Напиши програма со која табеларно ќе ги прикажеш оценките на неколку ученици по предметите информатика, математика и физика!

4.6 СТРУКТУРА ЗА ИЗБОР ОД ДВЕ МОЖНОСТИ

Како во животот, така и во програмирањето, многу често треба да се донесе некоја одлука. Токму „способност за одлучување“ програмите ги прави корисни. На пример, за да работиш со компјутер треба да внесеш лозинка, посебна програма ја проверува лозинката и ако таа е точна дозволува работа со компјутерот, во спротивно пристапот до компјутерот е оневозможен. Во општ случај, програмата испитува некој услов и ако тој услов е задоволен се извршуваат едни инструкции. Ако условот не е задоволен се извршуваат други инструкции.

4.6.1 Споредбени изрази

Одлуките се контролираат со *логичките изрази*. Наједноставните логички изрази се изрази во кои се споредуваат две вредности. Таквите изрази се нарекуваат *споредбени изрази*. Споредување се врши само помеѓу две вредности од ист тип, на пр. можат да се споредат два броја, два знака или две низи од знаци. За градење на споредбени изрази се користат оператори за споредување:

Оператор	Математички симбол	Опис
<	<	помало
<=	≤	помало или еднакво
>	>	поголемо
>=	≥	поголемо или еднакво
==	=	еднакво
!=	≠	не е еднакво

Внимавај!

Операторите кои се состојат од два симбола, немаат празно место помеѓу нив!

За секој споредбен израз секогаш може да се утврди дали тој е точен или е неточен, односно дали има вредност 1 (точно) или 0 (неточно). Резултатот на споредување е податок од тип bool.

Примери на споредбени изрази и нивните вредности:

Израз	Вредност
`7 >= 5`	1 (точно)
`3 + 8 <= 3`	0 (неточно)
`5 == 4 - 1`	0 (неточно)
`3 != 0`	1 (точно)

Секако, освен константи, може да се користат и изрази со променливи на кои им е доделена некоја вредност. На пример, за a=2, b=3 и c=6:

Израз	Вредност
`a == 5`	0, бидејќи 2==5 не е точно
`a*b >= c`	1, бидејќи 2*3>=6 е точно
`b+4 > a*c`	0, бидејќи 3+4>2*6 не е точно

И логичките податоци, како и сите други, можат да се зачуваат во некоја променлива. Од кој тип ќе биде таа променлива?

Пр. 4. 37. Користење на променлива од тип bool:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    bool c;
    c = 5>3;
    cout<<"Dali 5>3?"<<endl<<c<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Dali 5>3?
1
Press any key to continue . . .
```

Сложени логички изрази

Логичките изрази можат да бидат и посложени, а тие се конструираат од споредбените изрази со помош на логичките оператори *И*, *ИЛИ* и *НЕ*.

Логички оператори во C++:

Оператор	Ознака за оператор	Математички симбол	Опис
AND (И)	&&	$\wedge$	Конјункција
OR (ИЛИ)	||	$\vee$	Дисјункција
NOT (НЕ)	!	$\neg$	Негација

Таблиците на вистинитост се познати од математиката:

p	q	p&&q
0	0	0
0	1	0
1	0	0
1	1	1

p	q	p||q
0	0	0
0	1	1
1	0	1
1	1	1

p	!p
0	1
1	0

При одредување на вистинитост на некој сложен израз редослед на операции е: негација, конјункција па дисјункција.

Примери на логички изрази:

Логички израз	Вредност
(1 && 0)||0	0
!((3==3) || 2>7)	0
!((5>3 || 2==5) && 2>5)	1

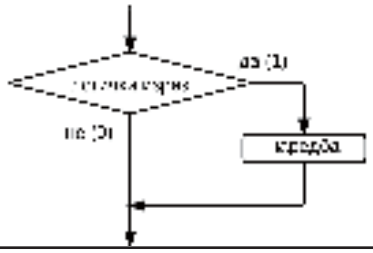
4.6.2 Структура (исказ) избор од две можности

Структурата за избор од две можности (разгранета структура) овозможува различен тек на програмата зависно од резултатот на поставениот услов. Ако изразот во условот е точен тогаш ќе се изврши некоја наредба, а ако изразот не е точен таа наредба нема да се изврши, а може но не мора да се изврши друга наредба.

Разгранувањето може да биде еднократно и двократно.

Еднократно разгранување

Ова е наједноставна форма на разгранета структура: ако некој услов е исполнет ќе се изврши некоја наредба, во спротивно таа наредба нема да се изврши.

Псевдо алгоритам за еднократно разгранување:	*Блок дијаграм за еднократно разгранување:*
ако услов *тогаш* наредба;	

Во програмскиот јазик C++ за еднократно разгранување се користи исказот `if` кој има општ облик:

```
if (logicki_izraz)
        iskaz;
iskaz_po_razgranuvanjeto;
```

Ако вредноста на логичкиот израз (`logicki_izraz`) е вистинита (`true`, `1`), се изведува исказот по изразот `if (logicki_izraz)`. Ако вредноста на логичкиот израз (`logicki_izraz`) е невистинита (`false`, `0`), исказот по изразот `if (logicki_izraz)` се прескокнува и се изведува следниот исказ по разгранувањето.

Со следниов исказ се проверува дали дадениот број x е негативен:

```
if (x < 0)
      cout<<"Brojot "<<x<<" e negativen";
```

Ако x има негативна вредност на пр. -10, логичкиот израз `x < 0` ќе биде точен, односно ќе има вредност 1, па ќе се изврши исказот

`cout<<"Brojot "<<x<<" e negativen";`.

На екранот ќе се прикаже: „`Brojot -10 e negativen`“.

Ако x нема негативна вредност, на пр. 10, тогаш логичкиот израз `x < 0` ќе биде неточен, односно ќе има вредност 0, па исказот

`cout<<"Brojot "<<x<<" e negativen";`

нема да се изврши. На екранот ништо нема да се прикаже.

Исказот може да се запише и како

```
if (x < 0) cout<<"Brojot "<<x<<" e negativen";
```

но се запишува на претходен начин заради прегледност.

Важно!

По изразот if (logicki_izraz) не се става точка и запирка (;). Ако ова го направиш преведувачот ќе мисли дека знакот ; припаѓа на наредба која треба да се изврши ако условот е исполнет, па ќе имаш несакан ефект. Помеѓу условот и знакот ; нема ништо, па преведувачот гледа празна наредба и програмата нема ништо да изврши.

Пр. 4. 38. Со следнава програма се проверува дали променливата `delitel` е еднаква на 0:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int delitel;
    cout<<"Vnesi broj: ";
    cin>>delitel;
    if (delitel == 0)
        cout<<"Ne e mozno delenje so 0!";
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 0
Ne e mozno delenje so 0!
Press any key to continue . . .
```

Ако променливата `delitel` има вредност 0 програмата ќе не извести дека со 0 не може да се дели.

Внимавај!

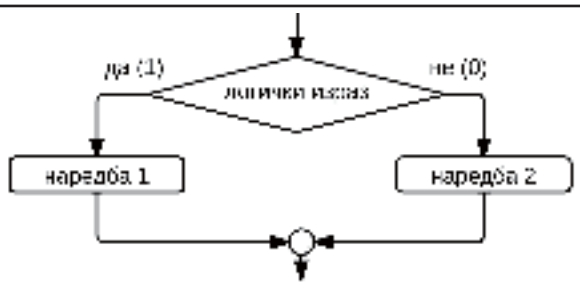
Често се случува грешка и наместо операторот == се користи операторот =. Ова може да доведе до неисправно работење на програма. Внимавај овие два оператори да не ги мешаш!

Двократно разгранување

Двократно разгранување значи: ако некој услов е исполнет ќе се изврши некоја наредба, во спротивно таа наредба нема да се изврши туку ќе се изврши друга наредба.

Псевдо алгоритам за двократно разгранување:

ако услов
тогаш наредба1
инаку наредба2;



Во програмскиот јазик C++ за двократно разгранување се користи наредба `if-else` која има општ облик:

```
if (logicki_izraz)
    iskaz1;
else
    iskaz2;
iskaz_po_razgranuvanjeto;
```

Ако вредноста на логичкиот израз (`logicki_izraz`) е вистина (true, 1), тогаш се извршува наредбата `iskaz1`. Откако таа ќе се изврши, се извршуваат наредбите по исказот `if else`.

Ако вредноста на логичкиот израз е невистина (`false, 0`), наредбата `iskaz1` се прескокнува и се извршува наредбата по `else` – `iskaz2`. Откако таа ќе се изврши, се извршуваат наредбите по исказот `if else`.

Со следниов исказ се проверува дали даден цел број е позитивен и се прикажува соодветна порака:

```
if (x < 0)
      cout<<"Brojot "<<x<<" e negativen"<<endl;
else
      cout<<"Brojot "<<x<<" ne e negativen"<<endl;
```

За `x=-8` ќе се изврши наредбата по `if` и ќе се прикаже пораката „Brojot -8 e negativen“. За `x=8` ќе се изврши наредбата по `else` и ќе се прикаже пораката „Brojot 8 ne e negativen“.

Важно!

По изразите if и else не се става знакот точка и запирка (;).

Пр. 4. 39. Програмата за два дадени различни цели броја го прикажува поголемиот број:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"Vnesi dva razlicni celi broja! ";
    cin>>a>>b;
    if (a>b)
        cout<<"Brojot "<<a<<" e pogolem."<<endl;
    else
        cout<<"Brojot "<<b<<" e pogolem."<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dva razlicni celi broja! 5 8
Brojot 8 e pogolem.
Press any key to continue . . .
```

```
Vnesi dva razlicni celi broja! 7 2
Brojot 7 e pogolem.
Press any key to continue . . .
```

Пр. 4. 40. Програмата проверува дали дадениот број е едноцифрен:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int x;
    cout<<"Vnesi cel broj! ";
    cin>>x;
    if (0 <= x && x <= 9)
        cout<<"Brojot "<<x<<" e ednocifren."<<endl;
    else
        cout<<"Brojot "<<x<<" ne e ednocifren."<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi cel broj! 6
Brojot 6 e ednocifren.
Press any key to continue . . .
```

```
Vnesi cel broj! 25
Brojot 25 ne e ednocifren.
Press any key to continue . . .
```

Важно!

Проверката дали бројот x е едноцифрен во математиката се запишува како $0 \le x \le 9$. Програмските јазици не дозволуваат ваков вид на запишување. Ова се два изрази кои мора да се поврзат со операторот && (И).

Блок од искази

Важно е да се знае дека исказот `if` контролира извршување само на првиот исказ кој се наоѓа веднаш по него. Веќе следниот исказ не е дел од исказот `if` па тој исказ ќе се изврши независно дали условот е точен или не. Да го погледнеме следниот пример:

```
if (x < 0)
        cout<<"Brojot "<<x<<" e negativen."<<endl;
        cout<<"Negovata apsolutna vrednost e "<<-x;
```

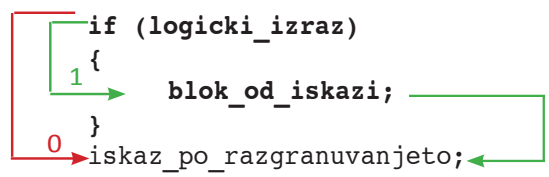
Втората наредба по исказот `if` ќе се изврши и за позитивните броеви па за нив ќе се испише погрешна апсолутна вредност.

Овој проблем се решава со градење на *блок од искази*. ***Кога исказот if контролира два или повеќе искази, сите тие се стават меѓу големи загради.***

Претходниот пример исправно се запишува на следниот начин:

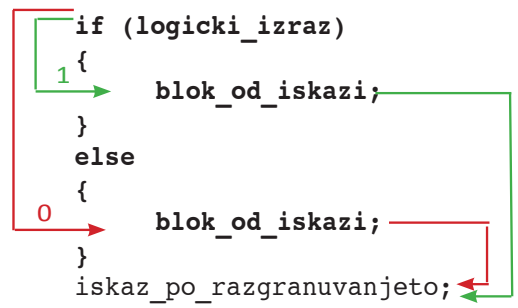
```
if (x <0)
{
        cout<<"Brojot "<<x<<" e negativen."<<endl;
        cout<<"Negovata apsolutna vrednost e "<<-x;
}
```

Во општ случај исказот за еднократно разгранување со користење на блок искази се запишува:



Ист случај е и кога има блок од искази по `else`.

Во општ случај исказот за двократно разгранување со користење на блок од искази се запишува:



Така, исказот if со кој се прикажува апсолутна вредност на бројот x се запишува:

```
if (x <0)
{
      cout<<"Brojot "<<x<<" e negativen."<<endl;
      cout<<"Negovata apsolutna vrednost e "<<-x;
}
else
{
      cout<<"Brojot "<<x<<" ne e negativen."<<endl;
      cout<<"Negovata apsolutna vrednost e "<<x;
}
```

Совет:

Нема да згрешиш ако само еден исказ ставиш помеѓу големи загради. Дури во почетокот би било добро тоа да го правиш.

Пр. 4. 41. Следнава програма од корисникот бара да внесе корисничко име и лозинка. Доколку тие се исправни ќе даде соодветна порака, во спротивно ќе даде друга порака.

```
1  #include<iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      string korisnicko_ime, lozinka;
7      cout<<"Vnesi korisnicko ime! "<<endl;
8      cin>>korisnicko_ime;
9      cout<<"Vnesi lozinka! "<<endl;
10     cin>>lozinka;
11     if (korisnicko_ime == "Darko" && lozinka == "C++")
12     {
13         cout<<"Vnesovte ispravni korisnicko ime i lozinka!";
14         cout<<"Dobredojdovte, "<< korisnicko_ime <<"!"<<endl;
15     }
16     else
17     {
18         cout<<"Korisnicko ime ili lozinka ne se tocni!";
19         cout<<"Pristapot ne e dozvolen!"<<endl;
20     }
21     system ("PAUSE");
22     return 0;
23 }
```

```
Vnesi korisnicko ime!
Darko
Vnesi lozinka!
C++
Vnesovte ispravni korisnicko ime i lozinka!Dobredojdovte, Darko!
Press any key to continue . . .
```

```
Vnesi korisnicko ime!
Darko
Vnesi lozinka!
C+
Korisnicko ime ili lozinka ne se tocni!Pristapot ne e dozvolen!
Press any key to continue . . .
```

Што ќе се случи ако наместо && ставиш ||?

Резиме

Изразите во кои се споредуваат две вредности се нарекуваат *споредбени изрази*. Операторите за споредување се: <, >, <=, >=, ==, !=. Сложените логички изрази се добиваат со поврзување на споредбените изрази со помош на логичките оператори && (И), || (ИЛИ) и ! (НЕ).

Структурата за избор од две можности (разгранета структура) овозможува различен тек на програмата зависно од резултатот на поставениот услов. Ако условот е точен ќе се изврши некоја наредба, а ако не е исполнет, таа наредба нема да се изврши, а може но не мора да се изврши друга наредба.

За еднократно разгранување се користи исказот *if*, а за двократно разгранување се користи исказот *if-else.* Кога исказот *if* контролира два или повеќе искази, тие се стават помеѓу големите загради.

Вештини што треба да ги усовршиш:

Да градиш сложени логички изрази со помош на логички оператори и да знаеш да ја провериш нивната вистинитост.

Да ги користиш структурите за еднократно и за двократно разгранување.

Да напишеш програма со користење на структурите за еднократно и за двократно разгранување.

Прашања:

1. Како се нарекуваат изразите во кои се споредуваат две вредности?
2. Кои оператори се користат за споредување?
3. Дали можат да се споредуваат вредности на променливи кои не се од ист тип?
4. Со помош на кои оператори се градат сложени логички изрази?
5. Одреди вредност на следниве споредбени изрази:

   а) 9 <= 17 б) 9 >= 25 в) 9 == 13 г) 9 != 13 д) 9 !< 25
6. Одреди вредност на следниве логички изрази:

   а) 14 > 7 && 5 <= 5 б) 4 > 3 || 17 <= 7

   в) ! (13 != 7) г) 9 != 7 && ! 0
7. Како се нарекува структура која овозможува разгранување во програма?
8. Кој исказ се користи за разгранување во програма?
9. Напиши ги синтаксите на исказите `if` и `if-else`!
10. Како се означува блок од искази кои се користат во исказите `if` и `if-else`?
11. Што ќе се отпечати со следната наредба?

```
if (x>10)
      cout<<"Brojot <<x<<" e pogolem od 10";
else
      cout<<"Brojot <<x<<" e pomal od 10";
```

    а) за x = 5 б) за x = 15.
12. Колкава ќе биде вредност на целобројната променлива `x` по извршување на следнава наредба?

```
if (x+2>10) x = x + 1;
```

    а) за x = 9 б) за x = 5.

13. Која вредност ќе ја има променливата **s** по извршување на следнава секвенца од искази:

```
s=0;
if (a>0) s=s+a;
if (b>0) s=s+b;
```

а) за a=5, b=3 б) за a=-5, b= 3 в) за a=-5, b= -3

Задачи:

1. Напиши програма која за дадениот цел број x ќе провери дали тој е парен или непарен и ќе прикаже соодветна порака!

Пример 1:	Пример 2:
Vnesi cel broj: 12 Brojot 12 e paren.	Vnesi cel broj: 35 Brojot 35 e neparen.

2. Внеси два различни броја, потоа отпечати го помалиот, па поголемиот број!

Пример 1:	Пример 2:
Vnesi dva broja: -3 7 Pomaliot broj e -3, a pogolemiot broj e 7	Vnesi dva broja: 3 -7 Pomaliot broj e -7, a pogolemiot broj e 3

3. Напиши програма која ќе пресметува периметар на квадрат само ако внесената вредност за страната на квадратот е позитивна! Во спротивно ќе даде соодветна порака.

Пример 1:	Пример 2:
Strana na kvadratot: 2.5 Perimetar na kvadratot e 10!	Strana na kvadratot: -2.5 Stranata mora da e pozitivna!

4. Напиши програма која ќе одзема во множество на природни броеви!

Пример 1:	Пример 2:
Vnesi go prviot broj: 8 Vnesi go vtoriot broj: 5 8 - 5 = 3	Vnesi go prviot broj: 5 Vnesi go vtoriot broj: 8 Razlikata 5 - 8 ne postoi vo N

5. Напиши програма со која ќе се пресмета апсолутна вредност на бројот x!

Пример 1:	Пример 2:
Vnesi eden broj: 8.2 Apsolutnata vrednost na brojot 8.2 e 8.2	Vnesi eden broj: -8.2 Apsolutnata vrednost na brojot -8.2 e 8.2

6. Напиши програма со која се собираат само позитивните вредности на броевите a, b и c!

Пример 1:	Пример 2:
Vnesi tri broja: -3 7 5 Zbirot na pozitivnite broevi e S=12	Vnesi tri broja: -3 -7 -5 Zbirot na pozitivnite broevi e S=0

7. Напиши програма со која ќе се најде најголем од дадените три броја!
8. Напиши програма со која ќе се подредат три броја по големина!
9. Напиши програма со која се одредува дали од дадените отсечки со должините a, b и c може да се формира триаголник!
10. Даден е трицифрен природен број. Напиши програма со која се одредува дали бројот е палиндром (ако исто се чита од десно и од лево)!

4.6.3 Техника на вгнездување на искази

Во блокот од искази може да се користат било кои искази, па така и исказот за избор од две можности. Впрочем, и во секојдневниот живот често се случува еден условен настан да зависи од друг условен настан и е потребно да се испитаат повеќе услови. Тогаш еден услов е надворешен, а друг е внатрешен или вгнезден.

Во програмските јазици дозволено е да се користат повеќе `if` искази вгнездени еден во друг било во блокот на искази по `if` било во блокот од искази по `else`. Ова се нарекува *техника на вгнездени искази*.

Пр. 4. 42. Во примерот 4.41 може од корисникот да се побара корисничко име и ако корисничкото име е исправно тогаш се бара лозинка. Ако корисничкото име не е исправно нема потреба да се бара лозинка.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    string korisnicko_ime, lozinka;
    cout<<"Vnesi korisnicko ime! ";
    cin>>korisnicko_ime;
    if (korisnicko_ime == "Darko")
    {
        cout<<"Vnesi lozinka! ";
        cin>>lozinka;
        if (lozinka == "C++")
        {
            cout<<"Ispravni korisnicko ime i lozinka!"<<endl;
            cout<<"Dobredojdovte, ";
            cout<< korisnicko_ime <<"!"<<endl;
        }
        else
        {
            cout<<"Pogresna lozinka!"<<endl;
            cout<<"Pristapot ne e dozvolen!"<<endl;
        }
    }
    else
        {
            cout<<"Pogresno korisnicko ime!"<<endl;
            cout<<"Pristapot ne e dozvolen!"<<endl;
        }
    system("PAUSE");
    return 0;
}
```

```
Vnesi korisnicko ime! Darko
Vnesi lozinka! C++
Ispravni korisnicko ime i lozinka!
Dobredojdovte, Darko!
Press any key to continue . . .
```

```
Vnesi korisnicko ime! Darko
Vnesi lozinka! C+
Pogresna lozinka!
Pristapot ne e dozvolen!
Press any key to continue . . .
```

Совет:

За посложените програми како што е оваа добро е да се користи блок-дијаграм.

Резиме:

Во програмските јазици дозволено е да се користат повеќе *if* искази вгнездени еден во друг, било во блокот од искази по *if*, било во блокот од искази по *else*. Ова се нарекува *техника на вгнездени искази*.

Вештини што треба да ги усовршиш:

Да препознаваш и да користиш техника на вгнездени искази.

Да напишеш програма со користење на до сега изучените техники.

Прашања:

1. Која вредност ќе ја добие променливата a по извршување на следните наредби:

```
float a = 1.56;
if (a < 1)
      a = a+1;
else
{
      a = a+4;
      a = -a+7;
}
```

2. Што ќе се прикаже по извршување на следниве искази?

```
int broj = -6;
char bukva = 'K';
if (broj < 0)
{
      if (bukva < 'S')
            cout<<1;
      else
            cout<<2;
}
else
      cout<<3;
```

Задачи:

1. Напиши програма која за даден цел број проверува дали е позитивен или негативен, ако е позитивен да се провери дали е парен или непарен и да се прикаже соодветна порака!

Пример 1:	Пример 2:	Пример 3:
Vnesi cel broj: - 6	Vnesi cel broj: 7	Vnesi cel broj: 6
Brojot -6 e negativen!	Brojot 7 e pozitiven i e neparen!	Brojot 7 e pozitiven i e paren!

2. Напиши програма со која ќе се најде решение на линеарната равенка ax + b = 0! Да се разгледаат и специјалните случаи за a=0 и b=0!

Пример 1:	Пример 2:	Пример 3:
a = 3	a = 0	a = 0
b = -21	b = -21	b = 0
Resenieto na ravenkata e x=7	Ravenkata nema resenie	Ravenkata ima beskonecno resenija

3. Да се внесат должини на три отсечки a, b и c и да се провери дали тие можат да градат триаголник. Ако можат, да се провери дали триаголникот е рамностран, рамнокрак, или разностран. За сите случаи да се прикаже соодветна порака!

4.7 Структура за избор од повеќе можности

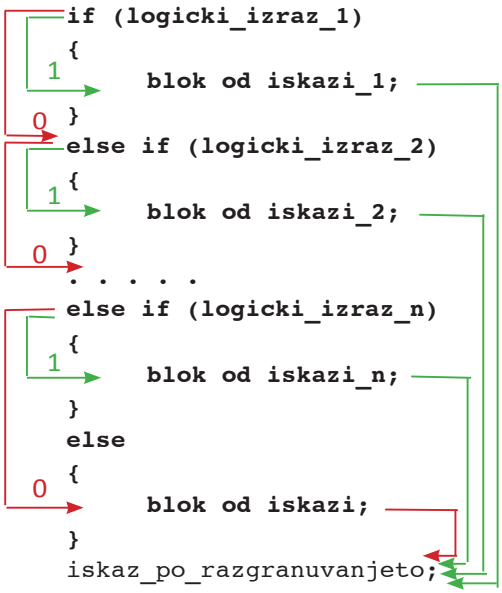
4.7.1 Повеќекратно разгранување

Да го погледнеме примерот на програма која прикажува поголем од два дадени броја:

```
if (a>b)
      cout<<"Brojot "<<a<<" e pogolem."<<endl;
else
      cout<<"Brojot "<<b<<" e pogolem."<<endl;
```

Што ќе се случи ако корисник (иако е предупреден) внесе два броја кои се еднакви? Програмата ќе провери дали a>b, и бидејќи овој услов не е исполнет, ќе го изврши исказот по else, што нема да биде исправно. За програмата да биде сосема коректна, треба да се провери и точноста на условот a==b.

Со исказот if-else може да се избира помеѓу две алтернативи и да се изврши само една од нив, зависно од вистинитоста на логичкиот израз. Но често се случува да треба да се бира помеѓу три или повеќе опции како во наведениот случај. Тогаш се применува *техника на вгнездување на искази*, односно се гради *повеќекратна разгранета структура*.



Пр. 4. 43. Примерот за поголемиот од два броја ќе го проширime со тоа што ќе се даде можност корисникот да внесе и два еднакви броја:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"Vnesi dva celi broja! "<<endl;
    cin>>a>>b;
    if (a == b)
        cout<<"Broevite se ednakvi"<<endl; //(1)
    else if (a > b)
            cout<<"Brojot "<<a<<" e pogolem"<<endl; // (2)
        else
            cout<<"Brojot "<<b<<" e pogolem"<<endl; //(3)
    system ("PAUSE");
    return 0;
}
```

Програмата проверува дали е точен изразот a == b, ако е точен ќе се исполни наредбата по делот if (наредбата коментирана со (1)). Ако изразот не е точен програмата поминува на делот по else и проверува дали е точен изразот a > b. Ако овој израз е точен програмата ќе го изврши исказот по делот if (2), инаку поминува на исказ по делот else (3).

Зад. 4. 13. Напиши програма која за даден цел број проверува дали тој број е позитивен, негативен или еднаков на 0 и печати соодветна порака!

Пр. 4. 44. Да погледнеме уште еден пример. Програмата за бројната оценка (1 - 5) печати соодветен успех (недоволен - одличен):

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int ocenka;
    cout<<"Vnesi ocenka od 1 do 5! "<<endl;
    cin>>ocenka;
    if (ocenka == 1)
        cout<<"Vasiot uspeh e nedovolen!"<<endl;    //(1)
    else if (ocenka == 2)
        cout<<"Vasiot uspeh e dovolen!"<<endl;    // (2)
    else if (ocenka == 3)
        cout<<"Vasiot uspeh e dobar!"<<endl;    // (3)
    else if (ocenka == 4)
        cout<<"Vasiot uspeh e mnogu dobar!"<<endl;    // (4)
    else if (ocenka == 5)
        cout<<"Vasiot uspeh e odlicen!"<<endl;    // (5)
    else
        cout<<"Vnesenata ocenka ne e regularna!"<<endl; //(greska)
    system ("PAUSE");
    return 0;
}
```

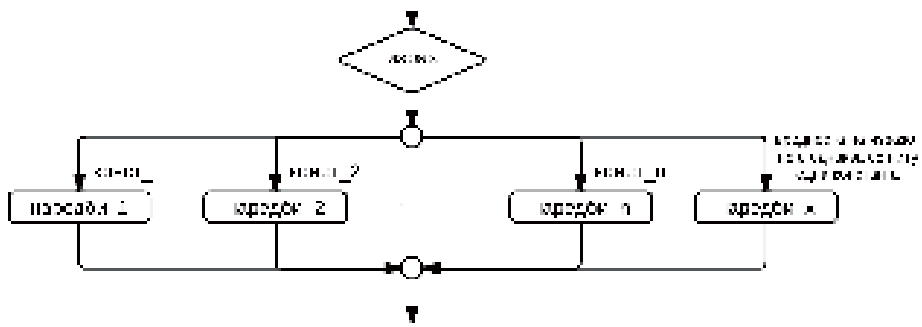
```
Vnesi ocenka od 1 do 5!
4
Vasiot uspeh e mnogu dobar!
Press any key to continue . . .
```

```
Vnesi ocenka od 1 do 5!
7
Vnesenata ocenka ne e regularna!
Press any key to continue . . .
```

4.7.2 Структура за избор од повеќе можности

Повеќекратното разгранување е добро решение кога се работи за избор од повеќе можности, но може да биде заморно и кодот да стане предолг ако се работи за поголем број на можни избори. Во програмските јазици во такви прилики се користи структурата за избор од повеќе можности.

Блок дијаграм за структурата за избор од повеќе можности:



За разлика од повеќекратното разгранување каде условите се логички изрази, кај структурата од повеќе можности условот е израз чија вредност мора да биде некоја целобројна или знаковна константа.

```
switch (izraz)
{
      case konst_1:
            blok_na_iskazi_1;
      break;
      case konst_2:
            blok_na_iskazi_2;
      break;
      . . . .
      case konst_n:
            blok_na_iskazi_n;
      break;
      default:
            posledniot_blok_na_iskazi;
}
```

Вредност на изразот се споредува со целобројните константи: `konst_1`, `konst_2`, `konst_3`, итн. Ако вредноста на изразот е еднаква на некоја од дадените константи, ќе се изврши блокот на искази придружен на таа константа. По извршувањето на тој блок, исказот `break` го прекинува извршувањето на `switch-case` исказот и програмата продолжува со првата наредба по овој исказ. Ако вредноста на изразот не е еднаква на ниту една од дадените константи, се извршува блокот на искази по `default`.

Пр. 4. 45. Програмата за успех на ученикот запишана со исказот `switch-case`:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int ocenka;
    cout<<"Vnesi ocenka od 1 do 5! "<<endl;
    cin>>ocenka;
    switch (ocenka)
    {
        case 1:
            cout<<"Vasiot uspeh e nedovolen!"<<endl;
        break;
        case 2:
            cout<<"Vasiot uspeh e dovolen!"<<endl;
        break;
        case 3:
            cout<<"Vasiot uspeh e dobar!"<<endl;
        break;
        case 4:
            cout<<"Vasiot uspeh e mnogu dobar!"<<endl;
        break;
        case 5:
            cout<<"Vasiot uspeh e odlicen!"<<endl;
        break;
        default:
            cout<<"Vnesenata ocenka ne e regularna!"<<endl;
    }
    system ("PAUSE");
    return 0;
}
```

```
Vnesi ocenka od 1 do 5!
4
Vasiot uspeh e mnogu dobar!
Press any key to continue . . .
```

```
Vnesi ocenka od 1 do 5!
7
Vnesenata ocenka ne e regularna!
Press any key to continue . . .
```

Важно!

Не заборавај на исказот break по блоковите на искази! Доколку тоа се случи, swich-case исказот нема да се прекине туку ќе продолжи со извршување на блокот од искази придружен на следната константа. Така, со следниот код ќе се отпечати DvaTri наместо Dva.

```
int x=2;
switch(x)
{
        case 1: cout<<"Eden";
        case 2: cout<<"Dva";
        case 3: cout<<"Tri";
}
```

Во исказот `switch-case` може да се користат и знаковни константи:

Пр. 4. 46. Корисникот внесува два броја и знак за аритметичка операција која сака да ја изврши. Програмата извршува соодветна операција и прикажува резултат.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    float a,b; char znak;
    cout<<"Vnesi dva broja: ";
    cin>>a>>b;
    cout<<"Vnesi znak za operacija: ";
    cin>>znak;
    switch (znak)
    {
        case '+':
            cout<<a<<znak<<b<<"="<<a+b<<endl;
        break;
        case '-':
            cout<<a<<znak<<b<<"="<<a-b<<endl;
        break;
        case '*':
            cout<<a<<znak<<b<<"="<<a*b<<endl;
        break;
        case '/':
            if (b == 0)
            {
                cout<<"Ne e dozvoleno delenje so 0!"<<endl;
            break;
            }
            cout<<a<<znak<<b<<"="<<a/b<<endl;
        break;
        default:
            cout<<"Pogresno vnesen znak!"<<endl;
    }
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dva broja: 7 12
Vnesi znak za operacija: +
7+12=19
Press any key to continue . . .
```

```
Vnesi dva broja: 15 0
Vnesi znak za operacija: /
Ne e dozvoleno delenje so 0!
Press any key to continue . . .
```

Понекогаш е потребно за различни вредности на изразот да се извршат исти наредби. На пример, со следниот код целобројната променлива broj добива вредност 1 па ќе се изврши исказот `cout << "Pomal od 3!";` со кој ќе се прикаже: „**Pomal od** 3!“:

```
int broj = 1;
switch (broj)
{
      case 0:
      case 1:
      case 2:
            cout << "Pomal od 3!";
      break;
      case 3:
            cout << "Ednakov na 3!";
      break;
      default:
            cout << "Pogolem od 3!";
}
```

Што ќе се прикаже ако на променливата **broj** наместо 1 ѝ се додели вредност 5? За кои уште вредности на променливата **broj** ќе се прикаже „**Pomal od** 3!“?

Зад. 4. 14. Напиши програма со која корисникот ќе внесе реден број на месец во година и согласно изборот на екранот ќе се отпечати на кое годишно време тој месец му припаѓа! Во случај на несоодветен влезен податок да се прикаже соодветна порака.

Резиме

Посебен случај на вгнездени услови е кога условите се вгнездуваат само во блокот на искази по делот *else*. Овој случај се нарекува *повеќекратно разгранување*. За избор од повеќе можности се користи структурата *switch-case* каде условот е израз чија вредност мора да биде некоја целобројна константа.

Вештини што треба да ги усовршиш:

Да познаваш синтакса на наредбата за повеќекратно разгранување.

Да напишеш програма со користење на досега изучените техники.

Прашања:

1. Во кои случаи се применува повеќеразгранета програмска структура?
2. Што ќе се прикаже на екранот по извршување на следниве искази за x=3?

```
int x;
cin>>x;
if (x == 1)
      cout<<"Slabo!"<<endl;
else if (x == 2)
      cout<<"Ne e loso!"<<endl;
else if (x == 3)
      cout<<"Bravo!"<<endl;
else
      cout<<"Pogresen vnes!"<<endl;
```

3. Дали постои разлика помеѓу следните кодови?

а)
```
if (znak == 1)
      broj = 10;
  else if (znak == 2)
      broj = 20;
  else
      broj = 30;
```

б)
```
switch (znak)
   {
      case 1:
           broj = 10;
       break;
       case 2:
            broj = 20;
       break;
       default:
            broj = 30;
 }
```

Задачи:

Следните задачи реши ги со двете структури!

1. Напиши програма со која корисникот ќе внесе реден број на денот во седмицата и на екранот ќе се отпечати името на соодветниот ден! Во случај на несоодветен влезен податок да се прикаже соодветна порака.

Пример 1:	Пример 2:
Vnesi reden broj na den vo sedmica: 5	Vnesi reden broj na den vo sedmica: 9
Denes e petok!	Pogresen podatok!

2. Напиши програма со која оценките A, B, C, D, E се преведуваат во бројните оценки 5, 4, 3, 2, 1! Во случај на несоодветен влезен податок да се прикаже соодветна порака!
3. Напиши програма со која корисникот ќе внесе реден број на месец во година и согласно изборот на екранот ќе се отпечати колку денови има соодветниот месец! Да се предвиди ситуација за престапна година! Во случај на несоодветен влезен податок да се прикаже соодветна порака!
4. Напиши програма со која корисникот ќе избере број пред хороскопскиот знак и согласно изборот ќе добие краток хороскоп! Во случај на несоодветен влезен податок да се прикаже соодветна порака!

4.8 Основна структура за повторување

Често пати се јавува потреба една или повеќе инструкции да се извршат повеќе пати. На пример: Додека има хартија печати покани, Напиши 100 пати „Никогаш повеќе нема да зборувам на час“ и слично. И во програмирањето многу често се јавува потреба од повторување на еден ист исказ (или на блок од искази) повеќе пати. За да не се повторува пишувањето на исказот, во програмскиот код се користи структура за повторување која се вика циклус, петелка или јамка (loop).

Структурите за повторување овозможуваат некој блок од искази да се изврши одреден број пати. При тоа, бројот на повторувањата на циклусот е дефиниран со однапред зададен природен број или зависи од некој услов кој определува кога повторувањето ќе се прекине, при што условот може да се испита пред почетокот на циклусот или по неговото завршување.

Секое повторување на циклусот се нарекува *итерација*.

4.8.1 Структура за повторување на циклус до исполнување на услов

Наједноставниот од сите циклуси е `while` циклусот. Во овој циклус прво се проверува дали некој услов е исполнет. Ако условот е исполнет, циклусот се повторува, ако условот не е исполнет тогаш не се влегува во циклусот.

Пр. 4. 47. Со кодот:

```
int i = 1;
while (i <= 10)
{
        cout<<i<<"\t";
        i++;
}
```

ќе се прикаже:

```
1   2   3   4   5   6   7   8   9   10
```

Пр. 4. 48. Со кодот:

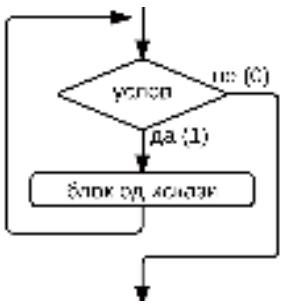
```
int i = 1;
while (i <= 10)
{
        cout<<'A'<<"\t";
        i++;
}
```

ќе се прикаже:

```
A   A   A   A   A   A   A   A   A   A
```

Основна форма на while циклусот е:

```
while (uslov)
{
   blok_od_iskazi;
}
iskaz_po_ciklusot;
```



- Условот во циклусот може да биде споредбен или логички израз кој може да добие вредност точно (`true, 1`) или неточно (`false, 0`).
- Блокот од искази во циклусот се повторува сè додека условот е исполнет. Бидејќи условот се проверува на почетокот може да се случи телото на циклусот да не се изврши ниту еднаш.
- Кога условот нема да биде исполнет, блокот од искази во циклусот се прескокнува и програмата ќе продолжи со извршување на првата наредба по телото на циклусот.
- Структурата `while` мора да биде напишана така што во конечен број на повторувања логичкиот израз ќе добие вредност неточно. Така се обезбедува да се излезе од циклусот по конечен број на повторувања. Во спротивно циклусот би се повторувал бесконечно.

Важно!

По изразот while (uslov) не се става точка и запирка (;).

Пр. 4. 49. Програмата испишува броеви наназад. Корисникот внесува број од кој ќе започне испишувањето.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Ke brojam nanazad!"<<endl;
    cout<<"Od koj broj da zapocnam? ";
    cin >>n;
    while (n > 0)
    {
        cout<<n<<endl;
        n--;
    }
    system ("PAUSE");
    return 0;
}
```

```
Ke brojam nanazad!
Od koj broj da zapocnam? 5
5
4
3
2
1
Press any key to continue . . .
```

Блокот од искази кои го сочинуваат циклусот се ставени во големите загради. Кои се тие искази?

На почетокот на програмата, пред циклусот, од корисникот се бара да внесе број од кој ќе започне броењето со што се определува дали циклусот ќе се повторува или не. Циклусот ќе се изврши само ако корисникот внесе број поголем од 0 (тоа е определено со условот (`n>0`)).

Ако корисникот внесе број 5 циклусот ќе се повтори точно 5 пати. Во секое повторување програмата го испишува бројот на екранот, потоа тој број го намалува за еден и повторно го проверува условот (`n>0`). Циклусот ќе се повторува сè додека бројот е поголем од 0. Кога бројот ќе добие вредност 0 изразот `n>0` ќе има вредност невистина (`false, 0`) и програмата ќе продолжи со извршување на првата наредба по циклусот. Која е тоа наредба?

Да видиме што се случува при извршување на програмата:

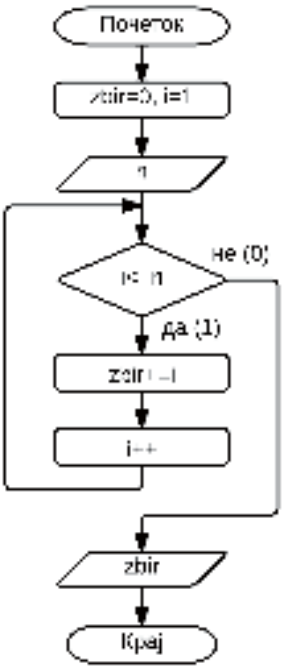
`n --;`	`(n>0)`	`cout<<n;`	Број на повторувања
5	1	5	1
4	1	4	2
3	1	3	3
2	1	2	4
1	1	1	5
0	0		

Што ќе се случи ако корисникот внесе 0 или негативен број? Циклусот нема да се изврши ниту еднаш, затоа што прво се проверува условот и во телото на циклусот се влегува само ако условот е исполнет, односно ако логичкиот израз даден во условот има вредност вистина (`true, 1`).

Ако бројот во секое повторување не се намалува за 1 тогаш логичкиот израз даден во условот (`n>0`) секогаш ќе биде точен (`true, 1`) и циклусот ќе се повторува бесконечен број пати, односно додека програмата не ја истроши работната меморија со која располага. Коментирај го исказот `n--;` и провери!

Пр. 4. 50. Програмата пресметува збир на првите n природни броеви. Бројот n го внесува корисникот.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n;
    int i=1;
    int zbir=0;
    cout<<"Do koj broj da sobiram? ";
    cin>>n;
    while (i<=n)
    {
        zbir+=i;
        i++;
    }
    cout<<"Zbirot na broevite od "<<1;
    cout<<" do "<<n<<" iznesuva "<<zbir<<endl;
    system ("PAUSE");
    return 0;
}
```




```
Do koj broj da sobiram? 6
Zbirot na broevite od 1 do 6 iznesuva 21
Press any key to continue . . .
```

Пр. 4. 51. Корисникот внесува n цели броеви. Да се најде нивниот збир.

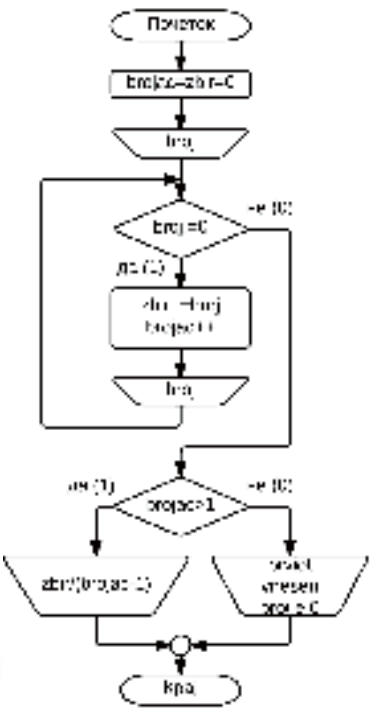
```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n, broj;
    int i = 1;
    int zbir = 0;
    cout<<"Kolku broevi ke vnesete? ";
    cin>>n;
    while (i<= n)
    {
        cout<<"Vnesi broj: "<<"\t";
        cin>>broj;
        zbir = zbir + broj;
        i++;
    }
    cout<<"Zbirot e "<<zbir<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Kolku broevi ke vnesete? 3
Vnesi broj:     6
Vnesi broj:     -12
Vnesi broj:     15
Zbirot e 9
Press any key to continue .
```

Зад. 4. 15. Измени ја претходната програма така да се собираат само позитивни броеви! Ако корисникот внесе негативен број тој нема да влезе во збирот.

Пр. 4. 52. Корисникот внесува броеви, внесувањето се прекинува кога ќе внесе бројот 0. Да се најде аритметичката средина на внесените броеви!

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int broj, brojac, zbir;
    brojac = zbir = 0;
    cout<<"Vnesi broj: ";
    cin>>broj;
    while (broj!=0)
    {
        zbir = zbir + broj;
        brojac++;
        cout<<"Vnesi broj: ";
        cin>>broj;
    }
    if (brojac!=0)
    {
        cout<<"Aritmeticka sredina na";
        cout<<" vnesenite broevi e ";
        cout<<(float)zbir/brojac<<endl;
    }
    else
        cout<<"Prviot vnesen broj e 0!"<<endl;
    system("PAUSE");
    return 0;
}
```




```
Vnesi broj (0 za kraj): 3
Vnesi broj: -2
Vnesi broj: 5
Vnesi broj: 6
Vnesi broj: 0
Aritmeticka sredina na vnesenite broevi e 3
Press any key to continue . . .
```

```
Vnesi broj (0 za kraj): 0
Prviot vnesen broj e 0!
Press any key to continue .
```

На почетокот на програмата, пред циклусот, од корисникот се бара да го внесе првиот број (`cin>>broj`) со што се определува дали циклусот воопшто ќе се изврши. Ако првиот внесен број е 0 ќе се прикаже соодветна порака.

4.8.2 Структура за повторување do-while

Во структурите за повторување кај кои бројот на повторувања на исказите од циклусот зависи од некој услов, условот може да се испита и на крајот на циклусот. Во овој случај, во програмскиот јазик C++ се користи `do-while` структура. Оваа структура поретко се користи од структурата `while`.

Пр. 4. 53. Со кодот:

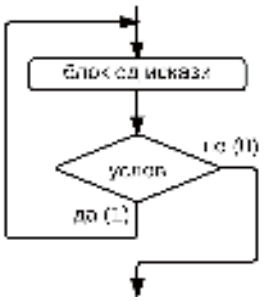
```
int i = 1;
do
{
        cout<<i<<"\t";
        i++;
}
while (i <= 10);
```

ќе се прикаже:

1 2 3 4 5 6 7 8 9 10

Основна форма на do-while циклусот е:

```
do
{
   blok_od_iskazi;
}
while (uslov);
```



Со структурата do-while условот се проверува на крајот од циклусот. Ако условот е исполнет, блокот на искази во циклусот ќе се повтори, а ако условот не е исполнет програмата ќе продолжи со следната наредба по циклусот.

Структурата do-while мора да биде напишана така што во конечен број на повторувања логичкиот израз ќе добие вредност неточно. Така обезбедува да се излезе од циклусот по конечен број на повторувања. Во спротивно циклусот би се повторувал бесконечно.

Внимавај:

Не заборавај по циклусот do-while да ставиш точка и запирка по условот.

Пр. 4. 54. Програмата испишува броеви наназад. Корисникот внесува број од кој ќе започне испишувањето.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Ke brojam nanazad!"<<endl;
    cout<<"Od koj broj da zapocnam so odbrojuvanje? ";
    cin >>n;
    do
    {
        cout<<endl<<n;
        n--;
    }
    while (n>0);
    cout<<endl<<"Odbrojuvanjeto zavrseno!"<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Ke brojam nanazad!
Od koj broj da zapocnam so odbrojuvanje? 5

5
4
3
2
1
Odbrojuvanjeto zavrseno!
Press any key to continue . . .
```

Истата задача ја решивме и со циклусот while. Да ги споредиме!

Кај `do-while` циклусот, блокот на искази од циклусот ќе се изврши барем еднаш независно дали условот е исполнет или не. Зошто?

Блок од искази кои го сочинуваат циклусот, исто така, се ставени во големи загради. Тоа се исти искази како во `while` циклусот.

На почетокот на програмата, пред циклусот, од корисникот се бара да внесе број од кој ќе започне броењето.

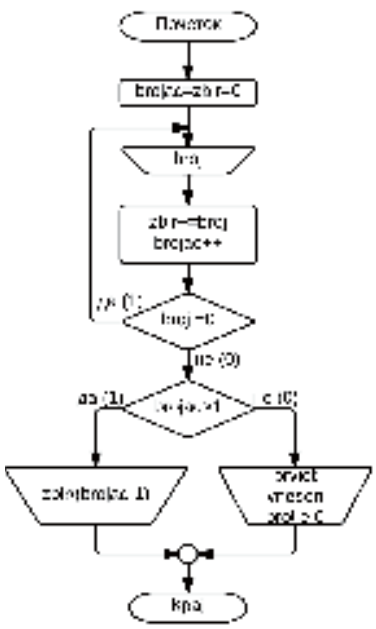
Ако корисникот внесе број 5 циклусот ќе се повтори точно 5 пати. Во секое повторување програмата го испишува бројот на екранот, потоа тој број го намалува за еден и повторно го проверува условот `(n>0)`. Циклусот ќе се повторува сè додека бројот е поголем од 0. Кога бројот ќе добие вредност 0 изразот `broj>0` ќе има вредност невистина `(false, 0)` и програмата ќе продолжи со извршување на прва наредба по телото на циклусот. Која е тоа наредба?

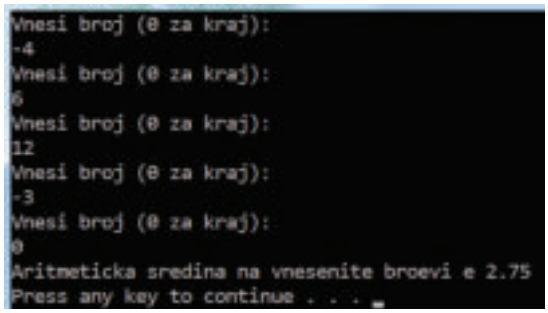
Што ќе се случи ако корисникот внесе `0` или негативен број? Независно од тоа кој број е внесен, блокот од искази во циклусот ќе се изврши, затоа што прво се влегува во циклусот, а условот се проверува на крајот од циклусот. Ова е основната разлика помеѓу `while` и `do-while` структурата.

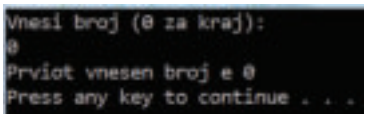
Ако бројот во секое повторување не се намалува за 1 тогаш логичкиот израз даден во условот `(n>0)` секогаш ќе биде точен `(true, 1)` и циклусот ќе се повторува бесконечен број пати, односно додека програмата не ја истроши работната меморија со која располага.

Пр. 4. 55. Корисник внесува броеви, внесувањето се прекинува кога ќе внесе бројот 0. Да се најде аритметичката средина на внесените броеви! Ако првиот внесен број е 0 да се прикаже соодветна порака.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int broj, brojac, zbir;
    brojac = zbir = 0;
    do
    {
        cout<<"Vnesi broj: "<<endl;
        cin>>broj;
        zbir+=broj;
        brojac++;
    }
    while (broj!=0);
    if (brojac>1)
    {
        cout<<"Aritmeticka sredina na";
        cout<<" vnesenite broevi e ";
        cout<<(float)zbir/(brojac-1)<<endl;
    }
    else
        cout<<"Prviot vnesen broj e 0!"<<endl;
    system("PAUSE");
    return 0;
}
```







Резиме

Структурите за повторување овозможуваат некој блок од искази да се изврши одреден број пати. Секое повторување на циклусот се нарекува *итерација*.

Во *while* циклусот прво се проверува дали некој услов е исполнет. Ако условот е исполнет циклусот се повторува, ако условот не е исполнет тогаш не се влегува во циклусот. Исказите од циклусот може да не се извршат ниту еднаш.

Во *do-while* циклусот условот се проверува на крајот од циклусот. Ако условот е исполнет циклусот се повторува, ако условот не е исполнет се излегува од циклусот. Исказите од циклусот мора барем еднаш да се извршат.

Прашања:

1. Што овозможуваат структурите за повторување?
2. Како уште се нарекува структурата за повторување?
3. Што е итерација?
4. Со што е дефиниран бројот на повторувањата на циклусите?
5. Што ќе се прикаже на екранот со следните кодови?

а)
```
int i = 2;
while (i <= 10)
{
    cout<<i<<" ";
    i+=2;
}
```

б)
```
int i = 20;
while (i <= 10)
{
     cout<<i<<" ";
     i+=2;
}
```

6. Што ќе се прикаже на екранот со следните кодови?

а)
```
int i = 2;
do
{
    cout<<i<<" ";
    i+=2;
}
 while (i <= 10);
```

б)
```
int i = 20;
 do
 {
     cout<<i<<" ";
     i+=2;
 }
 while (i <= 10);
```

7. Колку пати ќе се изврши следниот циклус?

а)
```
int x= 13;
 while (x<= 13)
 {
    x++;
    cout<<x;
 }
```

б)
```
int x= 0;
 while (x<= 10)
 {
    x+=2;
    cout<<x;
 }
```

8. Програмата 10 пати го отпечати знакот *. Пополни ги празните места:

```
int x=__;
while (x<=10)
{
      _____<<"*";
      x_____;
}
```

9. Постави ги изразите во исправен редослед (напиши редни броеви на цртичките):

```
___   do                              ___   (proizvod <100000);
___   proizvod = proizvod *x;         ___   {
___   cin<<x;                        ___   proizvod=1;
___   while                         ___   }
```

Задачи:

1. Напиши програма со која се пресметува збир на реципрочните вредности на првите n природни броеви (1 + 1/2 + 1/3 + 1/4 + 1/5 + ... + 1/n)! Бројот n го внесува корисникот.
2. Напиши програма со која се пребројуваат и собираат парните броеви во опсегот од m до n!
3. Корисникот внесува n цели броеви. Напиши програма со која ќе се преброи колку од внесените броеви се позитивни, а колку негативни!
4. Корисникот внесува броеви сè додека нивниот збир е помал од 100. Напиши програма со која ќе се изброи колку броеви се внесени!
5. Корисникот внесува природен број. Напиши програма со која се пребројува колку цифри има бројот!
6. Корисникот внесува природен број. Напиши програма со која ќе се отпечатат цифрите на внесениот број почнувајќи од цифрата на единици!
7. Корисникот внесува броеви, внесувањето се прекинува кога ќе внесе 0. Напиши програма со која ќе се прикаже најголемиот број од внесените броеви!

4.9 Останати структури за повторување

4.9.1 Структура за повторување на циклус со броење на циклусите

Кога однапред е познат бројот на повторување на циклусот најчесто се користи `for` циклусот.

Пр. 4. 56. Кодот со кој се печатат првите 10 природни броеви изгледа вака:

```
for (i = 1; i <= 10; i++)
      cout<<i<<" ";
```

По клучниот збор `for` следуваат три изрази во заграда кои го контролираат циклусот

```
(i = 1; i <= 10; i++).
```

Со првиот израз се определува почетната вредност на променливата `i(i=1;)`. Вториот израз е логички израз кој кажува до кога циклусот ќе се извршува (додека `i<=10;`). Третиот израз кажува за колку вредност на променливата `i` ќе се промени.

Исказот кој се извршува во циклусот (`cout<<i<<" ";`) следува по изразот `for`. Доколку има блок од искази кои се повторуваат во циклусот, тие се стават помеѓу големи загради.

Општа форма на for циклусот е:

```
for (pocetna_vrednost; uslov; promena)
{
        blok_na_iskazi;
}
```

Секој `for` циклус има своја контролна променлива на која вредноста ѝ се менува со секое извршување на циклусот што е одредено со вредноста на прирастот (оваа вредност може да биде и негативна). Условот мора да биде логички израз кој може да добие вредност точно `(true, 1)` или неточно `(false, 0)`. Блокот од искази кој се наоѓа во циклусот се извршува додека вредноста на условот е вистина. Кога условот ќе добие вредност невистина, циклусот се прекинува.

Внимавај!

По изразот for нема точка и запирка (;), како ниту по третиот израз внатре во заградите.

C++ дозволува контролната променлива да се декларира во самиот `for` исказ. На пример,

```
for (int i = 1; i <= n; i++)
      cout<<i;
```

е исто што и:

```
int i;
for (i = 1; i <= n; i++)
      cout<<i;
```

Зад. 4. 16. Што ќе се отпечати со следниов код?

```
for (i = 50; i > 20; i = i - 2)
      cout<<i<<" ";
```

Колкава е почетната вредност на контролната променлива? Колкава е промената? До кога циклусот ќе се извршува?

Ако наместо `i=i-2` се напише `i=i+2`, вредноста на променливата `i` во секоја итерација ќе се зголемува за 2 и секогаш ќе биде поголема од 20, во овој случај се добива бесконечен циклус.

Ако почетната вредност на променливата `i` е 5, циклусот нема да се изврши бидејќи условот при првата проверка е невистинит.

Пр. 4. 57. Програма со која се пресметува производ на првите n природни броеви. Бројот n го внесува корисникот.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int i, n;
    int proizvod = 1;
    cout<<"Do koj broj da mnozam? "<<endl;
    cout<<"Vnesi priroden broj: ";
    cin>>n;
    for (i = 1; i<=n; i++)
        proizvod = proizvod * i;
    cout<<"Proizvodot na broevite od "<<1;
    cout<<" do "<<n<<" e "<<proizvod<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Do koj broj da mnozam?
Vnesi priroden broj: 5
Proizvodot na broevite od 1 do 5 e 120
Press any key to continue . . .
```

На променливата `proizvod` ѝ се доделува почетна вредност 1. Зошто?

Резиме

Кога однапред е познат бројот на повторување на циклусот најчесто се користи *for* циклусот.

Секој *for* циклус има своја контролна променлива на која вредноста и се менува со секое извршување на циклусот што е одредено со вредноста на прирастот. Блокот од искази кој се наоѓа во циклусот се извршува додека вредноста на условот е вистина.

Прашања:

1. Кога се користи `for` циклусот?
2. Колку пати ќе се повтори циклусот?

а)
```
for(int i=0;i<10;i++)
    cout<<i;
```

б)
```
for(int i=0;i<10;i=i+2)
    cout<<i;
```

в)
```
for(int i=0;i<=10;i++)
    cout<<i;
```

г)
```
for(int i=0;i<=10;i=i+2)
    cout<<i;
```

3. Која вредност ќе се прикаже по извршување на следниот код?

а)
```
int br=0;
for(int a=1;a<=20;a++)
    if (a%2==0)
    br++;
cout<<br;
```

б)
```
br=0;
for(int a=1;a<=20;)
    if (a%2==0)
    br++;
cout<<br;
```

4. Што ќе се прикаже со следниот код?

а)
```
for(int i=10;i>8;i--)
    cout<<"Ana";
```

б)
```
for(int i=5;i>8;i--)
    cout<<"Ana";
```

5. Напиши го while циклусот како for циклус!

```
int i=1;
while(i<=10)
{
        if(i<5 && i!=2 )
        cout<<'x';
        i++;
}
```

6. Напиши го do-while циклусот како for циклус!

```
int n = 100;
do
{
        cout<<'x';
        n-=10;
}
while(n>0);
```

Задачи:

1. Напиши програма со која ќе се прикажат сите трицифрени броеви на кои последната цифра им е 0!
2. Напиши програма со која ќе се пресмета x^n, n е природен број!
3. Корисник внесува 10 броеви. Напиши програма со која ќе се соопшти дали повеќе се внесени позитивните или негативните броеви!
4. Напиши програма со која се пребројуваат парните броеви во опсегот од m до n! Ако m>n да се заменат вредностите на променливите m и n!
5. Напиши програма со која ќе се најдат и прикажат сите делители на природен број n!

4.9.2 Дополнителни структури за повторување

Вгнездени циклуси

Понекогаш е потребно некои активности да се извршат одреден број пати, а потоа истите тие активности да се повторат уште неколку пати. На пример, ако 5 играчи играат карти, на сите 5 играчи треба да им се поделат по 4 карти. Или професор треба да прегледа тестови во 3 паралелки, во секоја паралелка прегледува онолку тестови колку ученици има во таа паралелка. Во ваквите случаеви циклусите се вгнездуваат еден во друг.

На пример,

со кодот

```
int i,j;
for (i=1;i<=3;i++)
{
   for (j=1;j<=3;j++)
   {
      cout<<i<<" "<<j<<"\t";
   }
 cout<<"\n";
}
```

на екранот ќе се прикаже:

```
1 1   1 2   1 3
2 1   2 2   2 3
3 1   3 2   3 3
```

Пр. 4. 58. Пример за вгнездени for циклуси. Со програмата се прикажува таблица на множење од 1 до 10:

```
#include <iostream>
#include <cstdlib>
#include <iomanip>
using namespace std;
int main()
{
    int i, j;
    for(i=1;i<=10;i++)
    {
        for( j=1; j<=10; j++)
        {
            cout<<setw(3)<<i*j;
        }
        cout <<"\n";
    }
    system ("PAUSE");
    return 0;
}
```

```
  1   2   3   4   5   6   7   8   9  10
  2   4   6   8  10  12  14  16  18  20
  3   6   9  12  15  18  21  24  27  30
  4   8  12  16  20  24  28  32  36  40
  5  10  15  20  25  30  35  40  45  50
  6  12  18  24  30  36  42  48  54  60
  7  14  21  28  35  42  49  56  63  70
  8  16  24  32  40  48  56  64  72  80
  9  18  27  36  45  54  63  72  81  90
 10  20  30  40  50  60  70  80  90 100
Press any key to continue . . .
```

При влезот во надворешниот циклус, контролната променлива `i` добива вредност 1, се проверува условот на надворешниот циклус (`i<=10;`), па ако условот е исполнет се влегува во внатрешен циклус.

При влезот во внатрешниот циклус, контролната променлива `j` добива вредност 1, се проверува условот на надворешниот циклус (`j<=10;`), па ако условот е исполнет се извршуваат искази од внатрешниот циклус (`cout<<setw<<i*j;`).

Исказите од внатрешниот циклус ќе се извршуваат се додека условот (j<=10;) е исполнет. Кога овој услов нема да биде исполнет, се излегува од внатрешниот циклус и повторно започнува да се извршува надворешниот циклус. Вредноста на контролната променлива за надворешниот циклус се менува (i++), се проверува условот на надворешниот циклус (`i<=10;`), па ако тој е исполнет се влегува во внатрешен циклус. Сега се повторува опишаната постапка на извршување на внатрешниот циклус.

Сè ова се повторува додека условот за надворешниот циклус е исполнет, кога овој услов нема да биде исполнет се излегува од надворешниот циклус. За секоја вредност на контролната променлива `i` на надворешниот циклус се извршува целиот внатрешен циклус.

Забелешка:

За испишувањето да биде во правилни колони, се користи операторот за форматирано печатење setw(5) со кој се одредува колкав простор ќе се предвиди за испишување на податоците. Овој оператор е вклучен во библиотеката iomanip, па таа библиотека е вклучена со наредбата include.

Исказите break и continue

Се запознавме со сите три структури за повторување во јазикот C++ и тоа е вообичаен начин на кој овие структури се користат. Сепак, начинот на нивното извршување може да се смени со помош на `break` и `continue` исказите.

Со исказот `break` се прекинува понатамошното извршување на циклусот во кој овој исказ се наоѓа. Со овој исказ веќе се запознавме во `switch-case` структурата.

Со исказот `continue` се прекинува извршување само на моментната итерација на циклусот во кој овој исказ се наоѓа (таа итерација се прескокнува).

Пр. 4. 59. Пример за користење на исказот `break`:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    for (n=10; n>0; n--)
    {
        cout << n << ", ";
        if (n==5)
        {
            cout <<"\nOdbrojavanjeto e prekinato!
            break;
        }
    }
    system ("PAUSE");
    return 0;
}
```

```
10, 9, 8, 7, 6, 5,
Odbrojavanjeto e prekinato!
Press any key to continue . . .
```

Пр. 4. 60. Пример за користење на исказот `continue`:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    for (n=10; n>0; n--)
    {
        if (n==5)
        continue;
        cout << n << ",";
    }
    cout <<"\nOdbrojuvanje bez brojot 5!\n";
    system ("PAUSE");
    return 0;
}
```

```
10,9,8,7,6,4,3,2,1,
Odbrojuvanje bez brojot 5!
Press any key to continue . . .
```

Исказот continue може да предизвика бесконечен циклус, па се препорачува наместо овој исказ да се користи исказот за разгранување.

Исказот goto

Исказот `goto` е исказ за безусловен скок. Овој исказ овозможува извршувањето на програмата да продолжи од избраниот исказ. Исказот со кој понатаму се продолжува може да биде каде било во кодот, но мора да биде означен со симболичко име по кое се става знакот две точки (:).

Во општ случај наредбата goto се користи во следната форма:

```
ime: iskazot_na_koj_se_pominuva;
...
goto (ime);
```

Пр. 4. 61. Пример за користење на исказот `goto`:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a, P;
    vnes: cout<<"Vnesi strana na kvadratot: ";
    cin>>a;
    if (a<0)
    {
        cout<<"Strana ne moze da bide negativna!\n";
        goto vnes;
    }
    P=a*a;
    cout <<"Plostinata na kvadratot e "<<P<<"\n";
    system ("PAUSE");
    return 0;
}
```

```
Vnesi strana na kvadratot: -6
Strana ne moze da bide negativna!
Vnesi strana na kvadratot: 6
Plostinata na kvadratot e 36
Press any key to continue . . .
```

Бесконечен циклус

Може да се случи циклус да се извршува бесконечно пати ако услов во циклусот секогаш е исполнет. Програма во која е бесконечен циклус може да се прекине со затворање на прозорецот во кој програмата се извршува.

Пр. 4. 62. Пример на бесконечен циклус:

```
for(i=3;i<5;)
        cout<<"Beskonecen ciklus";
```

За да одбегнеш бесконечен циклус:

- води сметка услов во циклусот некогаш да мора да добие вредност неточно,
- во исказот for, во заградите, наведи ги сите три изрази,
- одбегнувај во циклусот да менуваш вредност на контролната променлива.

Резиме

Дозволено е циклуси да се вгнездуваат еден во друг. При тоа постои внатрешен и надворешен циклус.

Со исказот *break* се прекинува понатамошното извршување на циклусот во кој овој исказ се наоѓа. Со исказот *continue* се прекинува извршување само на моментната итерација на циклусот во кој овој исказ се наоѓа. Исказот *goto* е исказ за безусловен скок. Овој исказ овозможува извршувањето на програмата да продолжи од избраниот исказ.

Вештини што треба да ги усовршиш:

Да познаваш и да користиш дополнителни структури за повторување.

Да препознаваш и да знаеш како да одбегнеш бесконечен циклус.

Да напишеш програма со користење на досега изучените техники.

Прашања:

1. Дали е дозволено вгнездување на циклуси во C++?
2. Кога се користи исказот break?
3. Кога се користи исказот continue?
4. Кога се користи исказот goto?
5. Што е бесконечен циклус?
6. Како може да се одбегне бесконечен циклус?
7. Спореди ги следниве примери за користење на исказот continue:

а)

```
int n=11;
while(n>1)
{
      n--;
      if (n==5)
            continue;
      cout << n << ", ";
}
cout << "\nOdbrojuvanje bez brojot 5!";
```

Екран:

```
10, 9, 8, 7, 6, 4, 3, 2, 1,
Odbrojuvanje bez brojot 5!
```

б)

```
int n=10;
while(n>=1)
{
      if (n==5)
            continue;
      cout << n << ",";
      n--;
}
cout << "\nOdbrojuvanje bez brojot 5!";
```

Екран:

```
10, 9, 8, 7, 6,
Odbrojuvanje bez brojot 5!
```

Зошто во вториот пример одбројувањето е прекинато? Каде е грешката?

Задачи:

1. Напиши програма која ќе ги прикаже делителите на броевите од 10 до 100!
2. Корисник внесува 10 броеви. Напиши програма со која ќе се пребројат и ќе се најде збир само на позитивните броеви! Во случај корисникот да внесе негативен број да се прикаже соодветна порака и да се прекине циклусот!

```
Vnesi broj: 5
Vnesi broj: -2
Vnesi broj: 10
Vnesen e negativen broj, toj ne se broi i ne se sobira!
Vneseni se 2 broevi, nivniot zbir e 15!
```

3. Напиши програма со која се печатат броеви од 1 до n! Бројот n го внесува корисникот. Доколку корисникот внесе број помал или еднаков на 1 да се отпечати соодветна порака и корисникот да се упати повторно да го внесе бројот n!

4.10 Примери за посложени алгоритми и програми

Пр. 4. 63. Да се провери дали внесениот број е прост!

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;   // brojot sto se vnesuva
    int k;   // kandidat za delitel
    cout<<"Vnesi broj n(n>1): ";
    cin>>n;
    if(n%2==0 && n!=2) //(1)
    {
        cout<<"Brojot ne e prost!\n";
    }
    else    //(2)
    {
        k=3; //prviot kandidat za delitel e brojot 3
        while(k <= n/2) //dodeka kandidatot e pomal od n/2
        {
            if( n%k==0) //prasuvame dali toj broj e delitel na brojot n
            {
                cout<<"Brojot ne e prost!\n";
                //ako e, pecatime deka n ne e prost broj
                break; // i go prekinuvame ciklusot
            }
            k=k+2;
            //inaku sledniot kandidat za delitel e sledniot neparen broj
        }
        if(k>n/2)
            //ako ciklusot ne e prekinat so naredbata break, brojot n e prost
            cout<<"Brojot e prost!\n";
    }
    system ("PAUSE");
    return 0;
}
```

Објаснување:

//(1) Сите парни броеви се делат со 2 па тие не се прости броеви. Исклучок е бројот 2 кој е прост број. Ако корисникот внесе парен број кој е различен од 2, веднаш може да се отпечати дека тој број не е прост.

//(2) Треба да се провери дали внесениот број се дели со уште некој број освен со 1 и со n. Бидејќи се работи само со непарните броеви, кандидати за нивните делители можат да бидат само броевите 3, 5, 7 итн. заклучно со најголемиот број кој е помал од n/2 (ниту еден број нема делител поголем од n/2, а да тоа не е бројот n). Во некој циклус, на пр. во циклусот while, може да се провери дали кандидатите за делител навистина се делители на бројот n (if(n%k==0)). Ако има таков делител, веднаш штом тој ќе се најде може да се отпечати порака дека бројот n не е прост број и циклусот ќе се прекине (break;).

Зад. 4. 17. Да се најдат сите прости броеви од бројот 1 до бројот n кој го внесува корисникот!

Пр. 4. 64. Да се најде најголемиот заеднички делител на два цели броја x и y!

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int x,y;
    cout << " Vnesi go prviot broj: "<<endl;
    cin >> x ;
    cout << " Vnesi go vtoriot broj: "<<endl;
    cin >> y;
    while (x!=y)
    {
            if (x>y)
                x=x-y;
        else
                y=y-x;
    }
    cout<<"NZD e: "<<x<< endl;
    system ("PAUSE");
    return 0;
}
```

Објаснување: Алгоритамот на Евклид.

Овој алгоритам е заснован на својството дека НЗД на два броја е еднаков со НЗД на помалиот од двата броја и нивната разлика. На пример:

НЗД(18,12) = НЗД (12,6) = НЗД(6,6) = 6

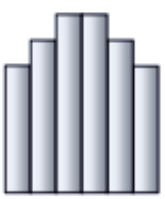
Постапката се повторува сè додека не се добијат два еднакви броја, и тој број претставува НЗД (`while (x!=y)`).

Поголемиот број се заменува со разликата на поголемиот со помалиот број:

Поминување 1:	x=18, y=12	18>12 → x=18-12=6, y=12
Поминување 2:	x=6, y=12	6<12 → x=6, y=12-6=6
Поминување 3:	x=6, y=6	6=6 → NZD=6

4.11 Задачи за талентираните ученици:

4.11.1 Линеарна структура

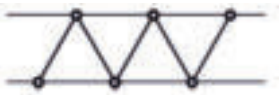


1. За училишна приредба Ана и Марио треба да направат замок од 6 штици поредени една до друга како на сликата. Првата штица е со должина k, втората е за 20 cm подолга од првата, третата е за 20 cm подолга од втората, четвртата има иста должина со третата, петтата има иста должина со втората и шестата има иста должина со првата штица. Напиши програма со која за дадена должина на првата штица се пресметува вкупната должина на сите штици.

   Пример:

   Влез: 50 Излез: 420

2. Учениците од I година на излет треба да направат висечки мост. Мостот се состои од две паралелни јажиња кои се врзани за крајбрежјата. Третото јаже треба со јазли да се прицврсти за првите две јажиња во цик-цак така да формира одреден број на рамнокраки триаголници. Напиши програма со која ќе се пресмета колку вкупно јазли треба да се направат ако треба да се формираат t триаголници!



Пример:

Влез: 4 Излез: 6

3. Во такси во исто време влегле тројца патници. Кога првиот патник излегол, таксиметарот покажувал C1 денари, кога излегол вториот патник, цената била C2 денари и кога излегол третиот патник цената била C3 денари. Колку треба да плати секој од патниците ако секој плаќа пропорционално, т.е. кога првиот патник ќе излезе, тој треба да плати само третина од C1 денари за првиот дел на патот?

Пример:

Влез: 30 50 100 Излез: 10 20 70

4. Банкомат издава банкноти од 1000, 500, 100, 50 и 10 денари. Напиши програма со која ќе се пресмета колку вкупно банкноти банкоматот ќе исплати за n денари, ако се исплатува по принцип на најмалку исплатени банкноти.

Пример:

Влез: 3780 Излез: 10

4.11.2 Разгранети структури

1. Горан сака во една соба да постави маса за билијард. Собата има форма на квадрат со страна со должина d, а масата мора да има правоаголна форма и мора да има одредени димензии c1 и c2. Од масата до ѕидовите мора да има одредено растојание. Напиши програма со која ќе се одреди дали масата може да ја собере во собата. Во програмата се внесуваат следните податоци: димензија на собата d, димензии на масата за билијард c1 и c2, најмало растојание на масата од ѕидовите m.

Пример: Влез: d = 4.5 c1 = 2.1 c2 = 2.6 m = 1 Излез: ne moze

Влез: d = 6.5 c1 = 2.1 c2 = 2.6 m = 0.6 Излез: moze

2. Марија учи деливост на броеви. Најлесно ѝ оди деливост со бројот 5, затоа што е доволно да ја види последната цифра на некој број за да дознае дали тој е делив со 5. Јана ѝ помага на Марија така што ѝ задава број x, а Марија мора да пронајде најмал број делив со 5 кој е помал од x и најголем број делив со 5 кој е поголем од x.

Напиши програма која ќе ѝ помогне ма Јана да ги провери одговорите на Марија!

Примери:

Влез: 12 Излез: 10 15

Влез: 27 Излез: 25 30

Влез: 15 Излез: 10 20

3. Матеј треба да оди во САД на натпревар по информатика. Тој дознал дека Американците поинаку го прикажуваат времето, како што се гледа во следнава табела:

македонско време	американско време	македонско време	американско време
00:00	12 AM	12:00	12 PM
01:00	1 AM	13:00	1 PM
02:00	2 AM	14:00	2 PM
03:00	3 AM	15:00	3 PM
04:00	4 AM	16:00	4 PM
05:00	5 AM	17:00	5 PM
06:00	6 AM	18:00	6 PM
07:00	7 AM	19:00	7 PM
08:00	8 AM	20:00	8 PM
09:00	9 AM	21:00	9 PM
10:00	10 AM	22:00	10 PM
11:00	11 AM	23:00	11 PM

Напиши програма која на Матеј ќе му помогне да претвора македонско време во американско.

Пример:

Влез	Влез
22	8
Излез	Излез
10 PM	8 AM

4.11.3 Структури со повторување

1. Маја треба да направи цветови од хартија. Секој цвет треба да биде со различна боја и ливчињата на секој цвет треба да бидат со иста боја. Од еден лист хартија можат да се исечат 4 ливчиња за цветови. Напиши програма со која за даден број на цветови и за дадени броеви на ливчињата на секој цвет се определува вкупен број на потребни листови хартија.

   Влезни податоци: Првата линија содржи ненегативен цел број n кој претставува број на цветови кои треба да се направат, во секоја од следниве n линии се наоѓа ненегативен цел број кој претставува број на ливчиња во секој од цветовите.

   Излези податоци: Вкупен број на листови потребни за правење на цветови.

   Пример.

Влез:	Излез:
5	9
5	
3	
9	
4	
6	

2. Во текот на учебната година Јана има работено n тестови по математика и ја интересира дали ќе има петка за крајот на годината. Јана ги знае поените по сите n тестови и знае колку вкупно поени се потребни за петка. Напиши програма со која Јана ќе пресмета дали ќе добие петка!

   Преку тастатурата се внесуваат следниве податоци:

   - број на тестови (n)
   - поени од сите тестови
   - број на потребните поени за петка

   Пример:

Влез	Влез
3	3
8	10
9	9
6	9
27	8
Излез	Излез
не	да

3. Напиши програма со која се одредува палиндром на даден природен број. Палиндром е број кај кој редоследот на цифри е обратен, на пр. на бројот 123 палиндром е бројот 321.

   Пример:

   Влез: 35472 Излез: 27453

4. Горјан ги послужил другарите со чоколадо во форма на правоаголник чии димензии се дадени со природни броеви a и b. Тој им кажал на другарите да кршат само квадрати. Другарите еден по еден откршувале квадрати, но секој земал најголем можен квадрат. Напиши програма со која се одредува колку другари се послужиле со чоколадото!

   Примери:

   Влез: a=12 b=7 Излез: 6

   (Вкупно 6 другари се послужиле со чоколадо, тие по ред земале квадрати со страните 7, 5, 2, 2, 1, 1.)

   Влез: a=7 b=7 Излез: 1

5. Во една гајба се наоѓаат m килограми јаболко и n килограми круши (m и n се природни броеви). Напиши програма која за дадени вредности на m и n ќе го прикаже најмалиот можен број на гајби потребни за препакување на јаболките и крушите така што гајбите ќе бидат со иста маса и јаболките и крушите треба да бидат одвоени.

   Пример:

   Влез: m=16 n=12 Излез: 7

ПРОГРАМИ ЗА ТАБЕЛАРНО ПРЕСМЕТУВАЊЕ

Клучни зборови

- Функција
- Аргумент
- Апсолутна адреса
- Релативна адреса
- Графикон
- Оска
- Легенда
- Извештај
- Пивот табела
- Филтер
- Критериум
- Сортирање
- Филтрирање
- Ниво
- База на податоци
- Форма за внес
- Лозинка

Од Office 2003 во Office 2007 и назад: http://office.microsoft.com/asstvid.aspx?&type=flash&assetid=XT010149329&vwidth=1044&vheight=788

5.1 Програми за табеларно пресметување

Важно!

Кога е можно содржините за Excel и за Calc ќе ги пишуваме заедно, при што наредбите и опциите ќе ги одделиме со знакот „/“. Ако има многу разлики, содржините ќе ги обработиме во посебни поднаслови.

Датотека креирана во MS Excel или Calc се нарекува *работна тетратка* или *работна книга* и се состои од повеќе *работни листови*. Секој работен лист е поделен на *редови* и *колони*, во пресек на секој ред и секоја колона се наоѓа *ќелија*. *Адресата на ќелија* е составена од ознака на колоната (буква) и ознака на редот (број) во кои таа се наоѓа.

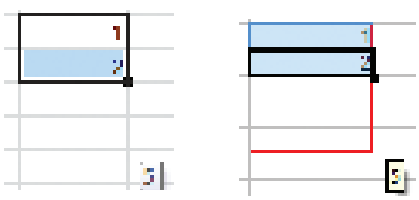
Со кликнување на ќелијата таа станува *активна* и може да се едитира (внесува, менува или уредува) нејзината содржина. Содржина на ќелија може да биде податок или формула, односно функција. Податоците можат да бидат текст, број, датум, време и друго.

При внесувањето на податоци, текстот автоматски се израмнува од десната страна, а броевите од левата страна. Со нумерички податоци (броеви) можат да се извршуваат аритметичките операции. Кога се внесува нумерички податок во ќелија е дозволено внесување само на цифри, децималната точка и предзнакот минус (-). Изгледот на ќелијата и на нејзината содржина најлесно се уредува во прозорецот Format cells/ Форматирај ќелии.

Во апликациите за табеларни пресметувања *формулите* имаат голема важност. Формулите можат да бидат кориснички дефинирани или вградени во форма на *функција*.

Секоја функција има *име* и *аргументи* кои се пишуваат во загради по името. Во MS Excel некои од најчесто користените функции (SUM, AVERAGE, MIN, MAX и COUNT) можат да се внесат преку листата која се добива со кликнување на стрелка до копчето *AutoSum* по што се отвора прозорец од кој се избираат аргументите на функцијата. Во Calc преку копчето *Сума* се внесува функцијата SUMA.

При внесување на податоци може да се користи *автоматско пополнување на ќелии*. Тоа најлесно се прави со повлекување на рачката за автоматско пополнување. На ист начин може да се прошири формула или функција од една ќелија во други ќелии.



Сл. 5. 1 Автоматско пополнување на ќелии во MS Excel (лево) и во Calc (десно)

Податоците од табелите можат да се претстави графички со различни типови на *графикони*. Најчесто користените графикони се: линискиот графикон, графиконот со столпчиња и графиконот пита. Во MS Excel графиконот во работниот лист се вметнува со кликнување на некое од копчињата од групата *Charts*, на картичката *Insert*. Во Calc графиконот се вметнува со кликнување на копчето *Графикон* на стандардната лента.

Внесување на формули

Основните правила за пишување и за примена на формули и функции се:

- Секое запишување на формула или функција започнува со знакот еднакво (=). Секој внес кој започнува со знакот еднакво (=) на програмата и кажува дека во ќелијата се наоѓа формула и дека е потребно да се извршат одредени пресметувања. Инструкциите за тоа кои пресметки треба да се извршат, и врз кои податоци, ги дава токму формулата.
- Операциите кои се користат во формула или функција се извршуваат со почитување на вообичаен математички редослед (приоритет).
- Доколку формулата или функцијата содржи адреси на ќелии, тие можат да се внесат преку тастатурата или со означување на саканите ќелии со помош на глувчето.
- Откако ќе се напише формула или функција се притиска копчето *Enter*. Тогаш во ќелијата се прикажува резултатот од формулата или функцијата, а нејзиниот запис може да се види во лентата за формули (сл. 5.2). На пр. Формулата =A1+B1, вметната во ќелијата C1, ги собира вредностите во ќелиите A1 и B1 и резултатот го прикажува во ќелијата C1.

Сл. 5. 2 Формула во MS Excel (лево) и формула во Calc (десно)

Оваа формула може да се внесе и на следниов начин: во ќелијата C1 се пишува знакот за еднаквост (=), потоа се кликнува на ќелијата A1, се пишува операторот за собирање, па се кликнува на ќелијата A2.

Уредување на формули

Постоечка формула може да се уреди:

- со кликнување во делот за содржина на лентата за формули кога е активна ќелија во која формулата е внесена,
- со двоклик на ќелија во која е внесена формула, по што се поминува во начин на едитирање и во ќелијата се гледа синтакса на формулата наместо нејзиниот резултат (кратенка F2).

Задачи за повторување:

1. Во твојата паралелка се бира претседател на класот. Кандидатите за претсетател се Борис, Елена и Јанко. За Борис гласале 7 ученици, за Елена 5 ученици и за Јанко 14 ученици. Направи табела и пресметај:
   - Колку вкупно ученици гласале? _____
   - Кој добил најмалку гласови? _____
   - Кој добил најмногу гласови? _____
   - На табелата постави рабови и бои на ќелии по избор;
   - Вметни еден ред пред табелата и во него напиши наслов „Избор за претседател“ со задебелени букви;

- Нацртај графикон пита и уреди го по желба;
- Зачувај ја работната книга со име Pretsedatel.

2. На една улица е вршено истражување колку возила и од кој вид поминуваат за еден час. Добиени се следниве резултати:

	A	B	C
1	Вид на возило	Број на возила за 1 час	Број на возила за 1 ден
2	Автомобил	29	
3	Комби	11	
4	Мотор	18	
5	Велосипед	8	
6	Автобус	3	
7	ВКУПНО		

- Според податоците креирај графикон со столпчиња;
- Во ќелијата B7 напиши функција со која ќе се пресмета вкупниот број на возила кои поминуваат за еден час на улицата;
  - Како гласи функцијата? _____
  - Кој е резултатот на функцијата? _____
- Во колоната C напиши формула со која за секој вид возило ќе пресметаш колку возила поминуваат за еден ден;
  - Како гласи формулата за автомобилите? _____
  - Кој е резултатот за автобусите? _____
- Работната книга зачувај ја со име Soobrakaj.

3. Отвори нова работната книга и креирај табела како на сликата:

	A	B	C	D	E	F	G
1	Планетите на Сончевиот систем						
2							
3	Планета	Дијаметар	Оддалеченост од Сонцето	Бројот на природните сателити	Должина на денот	Време на обиколување околу Сонцето	Температура на површината
4	Меркур						
5	Венера						
6	Земја						
7	Марс						
8	Јупитер						
9	Сатурн						
10	Уран						
11	Нептун						
12	Плутон						

- Истражи преку Интернет и внеси ги соодветните податоци во табелата. Внимавај мерните единици да бидат исти за сите планети.
- Пресметај и одговори на следниве прашања:
  - Која планета има најмногу природни сателити? _____
  - Која планета е најтопла? _____
  - Колку пати Земјата е поголема од Марс? _____

· Колку пати Јупитер е поголем од Меркур? _____

· Кои планети се со приближно иста големина? _____

- Вметни линиски графикон за температурите на површината на планетите;
- Спореди ги температурите на останатите планети со температурата на Земјата и дополни ги следниве реченици:

· Повисока температура од температурата на Земјата имаат планетите _____

· Пониска температура од температурата на Земјата имаат планетите _____

4. Креирај и уреди табела како на сликата:

	A	B	C	D	E	F	G
1	Боја на светлината	Бранова должина во nm		Фреквенција			
2		од	до	од	до		
3	Виолетова	390	450	7,7	6,7		
4	Сина	450	500	6,7	6		
5	Зелена	500	570	6	5,3		
6	Жолта	570	600	5,3	5		
7	Портокалова	600	630	5	4,8		
8	Црвена	630	780	4,8	3,8		
9							
10	Просечна бранова должина:						
11	Просечна фреквенција:						

- Спој ги ќелиите F1 и F2 и напиши „Просечна бранова должина“;
- Пресметај вредности во колоната F;
- Спој ги ќелиите G1 и G2 и напиши „Просечна фреквенција“;
- Пресметај вредности во колоната G;
- Пресметај вредности во ќелиите D10 и D11;
- Податоците од колоните F и G прикажи ги со линиски графикон. Графиконот е без наслов, легендата постави ја на дното;
- Работната книга зачувај ја со име Svetlinata.

5.2 Напредно користење на формули и функции

Програмите за табеларни пресметки овозможуваат пресметување со сложени математички формули и нудат голем број на функции распоредени во категории (математички, статистички, финансиски итн.).

Формула е множество од инструкции кои корисникот ги внесува во ќелија, и кои на програмата и кажуваат со кои податоци и на кој начин со нив да манипулира. Она што се гледа во ќелијата во која формулата е внесена, како последица од нејзиното делување, е резултат на формулата.

Функциите претставуваат вградени алгоритми кои изведуваат различни операции врз дадените податоци. Можат да се разгледуваат и како посебни формули кои се користат самостојно или како делови на формули кои се конструираат.

5.2.1 Синтакса на функции

Секоја функција започнува со знакот еднакво (=) по кој доаѓа името на функција. По името, во загради, се запишуваат аргументите на функција.

Значи синтаксата на секоја функција има форма:

=ImeNaFunkcija (lista na argumenti)

Ако во функција има повеќе аргументи тие се раздвојуваат со знакот точка и запирка (;).

Аргументи на функција се вредности врз кои функција извршува одредени операции. Тоа можат да бидат:

- адреси на ќелии (на пр. A5),
- опсези на адреси (на пр. A1:A5),
- константи (на пр. 1000),
- текст (на пр. “Битола”),
- логички вредности (TRUE, FALSE) или
- други функции.

Зад. 5. 1. Одреди ги името и аргументите на дадените функции:

Функција	Име	Аргументи
=ABS(A2)		
=SUM(A1:A5)		
=SUM(100;C1:C10)		
= MIN(A1:A7;B1)		
= MAX(10;5;A5)		
=AVERAGE(10;A1:A5)		

5.2.2 Оператори

За да можат да се изведуваат различни операции и пресметки во формули, програмите за табеларни пресметувања нудат низа оператори кои можат да се користат при конструкција на формулите. Операторите се поделени во четири категории:

- *аритметички оператори* – за изведување на аритметички операции
- *оператори за споредување* – за споредување на вредности, а изрази во кои тие се користат како резултат имаат логички вредности TRUE (точно) и FALSE (неточно)
- *текстуален оператор за спојување*
- *оператори за адреси.*

Аритметички оператори

+ (плус)	оператор за собирање
- (минус)	оператор за одземање
* (ѕвездичка)	оператор за множење
/ (коса црта)	оператор за делење
% (процент)	оператор за процент
^ (карет)	оператор за експонент (на пр. 3^2 претставува три на квадрат)

Оператори за споредување

=	еднакво
<>	не е еднакво
<	помало
>	поголемо
<=	помало или еднакво
>=	поголемо или еднакво

Текстуален оператор за спојување

& (ampersand) е оператор кој овозможува спојување на текст од две или повеќе ќелии. На пр. ако во ќелијата A1 е текст "Марко", а во ќелијата B1 "Марковски", формулата =A1&" "&B1 ќе даде резултат "Марко Марковски". Забележуваш дека во формулата е уфрлено и едно празно место за да се раздвои името од презимето.

Забелешка:

Секој текст треба да се стави во наводници, па така и знакот за празно место. Адресите и вредностите не треба да бидат во наводници, во спротивно програмата ќе ги протолкува како текст.

Оператори за адреси

: (две точки) претставува оператор за опсег, кој на програмата ѝ кажува дека сè што е внесено во ќелии помеѓу две адреси помеѓу кои се наоѓа знакот две точки, вклучувајќи ги и нив, претставува една адреса.

Пр. 5. 1. Изразот A1:B2 претставува една адреса која содржи четири ќелии: A1, A2, B1 и B2. Ваков тип на адреси се најчесто користени аргументи во формули.

; (точка и запирка) претставува оператор за унија, кој спојува две адреси во една. Овој оператор се користи исклучително во функции.

(празно место, space) е оператор на пресек. Овој оператор многу ретко се користи и многу корисници не знаат дека тој постои.

Хиерархија на операторите

Програмите за табеларно пресметување, слично како во математиката, имаат зададена хиерархија на редоследот по кој ги користи операторите за пресметување по формулите. Редоследот е следниов:

1. оператор за опсег (:),
2. оператор на пресек ( ),
3. оператор за унија (;),
4. оператор за негативни вредности (-),
5. оператор за процент (%),
6. оператор за степенување (^),
7. оператори за множење и делење (* и /),
8. оператори за собирање и одземање (+ и –) ,
9. оператор за спојување на текст (&),
10. оператори за споредување (=, <>, <, >, <=, >=).

Доколку во формулата се наоѓаат повеќе оператори од ист ранг, програмата ќе ги извршува од лево кон десно. Овој редослед може да се смени со употреба на загради,

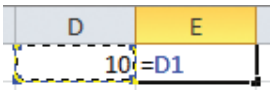
при што најпрво се пресметува она што се наоѓа во заграда. Ако има повеќе загради, се решаваат од внатре кон надвор.

Пр. 5. 2. Резултатот на формулата =5+3*2 е 11, додека резултатот на формулата =(5+3)*2 е 16.

5.2.3 Адресирање на ќелии

Адресите на ќелии кои се користат во формули и функции на програмата ѝ кажуваат од каде да зема вредности за пресметување.

Адресите можат да се внесат така што ќе се запишат во формула, но и така што ќе се означат ќелии или опсези на ќелии со помош на левото копче од глувчето во режимот за едитирање. Ваков начин на внесување се нарекува *адресирање со селекција*.



Сл. 5. 3 Адресирање со селекција

Адресирање на ќелија

Адреса на поединечна ќелија се состои од ознака за колоната (буква) и ознака за редот (број) во чиј пресек ќелијата се наоѓа.

Пр. 5. 3. A1 е адреса на ќелијата која се наоѓа во пресек на првата колона и првиот ред.

Адресирање на опсег

Адресирање на опсег се однесува на непрекинат правоаголен опсег на повеќе ќелии кој се протега преку повеќе колони и редови. При ваквото адресирање со знакот две точки (:) се спојуваат адреси на горната лева и на долната десна ќелија. На пр. адресата B2:C3 означува опсег од четири ќелии: B2, B3, C2 и C3.

Адресирање на ќелии од друг работен лист

Во формулите и функциите можат да се користат и адреси на ќелии и опсези кои се наоѓаат во друг работен лист. Во тој случај, во адресата треба да се стави името на тој работен лист како префикс, по кој се пишува знакот извичник (!) по кој доаѓа ознака за ќелија или ознака за опсег на ќелии.

Пр. 5. 4. Sheet1!A1 е адреса за ќелијата A1 во работниот лист Sheet1.

Пр. 5. 5. Uspeh!B5:B10 е адреса на ќелиите B5:B10 во работниот лист Uspeh.

Доколку работниот лист има име со повеќе зборови, на пр. Прва година, тогаш името се става помеѓу два апострофа (‘).

Пр. 5. 6. ‘Прва година’!A2:B3 е адреса за опсегот A2:B3 во работниот лист Прва година.

Совет:

Пишувањето на ваквите адреси е непрактично. Подобро е ќелиите и опсезите од други работни листови да се означат со глувче. Програмата автоматски ќе ги внесе во формула или функција.

Чекор по чекор:

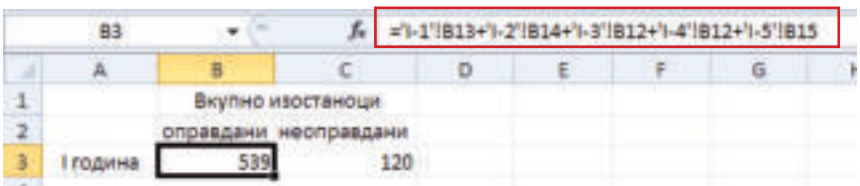
Пресметај го вкупниот број на изостаноците во првата година!

1. Отвори нова работна книга и додај уште три работни листа;
2. Именувај ги работните листови: I-1, I-2, I-3, I-4, I-5 и I година;

3. На листовите од I-1 до I-5 креирај табели како на сликата десно (за секоја паралелка внеси други податоци);
4. Во работниот лист I година, креирај табела како на сликата десно (сл. 5. 5);
5. Во ќелиите А3 и В3 треба да го пресметаш вкупниот број на оправданите и неоправданите изостаноци за сите паралелки од прва година:
   - во работниот лист I година, во ќелијата А3 напиши еднакво (=),
   - кликни на името на работниот лист I-1 и во него кликни на ќелија во која е прикажан бројот на вкупно оправданите изостаноци,
   - напиши плус (+),
   - претходните два чекори повтори ги на работните листови I-2, I-3, I-4 и I-5, со тоа што откако ќе кликнеш во ќелија во последниот работен лист нема да го напишеш знакот плус,
   - притисни го копчето *Enter*;
6. Постапката повтори ја и за ќелијата В3.

	A	B	C
1	Број на	Изостаноци	
2	ученик	оправд.	неопр.
3	1	55	12
4	2	12	1
5	3	5	0
6	4	0	0
7	5	15	3
8	6	25	0
9	7	11	1
10	8	32	2
11	9	0	2
12	10	15	0
13	Вкупно:	228	21

Сл. 5. 4 Табела за една паралелка



Сл. 5. 5 Табела за I година

Важно!

Откако ќе кликнеш на соодветна ќелија во последниот работен лист НЕМОЈ да кликнуваш на името на работниот лист во кој ја внесуваш формулата.

За љубопитните:

Адресирање на 3Д опсег

Може да се адресираат ќелии на повеќе работни листови така што помеѓу имињата на работните листови се става операторот за опсег (:), по тоа знакот извичник (!) и адреса на ќелија/ќелии. На пр. Sheet1:Sheet10!A1 е адреса за ќелијата А1 на сите работни листови кои се наоѓаат помеѓу листовите Sheet1 и Sheet10, вклучувајќи ги и нив самите. Sheet1:Sheet10!A1:C3 е адреса за опсегот А1:С3 на сите работни листови кои се наоѓаат помеѓу листовите Sheet1 и Sheet10, вклучувајќи ги и нив самите.

Собери ги вредностите на сите ќелии А1 во работните листови од Sheet1 до Sheet5! Тоа ќе го направиш со формулата: =SUM(Sheet1:Sheet5!A1).

Доколку името некој од работните листови кој е прв или последен во опсегот се состои од два или повеќе зборови, тогаш делот од адресата кој се однесува на имињата на работните листови треба да се стави помеѓу два апострофи (‘). На пр. ‘Прва година:Четврта година’!A1:B10

Адресирање на ќелии од друга работна книга

Може да се адресира и ќелија од друга работна книга. Тогаш пред името на работниот лист во кој ќелијата се наоѓа, се става името на работната книга во средни загради. На пр. [Book2]Sheet1!A1 е адреса за ќелијата A1 во листот Sheet1 од работната книга Book2.

5.2.4 Релативно и апсолутно адресирање на ќелии

Во формулите и функциите се користат два вида на адресирање на ќелии од кои се земаат податоци:

- *релативно адресирање* – адресирање во однос на активна ќелија
- *апсолутно адресирање* – адресирање независно од активна ќелија

Релативно адресирање

Програмите за табеларни пресметки стандардно користат релативно адресирање – адреса на ќелија се формира од ознака на колона и од ознака на ред, на пр. E3. Кога се копира формула која содржи релативни адреси програмата нема да креира нејзина изворна копија. Адресите на ќелиите ќе се сменат и формулата ќе се прилагоди кон новата колона и/или ред.

Пр. 5. 7. Ако формулата =A1+B1 внесена во ќелијата C1 се ископира во ќелијата E5 таа ќе гласи =C5+D5. На овој начин секогаш се собираат вредностите во двете ќелии лево од ќелијата во која се наоѓа формулата.

Благодарејќи на ваквиот начин на адресирање можат да се копираат формули со постапка за автоматско пренесување на формули. Програмата ќе ја прилагоди формулата според ќелијата во која таа е копирана.

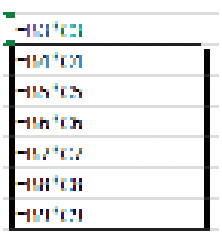
Потсети се!

За да се прошири некоја формула или функција, односно да се ископира во други ќелии, тоа најбрзо се прави така што се повлекува рачката за автоматско пополнување од ќелијата во која таа се наоѓа кон ќелиите во кои истата треба да се прошири.

Пр. 5. 8. Ако формулата =A1+A2 се копира кон десно, адресите во новите формули ќе се сменат како што е прикажано на сликата долу.

Сл. 5. 6 Проширување на формула по колони

Пр. 5. 9. Ако формулата =B3*C3 се копира надолу, адресите во новите формули ќе се сменат и прилагодат кон новите локации како што е прикажано на сликата десно.



Сл. 5. 7 Проширување на формула по редови

Совет:

За наеднаш да ги видиш сите формули внесени во работен лист притисни Ctrl +`. Со истата кратенка повторно ќе се прикажат резултатите на формулите.

Чекор по чекор:

Во табела внеси податоци како на следнава слика. Во колоната C пресметај вредности на производите:

1. Во ќелијата C2 напиши формула =B2*C2;
2. Формулата прошири ја на ќелиите D3 до D6;

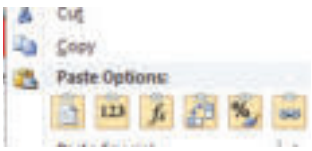
	A	B	C	D
1	Производ	Количина	Цена	Вредност
2	Сок	5	120,00 ден.	
3	Помфрит	4	60,00 ден.	
4	Сендвичи	6	80,00 ден.	
5	Сладолед	7	65,00 ден.	

3. Погледни ја формата на сите формули во ќелиите од D2 до D6. Забележуваш дека адресите на ќелиите од кои се земаат вредности се менуваат согласно ќелија во која формулата се наоѓа.

D
Вредност
=B2*C2
=B3*C3
=B4*C4
=B5*C5

Забелешка:

Понекогаш е потребно да се ископира формула, а понекогаш само нејзината вредност. По наредбата Copy, од наредбата Paste Options и се одбира опцијата Formulas (за формула) или Values (за вредност).



Апсолутно адресирање

За подобро да се разбере апсолутното адресирање и потреба од истото, следниот пример ќе се обидеме да го решиме со релативно адресирање.

Пр. 5. 10. Нека во колоната D се дадени вредности изразени во денари; во колоната E истите вредности сакаме да ги изразиме во евра. Курсот на евро е даден во ќелијата G1.

D	E	F	G
Вредност	Вредност во евра	Курсна евро:	61,5
600,00 ден.	9,76 C		
240,00 ден.	#DIV/0!		
480,00 ден.	#DIV/0!		
453,00 ден.	#DIV/0!		

Во ќелијата E2 внеси формула за претворање на вредност во денари во вредност во евра (=D2/G1) и прошири ја до ќелијата E6.

Забележуваш дека пресметувањето не е добро. При проширувањето адресата G1 е заменета со адресите G2, G3, G4, G5 и G6 (слика десно). Во овој случај тоа не ни одговара, па мораме на програмата да ѝ „кажеме“ дека сакаме адресата G1 да остане непроменета при проширувањето на формулата. За таа цел ќе користиме *апсолутно адресирање*.

I
Вредност во евра
=D2/G1
=D3/G2
=D4/G3
=D5/G4

Кога се користи апсолутно адресирање, адресите нема да се сменат ако формулата се копира или пренесе во друга ќелија. Ова значи дека формулата секогаш се однесува на иста ќелија без разлика каде таа ќе се ископира или ќе се премести.

Адресите при апсолутното адресирање се означуваат така што пред ознаките на колоната и на редот се става знакот долар ($). На пр. адресата $A$1 секогаш се однесува на ќелијата A1 без разлика каде формула која ја содржи оваа адреса ќе биде ископирана или пренесена. Исто правило се применува и кога се адресира опсег на ќелии, на пр. $A$1:$B$2.

Чекор по чекор:

Во табелата од претходниот пример внеси исправна формула за претворање на вредности изразени со денари во вредности изразени во евра:

1. Во ќелијата E2 внеси формула =D2/$G$1) и прошири ја до ќелијата E6.

D	E	F	G
	Вредност		
Вредност	во евра	Курс на евро:	61,5
600,00	9,75		
240,00	3,90		
480,00	7,80		
455,00	7,40		

Мешовито адресирање

Мешовито адресирање е адресирање во кое се комбинира една релативна и една апсолутна ознака за ред или колона.

Пр. 5. 11. Во адресата $B3 е дадено апсолутно адресирање на колона и релативно адресирање на ред. Ако формулата која содржи ваква адреса се копира во друга ќелија, во адресата ќе се смени само адресата на редот.

Пр. 5. 12. Во адресата C$2 е дадено релативно адресирање на колона и апсолутно адресирање на ред. Ако формулата која содржи ваква адреса се копира во друга ќелија, во адресата ќе се смени само адресата на колона.

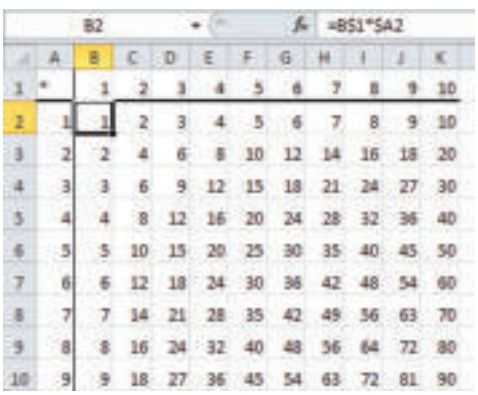
Совет:

Адресите во формули побрзо ќе се внесат со означување на соодветните ќелии. Секогаш може да се смени начинот на адресирање од релативно во апсолутно или мешовито со притискање на копчето F4 во MS Excel, односно Shift+F4 во Calc. На пр. формулата =B3 ќе добие изглед =$B$3, потоа =B$3 и на крај =$B3.

Вежба:

Креирај таблица на множење до 10!

1. Во ќелијата B2 внеси формула =B$1*$A2;
2. Формулата прошири ја по редови и колони!



	A	B	C	D	E	F	G	H	I	J	K
1	*	1	2	3	4	5	6	7	8	9	10
2	1	1	2	3	4	5	6	7	8	9	10
3	2	2	4	6	8	10	12	14	16	18	20
4	3	3	6	9	12	15	18	21	24	27	30
5	4	4	8	12	16	20	24	28	32	36	40
6	5	5	10	15	20	25	30	35	40	45	50
7	6	6	12	18	24	30	36	42	48	54	60
8	7	7	14	21	28	35	42	49	56	63	70
9	8	8	16	24	32	40	48	56	64	72	80
10	9	9	18	27	36	45	54	63	72	81	90

Во првиот множител (B$1) се користи апсолутна адреса на ред и релативна адреса на колона, па со проширување надолу секогаш е адреса на ќелија во првиот ред, додека со проширување кон десно се менува знакот на колона за ќелијата.

Кај вториот множител ($A2) е обратна ситуација. Со проширување надолу се менува ознака на редот, додека со проширување кон десно не се менува ознаката на колоната, односно секогаш е адреса на ќелија од првата колона (A).

Резиме

Секоја функција започнува со знакот еднакво (=) по кој доаѓа името на функцијата. По името, во загради, се запишуваат аргументите на функцијата.

Синтаксата на секоја функција има форма:

=ImeNaFunkcija (lista na argumenti)

Ако во функцијата има повеќе аргументи тие се разделуваат со знакот точка и запирка (;).

Операторите се поделени во четири категории: аритметички оператори, оператори за споредување, текстуален оператор за спојување и оператори за адреси.

Релативно адресирање се заснова на позиција на ќелија во однос на ќелија во која е содржана формула. Адресата на релативно адресирана ќелија се менува со копирање или поместување на формула.

Апсолутното адресирање се однесува на точно одредена ќелија. Пред адресата на апсолутно адресираната ќелија се наоѓа знакот за долар ($) и адресата не се менува со копирање или поместување на формулата.

Вештини што треба да ги усовршиш:

Да внесеш формула во работен лист.

Да ги препознаваш функцијата во табела и нејзините аргументи.

Да вметнеш функција и да одредиш аргументи.

Да адресираш ќелии.

Да препознаваш и да користиш апсолутно и релативно адресирање на ќелии.

Прашања:

1. Што е формула, а што е функција?
2. Со кој знак започнува пишувањето на формула или функција?
3. Која е општа форма на функција?
4. Што е аргумент на функција?
5. Што може да претставува аргумент на функција?
6. Ако некоја функција има повеќе аргументи, како тие се раздвојуваат?
7. Кои се аргументи на функцијата = MIN(A2:A5;B7)?
8. На кои категории се поделени операторите?
9. Која е разлика помеѓу операторот „:“ и операторот „;“?
10. Како се адресира опсег на ќелии?
11. Како се адресираат ќелии од друг работен лист?
12. Напиши адреса на ќелиите од A2 до C3!
13. Напиши адреса на ќелиите од A2 до C3 на работниот лист Promet?
14. Што е релативно, а што е апсолутно адресирање?
15. Напиши апсолутна адреса на ќелијата B5!
16. Наведи пример за користење на апсолутно адресирање!

Задачи:

1. Ученици од првата година собираат електронски отпад. Секој ученик донел одреден број на стари дискети и компакт дискови. На посебни работни листови креирај табели за секоја паралелка во кои ќе го внесеш бројот да собраните дискети и компакт дискови за секој ученик! На пример:

	A	B	C
1	Име и презиме	Дискети	Компакт дискови
2	Марио Николов	10	00
3	Маја Андоновска	15	10
4	Гоце Петрески	12	115
5	Арбен Јусуфи	65	0
6		102	185

За секоја паралелка најди го вкупниот број на собраните дискети и компакт дискови!

На посебен работен лист најди го вкупниот број на собраните дискети и компакт дискови за сите паралелки од прва година!

2. Внеси ги следниве податоци во табела:

	A	B	C	D
1		Данок	18,00%	
2				
3				
4	Производ	Цена без данок	Износ на данок	Цена со данок
5	Монитор	€180,00		
6	Печатач	€200,00		
7	Скенер	€120,00		
8	Компјутер	€500,00		

- Во ќелијата C5 пресметај го износот на данок;
- Во ќелијата D5 пресметај ја вкупната цена;
- И двете формули прошири ги на ќелиите C6 и C7, односно D6 и D7!

3. Во табела внеси податоци како на сликата:

	A	B	C
1			
2			Сметка за струја
3			2.000,00 ден.
4			
5	Семејство	Број на членови	Струја
6	Маркоа	3	
7	Дитиќ	5	
8	Теовски	4	
9	Андов	4	
10	Арсовски	5	

- Во ќелијата B11 најди го вкупниот број на членови;
- Износот на сметката за струја, даден во ќелијата C3, подели го на сите семејства според бројот на членови (износот подели го со вкупниот број на членови и помножи го со бројот на членови на соодветното семејство)!

4. Внеси ги следниве податоци во табела:

	A	B	C	D	E	F	G
1	**Контролен тест по математика**						
2							
3		Зад. 1	Зад. 2	Зад. 3	Зад. 4	ВКУПНО	
4	Максимален број на поени	20	15	25	20	80	
5	**Име и презиме**	**Освоени поени по задача**				**Вкупно**	**Процент**
6	Мите Гоцев	18	10	22	20		
7	Мите Гоцев	20	15	25	20		
8	Селим Османи	20	10	15	18		
9	Маја Андова	10	15	10	12		
10	Марко Симиќ	12	8	15	8		
11	Маја Андова	20	15	25	20		

- во колоната F пресметај вкупно освоени поени за секој ученик;
- во колоната G пресметај процент на вкупно освоените поени за секој ученик (вкупно поени за ученикот поделен со вкупен максимален број на поени даден во ќелијата F4); ќелиите форматирај ги да прикажуваат вредност во проценти!

5.2.5 Некои посложени функции

Внесување на функции

Функција може да се внесе во ќелија на два начина.

Првиот начин е со директно запишување при што во ќелија се внесува знакот за еднаквост (=), потоа името на функцијата и аргументите според синтаксата на функцијата.

Вториот начин е преку дијалог прозорец за внесување на функции.

Внесување на функции во Ms Excel

Во Ms Excel дијалог прозорец за внесување на функции *Insert Function* се отвора со кликнување на копчето *Insert Function* ( ) од лентата со формули или од картичката *Formulas* (кратенка *Shift + F3*).

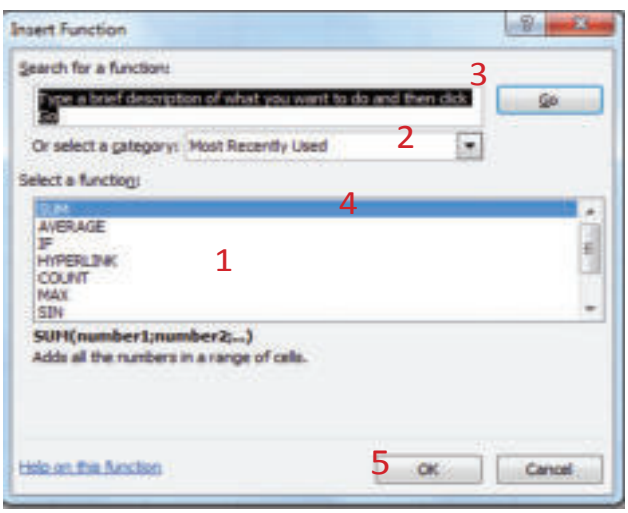
1 Во прозорецот *Insert Function* се добива список од сите расположиви функции;

2 Се прикажуваат најчесто користените функции, но доколку функција која е потребна не се наоѓа на списокот, се избира друга категорија од листата *Select a category*;

3 Ако се знае точното име на функцијата, името може да се напише во полето *Search for a function* по што се кликнува на копчето Go;

4 Откако ќе се најде бараната функцијата, се кликнува на нејзиното име;

5 Се кликнува на копчето *OK*



Сл. 5. 8 Избор на функција во MS Excel

По изборот на функција се отвора прозорецот *Function Arguments*

6 во кој се внесуваат аргументи на функцијата. Овој прозорец се разликува во зависност од избраната функција

7 Како помош околу аргументите може да послужи објаснувањето дадено во самиот прозорец.

8 Помош околу функцијата може да се добие со кликнување на линкот *Help on this function*;

9 На крај се кликнува на копчето *ОК* , по што резултатот на функцијата се појавува во соодветната ќелија.

Сл. 5. 9 Внесување на аргументи на функција во MS EXcel

Внесување на функции во Calc

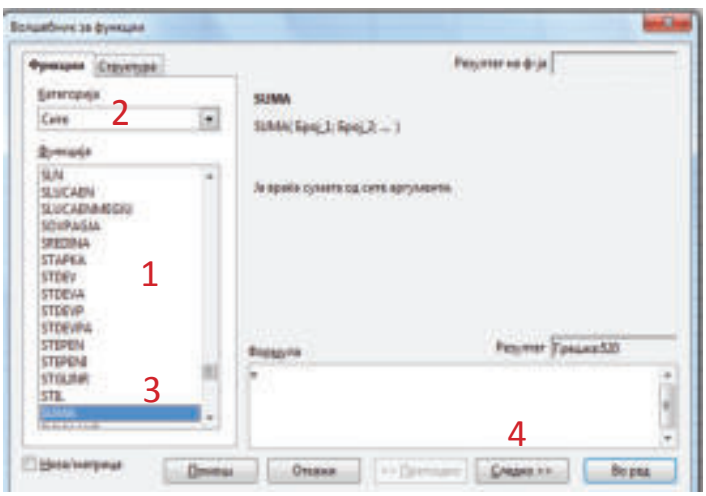
Во Calc дијалог прозорец за внесување на функции *Волшебник за функции* се отвора со кликнување на копчето *Волшебник за функции* ( ) од лентата со формули.

1 Во прозорецот *Волшебник на функции* се добива список од сите расположиви функции;

2 Се прикажуваат најчесто користените функции; доколку функција која е потребна не се наоѓа на списокот, се избира друга категорија од листата *Категорија*;

3 Откако ќе се најде бараната функција, се кликнува на нејзиното име;

4 Се кликнува на копчето *Следно*;



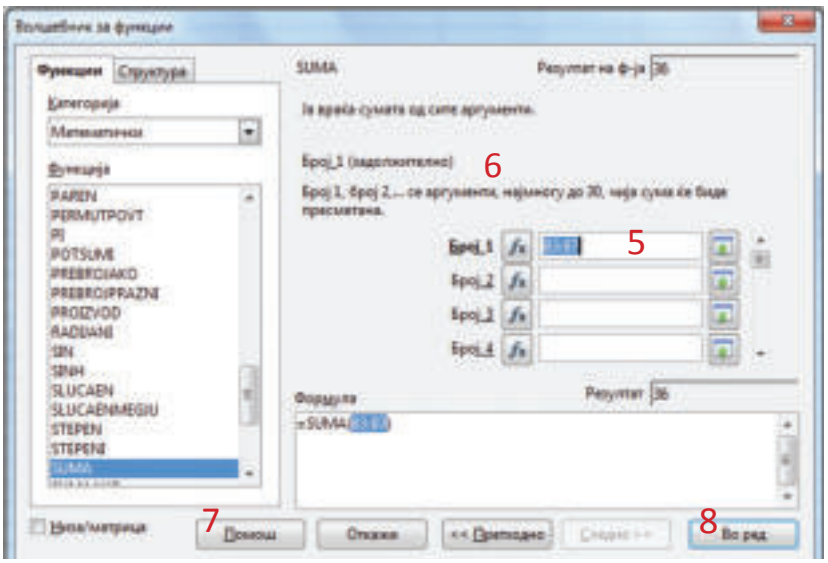
Сл. 5. 10 Избор на функција во Calc

5 Се отвора вториот дел од прозорецот *Волшебник за функции* во кој се внесуваат аргументи на функцијата. Овој прозорец се разликува во зависност од избраната функција;

6 Како помош околу аргументите може да послужи објаснувањето дадено во самиот прозорец;

7 Помош околу функцијата може да се добие со кликнување на копчето *Помош*;

8 На крај се кликнува на копчето *Во ред*, по што резултатот на функцијата се појавува во соодветната ќелија.



Сл. 5. 11 Внесување на аргументи на функција во Calc

Функцијата COUNTA/ПРЕБРОЈА

Функцијата COUNTA/ПРЕБРОЈА се користи за пребројување на ќелии од некој опсег кои не се празни, т.е. ќелии во кои се наоѓа некоја вредност. Синтаксата на функцијата е:

MS Excel: *=COUNTA(value1; value2; ...)*

Calc: *=ПРЕБРОЈА(вредност1; вредност2;...)*

Аргументи на оваа функција се адреси на ќелии (на пр. A1; E5;E8) или опсег на ќелии (на пр. C2:C9) кои се испитуваат и пребројуваат.

Пр. 5. 13. Да се одреди бројот на учениците кои се наоѓаат на списокот за полагање тест.

	A	B	C
1	Име и презиме	Поени	Оцена
2	Дарко Јовановски	35	2
3	Ана Мишева	100	5
4	Маја Митиќ	92	5
5	Марко Костов	64	4
6	Дина Ацевска	12	1
7	Јане Петров	/	/
8	Енис Бајрами	74	4
9	Миа Лазова	2	1
10	Вкупно ученици:		8

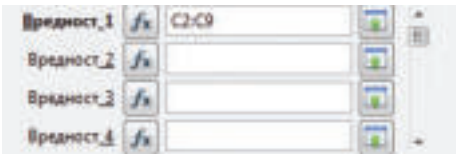
MS Excel:
=COUNTA(C2:C9)

Calc:
=ПРЕБРОЈА(C2:C9)

Сл. 5. 12 Пример за користење на функцијата COUNTA / ПРЕБРОЈА

Внесување на аргументи на функцијата COUNTA во MS Excel и ПРЕБРОЈА во Calc се дадени на следниве слики:

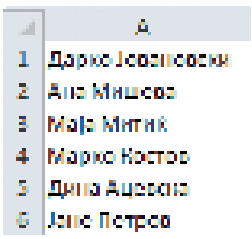
Сл. 5. 13 Внесување на аргументите на функцијата COUNTA



Сл. 5. 14 Внесување на аргументите на функцијата ПРЕБРОЈА

Чекор по чекор:

Во работен лист внеси ги следниве податоци:



Преброј колку ученици има на список!

1. Позиционирај се во ќелијата A7 (тука ќе се прикаже резултатот на функцијата);
2. На лентата за формули кликни на копчето *Insert Function/Волшебник за функции*;
3. Во прозорецот *Insert Function/Волшебник за функции* најди ја функцијата COUNTA/ПРЕБРОЈА и кликни на копчето *ОК/Следно*;
4. Позиционирај се во лентата *Value1/Вредност1* и означи ги ќелиите A1:A6;
5. Кликни на копчето *ОК/Во ред*;
6. Функцијата гласи: =COUNTA(A1:A6)/=ПРЕБРОЈА(A1:A6), резултатот на функцијата е 6.

Функцијата COUNT/ПРЕБРОЈ

Функцијата COUNT/ПРЕБРОЈ ги пребројува ќелиите од некој опсег во кои се наоѓа некоја нумеричка вредност. Синтакса на функцијата е:

MS Excel:	Calc:
=COUNT(value1; value2; ...)	*=ПРЕБРОЈ(вредност1; вредност2;...)*

Аргументите на оваа функција се адреси на ќелии (на пр. A1; E5;E8) или опсег на ќелии (на пр. C2:C9) кои се испитуваат и се пребројуваат оние ќелии во кои се наоѓа нумеричка вредност.

Пр. 5. 14. Да се одреди бројот на ученици кои го полагале тестот.

	A	B	C
1	Име и презиме	Поени	Оцена
2	Дарко Јовановски	35	2
3	Ана Мишева	100	5
4	Маја Митиќ	92	5
5	Марко Костов	64	4
6	Дина Ацевска	12	1
7	Јане Петров	/	/
8	Енис Бајрами	74	4
9	Миа Лазова	2	1
10		Вкупно ученици:	8
11		Вкупно полагале:	7

MS Excel:
=COUNT(C2:C9)

Calc:
=ПРЕБРОЈ(C2:C9)

Сл. 5. 15 Пример за користење на функцијата COUNT / ПРЕБРОЈ

Внесување на аргументите на функцијата COUNT во MS Excel и ПРЕБРОЈ во Calc се дадени на следниве слики:

Сл. 5. 16 Внесување на аргументи на функцијата COUNT

Сл. 5. 17 Внесување на аргументи на функцијата ПРЕБРОЈ

Чекор по чекор:

Во работен лист внеси ги следниве податоци:

	A	B
1	Дарко Јовановски	50,00 ден.
2	Ана Мишева	
3	Маја Митиќ	80,00 ден.
4	Марко Костов	150,00 ден.
5	Дина Ацевска	
6	Јане Петров	100,00 ден.

Преброј колку ученици донирале средства на хуманитарна акција:

1. Позиционирај се во ќелијата B7;
2. На лентата за формули кликни на копчето *Insert Function/Волшебник за функции*;
3. Во прозорецот *Insert Function/Волшебник за функции* најди ја функцијата COUNT/ ПРЕБРОЈ и кликни на копчето *ОК/Следно*;
4. Позиционирај се во лентата *Value1/Вредност1* и означи ги ќелиите B1:B6;
5. Кликни на копчето *ОК/Во ред*;
6. Функцијата гласи: =COUNT(B1:B6)/=ПРЕБРОЈ(B1:B6), резултатот на функцијата е 4.

Функцијата COUNTIF/ПРЕБРОЈАКО

Функцијата COUNTIF/ПРЕБРОЈАКО се користи за условно пребројување, т.е. ги пребројува ќелиите од опсегот чии вредности го задоволуваат дадениот критериум. Синтаксата на оваа функција е:

MS Excel: *=COUNTIF (range; criteria)*

Calc: *=ПРЕБРОЈАКО(опсег; критериум)*

каде *range/опсег* е опсег на ќелии кои се испитуваат (на пр. E3:E10), а *criteria/критериум* е логички услов чија вредноста во ќелијата мора да го исполни за да се преброи (на пр. “>10”).

Пр. 5. 15. Да се одреди бројот на учениците кои го положиле тестот.

	A	B	C
1	Име и презиме	Поени	Оцена
2	Дарко Јовановски	35	2
3	Ана Мишева	100	5
4	Маја Митиќ	92	5
5	Марко Костов	64	4
6	Дина Ацевска	12	1
7	Јане Петров	/	/
8	Енис Бајрами	74	4
9	Миа Лазова	2	1
10	Вкупно ученици:		8
11	Вкупно полагале:		7
12	Дали сите полагале?		
13	Положиле:		5

MS Excel:

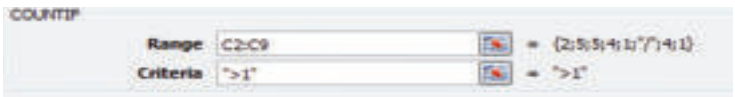
=COUNTIF(C2:C9; “>1”)

Calc:

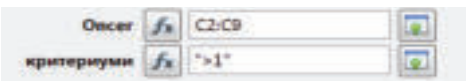
=ПРЕБРОЈАКО(C2:C9; “>1”)

Сл. 5. 18 Пример за користење на функцијата COUNTIF / ПРЕБРОЈАКО

Внесување на аргументи на функцијата COUNTIF во MS Excel и ПРЕБРОЈАКО во Calc се дадени на следниве слики:



Сл. 5. 19 Внесување на аргументи на функцијата COUNTIF



Сл. 5. 20 Внесување на аргументи на функцијата ПРЕБРОЈАКО

Чекор по чекор:

Во работен лист внеси ги следниве податоци:

	A	B
1	Име	Место
2	Марко Николов	Скопје
3	Маја Андоновска	Кичево
4	Гоце Петрески	Прилеп
5	Арбен Јусуфи	Гостивар
6	Ана Марковиќ	Скопје
7	Зорица Петрова	Скопје

Преброј колку ученици се од Скопје:

1. Позиционирај се во ќелијата B8;
2. На лентата за формули кликни на копчето *Insert Function*/*Волшебник*;
3. Во прозорецот Insert *Function*/*Волшебник* за функции најди ја функцијата COUNTIF/ ПРЕБРОЈАКО и кликни на копчето *ОК*/*Следно*;
4. Во полето *Range*/*Опсег* напиши B2:B78 (или можеш да ги означиш овие ќелии);
5. Во полето *Criteria*/*критериум* напиши "Скопје";
6. Кликни на копчето *ОК*/*Во ред*;
7. Функцијата гласи: =COUNTIF(B2:B7;"Скопје")/=ПРЕБРОЈАКО(B2:B7;"Скопје"), резултатот на функцијата е 3.

Функцијата SUMIF/СУМААКО

Функцијата SUMIF/СУМААКО се користи за условно собирање, т.е. собира вредности во ќелии од некој опсег за кои е исполнет одреден логички услов.

Синтакса на оваа функција е:

MS Excel: *=SUMIF(range; criteria;sum_range)*

Calc: *=СУМААКО(опсег;критериум;опсег_на_сумата)*

каде *range/опсег* е опсег на ќелии кои се испитуваат (на пр. C2:C8), *criteria/критериум* е логички услов кој вредност во ќелија мора да го исполни за да се собере (на пр. ">10"), *sum_range/опсег_на_сумата* е незадолжителен аргумент кој се користи само ако се бара збир на ќелии кои не се во истиот опсег со ќелии за кои се испитува дадениот услов.

Пр. 5. 16. Да се соберат поени на учениците кои го положиле тестот.

	A	B	C
1	Име и презиме	Поени	Оцена
2	Дарко Јовановски	35	2
3	Ана Мишева	100	5
4	Маја Митиќ	92	5
5	Марко Костов	64	4
6	Дина Ацевска	12	1
7	Јане Петров	/	/
8	Енис Бајрами	74	4
9	Миа Лазова	2	1
10	Вкупно ученици:		8
11	Вкупно полагале:		7
12	Дали сите полагале?		
13	Положиле:		5
14	Просечна оцена на учениците кои положиле:		
15	Вкупно поени на учениците кои положиле:		365

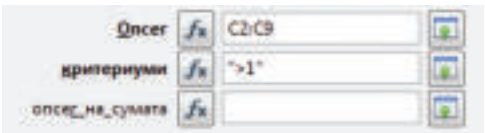
Сл. 5. 21 Пример за користење на функцијата SUMTIF/ СУМААКО

MS Excel:
=SUMIF(C2:C9; ">1";B2:B9)

Calc:
=СУМААКО(C2:C9; ">1";B2:B9)

Внесување на аргументите на функцијата SUMIF во MS Excel и СУМААКО во Calc се дадени на следниве слики:

Сл. 5. 22 Внесување на аргументи на функцијата SUMIF



Сл. 5. 23 Прозорец за внесување аргументи на функцијата СУМААКО

Зад. 5. 2. Во ќелијата C13 да се најде просечна оцена на учениците кои го положиле тестот. Просечната оцена се пресметува така што се собираат сите оцени кои се поголеми од 1 и тој збир се дели со бројот на учениците кои го положиле тестот.

Чекор по чекор:

Отвори нова работна тетратка и внеси ги следниве податоци:

	A	B	C
1	Име	Место	Патни трошоци
2	Марко Николов	Скопје	60
3	Маја Андоновска	Кичево	1050
4	Гоце Петрески	Прилеп	1200
5	Арбен Јусуфи	Гостивар	700
6	Ана Марковиќ	Скопје	60
7	Зорица Петрова	Скопје	60

Да се соберат патните трошоци за учениците кои не се од Скопје:

1. Позиционирај се во ќелијата C8;
2. На лентата за формули кликни на копчето *Insert Function*/*Волшебник*;
3. Во прозорецот Insert *Function*/*Волшебник за функции* најди ја функцијата SUMIF/ СУМААКО и кликни на копчето *ОК*/*Следно*;
4. Во полето *Range*/*Опсег* напиши B2:B7 (или означи ги овие ќелии);
5. Во полето *criteria*/*критериум* напиши “<>Скопје“;
6. Во полето *Sum_range*/*опсег_на_сумата* напиши C2:C7 (или означи ги овие ќелии);
7. Кликни на копчето *ОК*/*Во ред*;
8. Функцијата гласи: =SUMIF(B2:B7;”<>Скопје”;C2:C7)
   =СУМААКО (B2:B7;”<>Скопје”;C2:C7)
   Резултатот на функцијата е 2950.

Функцијата IF/АКО

Функцијата IF/АКО е условна функција чија синтакса е:

MS Excel:

=IF(Logical_test; Value_if_true; Value_if_false)

Calc:

=АКО (тест; тогаш_вредност;инаку_вредност)

Аргументи на оваа функција се:

- *Logical_test/тест* – логички услов (на пр. A5>=100) кој може да биде точен (TRUE) или неточен (FALSE) ,

- *Value_if_true/тогаш_вредност* – вредност која функцијата ќе ја има доколку логичкиот услов е точен,

- *Value_if_false/инаку_вредност* – вредност која функцијата ќе ја има доколку логичкиот услов не е точен.

Значи, функцијата IF/АКО проверува дали условот наведен во полето *Logical test/ тест* е исполнет – ако е исполнет, функцијата ќе има вредност која е дадена во полето *Value_if_true/тогаш_вредност*; ако условот не е исполнет, функцијата ќе има вредност која е дадена во полето *Value_if_false/инаку_вредност*.

Пр. 5. 17. Да се провери дали сите ученици го полагале тестот! Ако сите полагале во полето C14 да се напише „Да“, во спротивно да се напише „Не“.

	A	B	C
1	Име и презиме	Поени	Оцена
2	Дарко Јовановски	35	2
3	Ана Мишева	100	5
4	Маја Митиќ	92	5
5	Марко Костов	64	4
6	Дина Ацевска	12	1
7	Јане Петров	/	/
8	Енис Бајрами	74	4
9	Миа Лазова	2	1
10		Вкупно ученици:	8
11		Вкупно полагале:	7
12		Дали сите полагале?	Не

MS Excel:

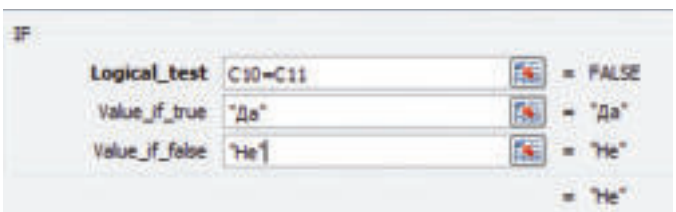
=IF(C10=C11; “Да“; ”Не”)

Calc:

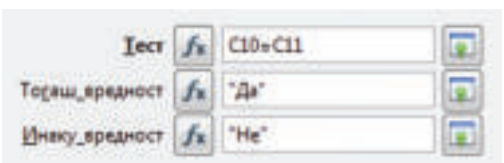
=АКО (C10=C11; “Да“; ”Не”)

Сл. 5. 24 Пример за користење на функцијата IF/АКО

Внесување на аргументите на функцијата SUMIF во MS Excel и СУМААКО во Calc се дадени на следниве слики:



Сл. 5. 25 Внесување на аргументи на функцијата IF



Сл. 5. 26 Внесување на аргументи на функцијата АКО

Последното барање може да се реши и со функцијата

=IF(COUNT(C2:C9)=COUNTA(C2:C9);”Да се повикаат родителите”).

Забележуваш дека во оваа сложена функција како аргумент се појавуваат функциите COUNT и COUNTA. Функции кои се појавуваат како аргументи на некоја друга функција се нарекуваат *вгнездени функции*.

Зад. 5. 3. Одреди кои се вгнездени функции во следните функции:

=IF(AVERAGE(E2:E10)>10;SUM(G2:G10);0) _____

=IF(A1=1;AVERAGE(L12:L22);AVERAGE(L12:L21)) _____

Чекор по чекор:

Во работен лист внеси ги следниве податоци:

C	D
	Вредност
	600,00 ден.
	300,00 ден.
	450,00 ден.
	125,00 ден.
Вкупно:	1.475,00 ден.
За наплата:	

За сметка поголема од 1500 денари да се наплати 90% од сметката:

1. Во ќелијата D6 најди го збирот на вредностите на ќелиите D1 до D5;
2. Позиционирај се во ќелијата D7;
3. На лентата за формули кликни на копчето *Insert Function*/*Волшебник*;
4. Во прозорецот *Insert Function*/*Волшебник за функции* најди ја функцијата IF/АКО и кликни на копчето *ОК*/*Следно*;
5. Во полето *Logical_test*/*Тест* напиши D6>1500;
6. Во полето *Value_if_true*/*Тогаш_вредност* напиши D6 *90%;
7. Во полето *Value_if_false*/*Инаку_вредност* напиши D6;
8. Кликни на копчето *ОК/Во ред*;
9. Функцијата гласи: =IF(D6>1500;D6 *90%;D6)/=АКО(D6>1500;D6 *90%;D6), а резултатот е 1597,5 денари.

За љубопитните:

При работа со податоци и формули се случуваат грешки. Најчестите грешки се:
####### – колоната не е доволно широка за да во неа се прикаже вредност, колоната треба да се прошири,
#VALUE! – внесен е погрешен тип на податоци, на пр. текст во формула за собирање,
#DIV0! – обид да се дели со нула што не е можно (треба да се провери делителот)
#NAME? – непознато име (погрешно внесено име на функција или на опсег на ќелии)
#REF! – не постојат ќелии од кои се земаат податоци (најчесто се јавува кога ќе се избришат ќелии кои се аргументи на формула за пресметување на вредности во други ќелии),
#NUM! – користење на неприфатлив нумерички податок, на пр. ако се бара квадратен корен од негативен број
#NULL! – лошо дефиниран опсег на ќелии, на пр. =SUM (A1 A4).

Резиме

Функција може да се внесе во ќелија на два начина.

Првиот начин е со директно запишување при што во ќелија се внесува знакот еднакво (=), потоа името на функцијата и аргументи според синтаксата на функцијата.

Вториот начин е преку дијалог прозорец за внесување на функции кој се повикува преку копчето *Insert Function/Волшебник за функции* од лентата за формули. Во овој прозорец се внесуваат аргументите на функцијата.

Функцијата *COUNTA/ПРЕБРОЈА* се користи за пребројување на ќелии од некој опсег кои не се празни, т.е. ќелии во кои се наоѓа некоја вредност.

Функцијата *COUNT/ПРЕБРОЈ* пребројува ќелии од некој опсег во кои се наоѓа некоја нумеричка вредност.

Функцијата *COUNTIF/ПРЕБРОЈАКО* се користи за условно пребројување, т.е. ги пребројува ќелиите од опсегот чии вредности го задоволуваат дадениот критериум.

Функцијата *SUMIF/СУМААКО* се користи за условно собирање, т.е. собира вредности во ќелии од некој опсег за кои е исполнет одреден логички услов.

Функцијата *IF/АКО* е условна функција која проверува дали условот наведен во полето *Logical test/тест е исполнет* – ако е исполнет, функцијата ќе има вредност која е дадена во полето *Value_if_true/тогаш_вредност;* ако не е, ќе има вредност која е дадена во полето *Value_if_false/инаку_вредност*.

Вештини што треба да ги усовршиш:

Да вметнеш функција во работен лист.

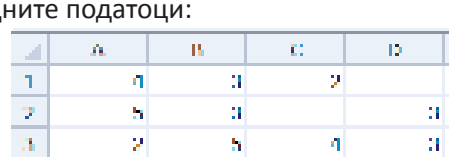
Правилно да ги користиш функциите:

- COUNTA/ПРЕБРОЈА
- COUNT/ПРЕБРОЈ
- COUNTIF/ПРЕБРОЈАКО
- SUMIF/СУМААКО
- IF/АКО

Прашања:

1. Како се внесува функција во работен лист?
2. Како се повикува прозорецот *Insert Function/Волшебник за функции*?
3. Објасни ги функциите и нивните аргументи:
   - COUNTA/ПРЕБРОЈА
   - COUNT/ПРЕБРОЈ
   - COUNTIF/ПРЕБРОЈАКО
   - SUMIF/СУМААКО
   - IF/АКО

   Наведи примери за користење на функциите од претходното прашање!
4. Внесени се следните податоци:



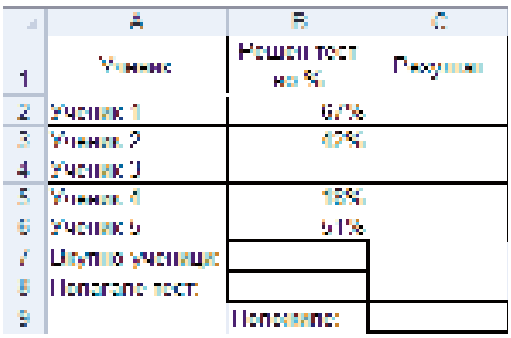
Колку ќе биде резултатот на функцијата:

а) =COUNT(A1:D2) б) =COUNT(A1:D2) в) =SUMIF(A1:D2;”>4”)

г) =COUNTIF(A1:D2;3) д) =IF(A1>4;A1+B1;A1-B1)

Задачи:

1. Во табелата се дадени резултати од тестот по информатика:



	A	B	C
1	Ученик	Решен тест во %	Резултат
2	Ученик 1	67%	
3	Ученик 2	47%	
4	Ученик 3		
5	Ученик 4	88%	
6	Ученик 5	51%	
7	Вкупно ученици:		
8	Полагале тест:		
9		Положиле:	

Во колоната С прикажи го резултатот од тестот за секој ученик: ако тестот е решен преку 50% резултатот е „Положил“ , во спротивно резултатот е „Не положил“.

Пресметај ги следните податоци:

- Во ќелијата B7 пресметај колку вкупно ученици се на списокот за полагање на тестот;
- Во ќелијата B8 пресметај колку вкупно ученици го полагале тестот;
- Во ќелијата C9 пресметај колку вкупно ученици го положиле тестот!

2. Во табелата се внесени податоци за бројот на продадените карти на еден стадион во период од 4 месеци:

	A	B	C	D	E	F
1	**Месец**	**Трибини**	**Продадени карти**			
2	Јан	Север	8680		**Збирно по трибини**	
3	Јан	Југ	5620		Север	
4	Јан	Запад	9098		Југ	
5	Јан	Исток	6253		Запад	
6	Фев	Север	7315		Исток	
7	Фев	Југ	8162		**ВКУПНО:**	
8	Фев	Запад	6543			
9	Фев	Исток	6532		**Збирно по месеци**	
10	Март	Север	5905		Јан	
11	Март	Југ	968		Фев	
12	Март	Запад	4789		Март	
13	Март	Исток	5267		Април	
14	Април	Север	3651		**ВКУПНО:**	
15	Април	Југ	2458			
16	Април	Запад	3968			
17	Април	Исток	7451			

Пресметај ги податоците *Збирно* по трибини и *Збирно по месеци*!

3. Внеси ги следниве податоци:

	A	B	C	D	E	F	G	H	I	J
1		мак	анг	фра	гер	мат	инф	физ	хем	Изостаноци
2	Јована Арсовска	5		5	5	4	4	5	5	6
3	Марко Симиќ	3		2	4	5	1	2	1	32
4	Мите Гоцев	1	3		3	3	1	1	1	12
5	Селим Османи	5		5	4	5	5	3	5	
6	Борче Костов	4	4		3	2	3	1	4	10
7	Јане Спировски	4	5		5	5	4	5	5	
8	Јована Арсовска	4	5		5	5	5	5	5	24
9	Борче Костов	2		1	4	3	1	3	2	
10	Ана Мицевска	4		4	4	5	4	5	5	30
11	Јане Спировски	1		3	2	3	4	2	2	6
12	Јована Арсовска	3	3		3	2	1	2	3	5

- За секој предмет одреди колку има оценети ученици;
- За секој предмет одреди колку ученици имаат оцена 5;
- Одреди колку ученици имаат изостаноци;
- За секој ученик одреди го бројот на негативните оцени (оцена 1);
- Собери ги изостаноците на учениците кои имаат една или повеќе оцени 1;
- За секој ученик да се пресмета средната оцена ако ученикот нема слаби оцени, во спротивно да се напише бројот на слабите оцени;
- За секој ученик да се провери дали има повеќе од 3 единици – ако има да се напише текстот "Да се повикаат родителите"!

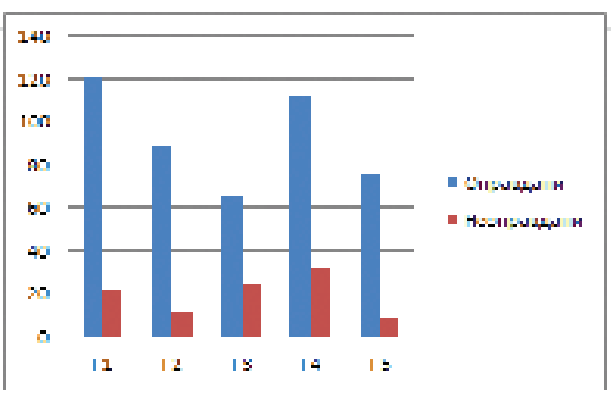
5.3 Напредна работа со графикони

Потсети се!

Графиконите се користат за графичко прикажување на податоците од табелите за полесен преглед и споредување на податоците.

Зад. 5. 4. Внеси податоци во табела и креирај графикон како на следната слика:

Паралелка	Изостаноци	
	Оправдани	Неоправдани
I 1	120	21
I 2	89	11
I 3	65	25
I 4	112	32
I 5	75	8



5.3.1 Елементи на графиконот

Графиконите се состојат од повеќе елементи, некои од нив се дефинираат и прикажуваат со самото креирање на графиконот, а некои можат да се додадат според потребите:

1. Област на графиконот;
2. Ѕид на графиконот;
3. Податочни точки и низи на податоците кои претставуваат вредности во графиконот;
4. Оски. Дводимензионалните графикони имаат две оски:
   - хоризонтална – оска на категорија и
   - вертикална – оска на вредности;
5. Легенда ги дефинира одделните низи на графиконот;
6. Наслов на графиконот и наслови на оските;
7. Ознаки на податоци во кои се претставуваат вредностите на податочните точки во низите на податоци.

Сл. 5. 27 Елементи на графиконот

5.3.2 Корекции на графиконот

Откако ќе се креира графикон, може да се измени било кој негов елемент, на пр. да се прикажат или сокријат некои елементи на графиконот или да се уреди изгледот на некои елементи и слично.

Уредување на графикон во MS Excel

Кога ќе се кликне на графиконот, на рибонот ќе се појават алатите за работа со графиконот *Chart Tools* распоредени на три картички: *Design* (дизајн), *Layout* (изглед) и *Format* (уредување).

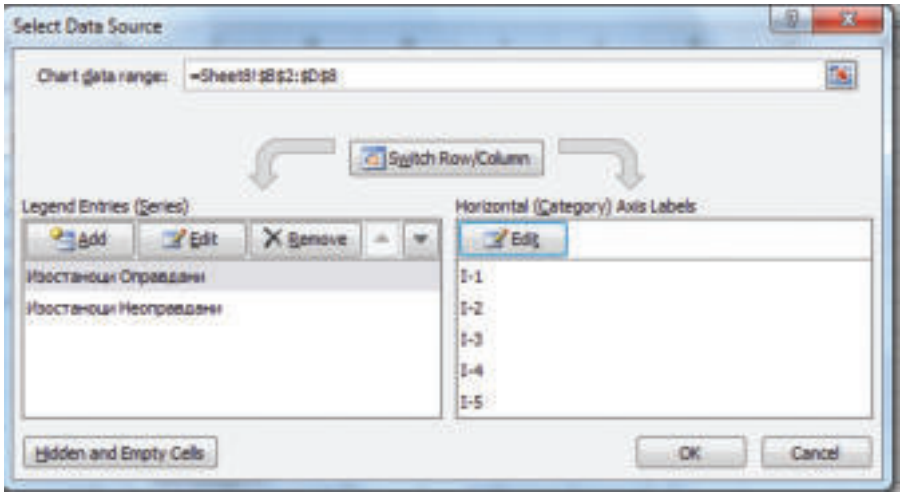
Примена на однапред дефиниран изглед и стил на графикон

Изгледот и стилот на графиконот може да се измени и да се избере еден од однапред дефинираните изгледи и стилови со помош на алатките од картичката *Design*.

Сл. 5. 28 Картичката Design од менито Chart Tools

1 Со копчето *Change Chart Type* може да се измени типот на графиконот;

2 Со копчето *Select Data* може повторно да се изберат податоци за кои се креира графикон во прозорецот *Select Data Source*;



Сл. 5. 29 Прозорец за означување на податоци за графикон

3 Со кликнување на надолната стрелка во групата *Chart Layouts* се отвораат различни изгледи на графикони од кој може да се избере изглед кој најмногу одговара според потребите;

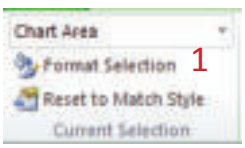
Сл. 5. 30 Изгледи на графикон

4 Со кликнување на надолната стрелка во групата *Chart Styles* се отвораат различни стилови на графиконите од кои може да се избере стил кој најмногу одговара.

Измена на изглед на елементите на графиконот

Прикажувањето и изгледот на елементите на графиконот може да се измени со помош на алатките од картичката *Layout*.

1 Со кликнување на копчето *Format Selection* преку соодветен прозорец се уредуваат површини, граници, сенки, големина и слично.



Сл. 5. 31 Копчето за уредување на графикон

Забелешка:

Изгледот кој ќе се избере се применува на оние елементи на графиконот кои се означени.

Во групата *Labels* се копчињата за прикажување и уредување на елементите:

2 наслов на графиконот,

3 наслов на оските,

4 легенда,

5 вредности на податоците и

6 табела со податоците.

Сл. 5. 32 Копчиња за уредување на елементите на графиконот

Во групата *Axes* се уредуваат:

7 оските и

8 линиите на оските.

Сл. 5. 33 Копчиња за уредување на оските на графиконот

Речиси сите од копчињата имаат опција одреден елемент да не се прикажува или да се прикаже на одредено место. Обиди се со различни опции кои ги даваат овие копчиња!

Додавање на насловот на графиконот и насловите на оските

Можат да се додадат наслови на графиконот и на оските за да се појаснат информациите на графиконот.

Забелешка:

Ако се смени типот на графиконот во тип кој не поддржува наслови, тие веќе нема да се прикажат, но пак ќе се појават ако се избере тип кој ги поддржува.

Додавање на легенда или таблица со податоци

Легенда може да се прикаже или да се сокрие и да се помести на друго место. За некои графикони може да се прикаже и таблица во која се прикажани вредностите прикажани со графиконот.

Сл. 5. 34 Графикон со прикажана табела со податоци

Совет:

За уредување на текст во елементите на графиконот, може да се кликне со десното копче на текстот и потоа текстот се уредува од малата лента со алатките кои ќе се појават до текстот.

Уредување на графикон во Calc

Откако во Calc е креиран графикон, можат да се изменат подесувањата направени во некој од чекорите при неговото креирање. Тоа се прави така што графиконот се означува и се кликнува на копчето *Графикон* во стандардната лента. Се отвора прозорецот *Волшебник за графикони* во кој може да се избере еден од чекорите:

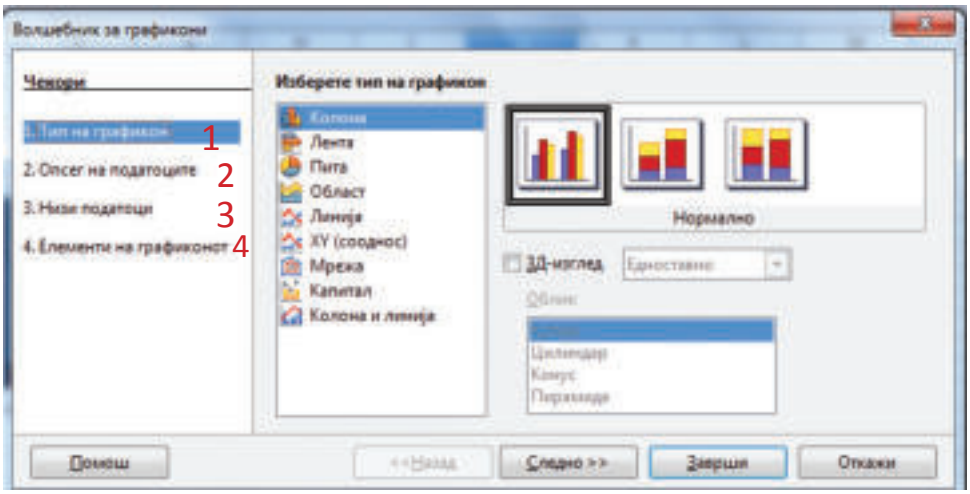
1 *Тип на графикон*

2 *Опсег на податоци*

3 *Низи податоци*

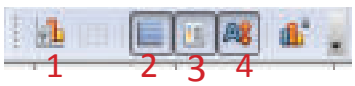
4 *Елементи на графиконот*

и да се изменат подесувањата на ист начин како при креирањето на графиконот.



Сл. 5. 35 Прозорецот Волшебник за графикони

За да се изменат елементите на графиконот, два пати се кликнува на него по што се добива помошна лента за форматирање со следниве копчиња:



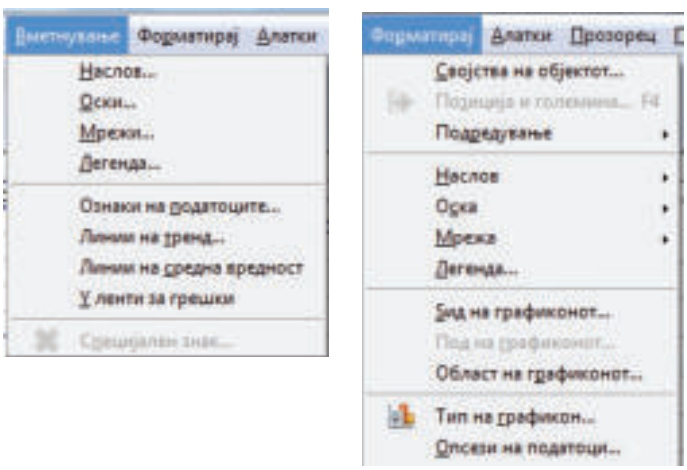
1 *Тип на графикон* со кое може да се избере друг тип на графикон;

2 *Хоризонтална мрежа Вклучи/Исклучи* со кое се вклучува или исклучува прикажување на хоризонтална мрежа од линии;

3 *Легенда Вклучи/Исклучи* со кое се вклучува или исклучува легенда на графиконот;

4 *Промени големина на текст* со кое се менува големината на текстот во графиконот кога ќе се промени големината на графиконот.

Кога два пати ќе се кликне на графиконот, во менијата *Вметнување* и *Форматирај* ќе има наредби кои се однесуваат на графиконот:

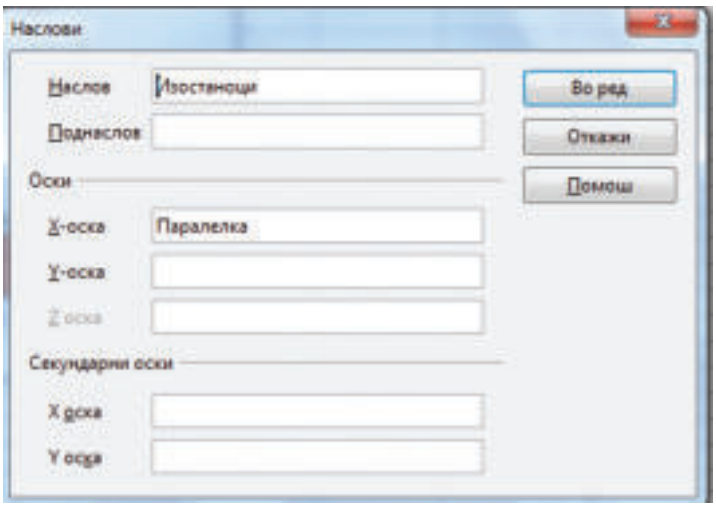


Сл. 5. 36 Наредби за графикон во менијата Вметнување и Форматирај

Вметнување на наслови

За да се вметне наслови во графиконот се повикува наредбата *Вметнување/ Наслов...* по што ќе се добие прозорецот *Наслови*. Во овој прозорец се пишуваат

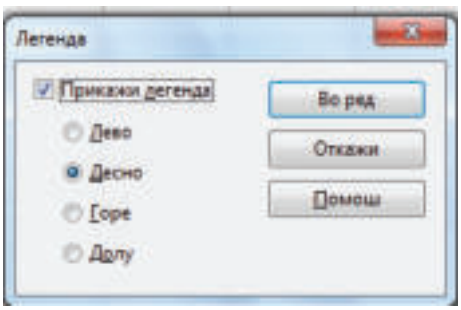
насловите во полињата *Наслов*, *Поднаслов*, *X-оска* и *Y-оска* и се кликнува на копчето *Во ред*.



Сл. 5. 37 Прозорец за вметнување на наслови во графикон

Прикажување на легенда

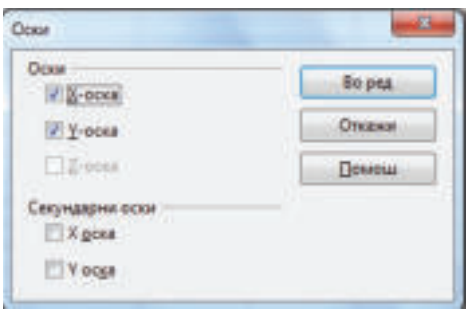
За да се прикаже легенда во графиконот, се повикува наредбата *Вметнување →Легенда...* по што ќе се добие дијалог прозорецот *Легенда*. Во овој прозорец се потврдува копчето *Прикажи легенда* по што ќе станат достапни копчињата *Лево, Десно, Горе* и *Долу* од кои се избира едно копче за место на прикажување на легендата. На крај се кликнува на копчето *Во ред*.



Сл. 5. 38 Прозорец за прикажување на легенда на графикон

Прикажување на вредности на оските

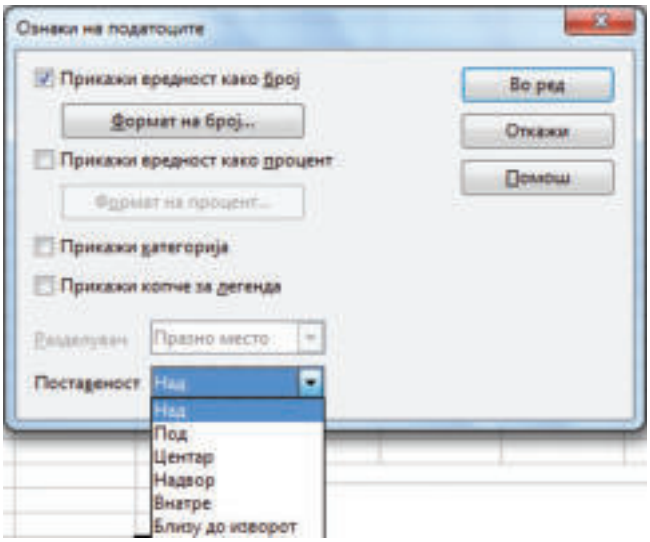
За да се прикажат вредности по хоризонталната (X) и по вертикалната (Y) оска се повикува наредбата *Вметнување→Оски...* по што се добива дијалог прозорецот *Оски*. Во овој прозорец се потврдуваат копчињата *X-оска и/или Y-оска*. На крајот се кликнува на копчето *Во ред*. Доколку графиконот е тридимензионален, ќе биде достапно и копчето *Z-оска*.



Сл. 5. 39 Прозорец за прикажување на вредности на оските на графикон

Прикажување на ознаки на податоците

За да се прикажат ознаки на податоците до податочните точки на графиконот се кликнува на наредбата *Вметнување→Ознаки на податоците...* по што се добива дијалог прозорецот *Ознаки на податоците*. Во овој прозорец се потврдуваат копчињата *Прикажи ознака како број и/или Прикажи ознака како процент*. Од паѓачката листа *Поставеност* се избира позиција на ознаките на податоците во однос на податочните точки на графиконот.



Сл. 5. 40 Прозорец за прикажување на ознаки за податоците на графикон

Уредување на елементите на графиконот

Прозорецот за уредување на елементите на графиконот се добива кога два пати ќе се кликне на некој од елементите или кога ќе се означи некој елемент и ќе се повика наредбата *Форматирај→Својства на објект*. Претходно треба два пати да се кликне на самиот графикон.

Прозорците за уредување на елементите на графиконот имаат различни точки во зависност за кој елемент се работи.

Истражи ги сите можности на прозорците за уредување на елементите на графиконите!

Резиме

Графиконите се состојат од повеќе елементи, некои од нив се дефинираат и се прикажуваат со самото креирање на графиконот, а некои можат да се додадат според потребите. Откако ќе се креира графикон, може да се измени кој било негов елемент.

MS Excel:

Кога ќе се кликне на графиконот, на рибонот ќе се појават алатите за работа со графиконот *Chart Tools* распоредени на три картички: *Design, Layout* и *Format*.

Изгледот и стилот на графиконот може да се измени и да се избере еден од однапред дефинирани изгледи и стилови со помош на алатките од картичката *Design*. Изгледот на елементите на графиконот може да се измени со помош на алатките од картичката *Layout*.

Calc:

Подесувањата на графиконот можат да се изменат во прозорецот *Волшебник за графикони*, кој се повикува така што графиконот ќе се означи и ќе се кликне на копчето *Графикон* во стандардната лента. За да се изменат елементите на графиконот, два пати се кликнува на него по што се добива помошна лента за форматирање. Прозорец за уредување на елементите на графиконот може да се добие кога два пати ќе се кликне на некој од елементите или кога ќе се означи некој елемент и ќе се повика наредбата *Форматирај/Својства на објект*.

Вештини што треба да ги усовршиш:

Да вметнеш графикон за соодветни податоци во работен лист.
Да прикажеш и да уредиш елементи на графикон.

Прашања:

1. Кои се елементи на графикон?
2. Како може да се избере изглед на графикон?
3. Како се прикажуваат насловите на графиконот и на оските?
4. Како се прикажуваат вредности на графиконот?
5. Како се прикажува легенда на графиконот?
6. Како се прикажува табела со вредности на графиконот?
7. Како може да се уредат елементите на графиконот?

Задачи:

1. Отвори нова работна книга! Внеси податоци како што е покажано (користи Мк поддршка, фонт Arial, големина10):

	A	B	C
1	Македонскиот идол		
2			
3	Натпреварувач	Гласови за победник	
4	Марија Димовска	7	
5	Петре Попов	5	
6	Ервин Бајрами	3	
7	Јован Митиќ	5	
8	Марко Петрески	4	

- Во ќелијата А1 смени ја големината на фонт на 16 точки;
- Прилагоди ја ширината на колоната А на 125 пиксели / 3,5 см и содржините израмни ги лево;
- Прилагоди ја ширината на колоната В на 120 пиксели / 3 см и содржините израмни ги во средина;
- Задебели ги буквите во редовите 1 и 3;
- Креирај графикон пита според следниве упатства:
  - Селектирај ги ќелиите од А1до В8;
  - Вметни графикон, избери го типот *Pie / Пита*, поттип *Pie / Нормално*;
  - За наслов на графиконот напиши „Македонскиот идол“;
  - Легендата постави ја десно од графиконот;
  - Прикажи ги вредностите на графиконот;
  - Графиконот зачувај го на нов лист; Преименувај го листот во „Графички приказ“;
  - Боите на графиконот уреди ги по избор;
  - Насловот на графиконот уреди го со задебелени букви, големина 24 точки, сина боја, останатиот текст е со големина 16 точки;
- Работната книга зачувај ја со име Makedonskiot idol.

2. Внеси ги следните податоци:

Име и презиме	Зад. 1	Зад. 2	Зад. 3	Зад. 4
Митко Грпов	18	10	22	20
Елена Митиќ	20	15	25	20
Селим Османи	20	10	15	18
Маја Андова	10	15	10	12
Марко Симов	12	8	15	8
Енвер Реџепи	20	15	25	20
Ана Мицевска	20	15	20	17
Алма Мемети	10	10	0	15
Мина Маркоска	0	15	0	10

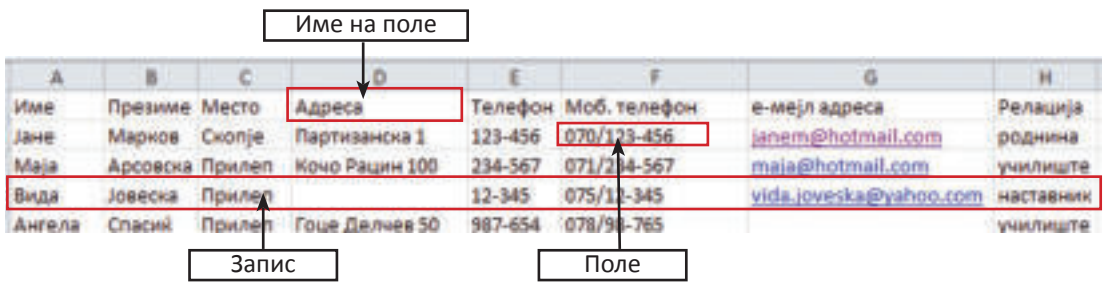
За дадената табела креирај графикон:

- Податоците од графиконот прикажи ги со различни типови на графикон;
- Додај наслови;
- Промени го изгледот на графиконот;
- Легендата прикажи ја на различни позиции;
- Додај вредности на графиконот;
- Уреди ги елементите на графиконот по желба.

5.4 Табела како база на податоци

База на податоци претставува организирана колекција од податоци. Во неа можат да се чуваат податоци за луѓе, за книги, за оцени, за автомобили и за било што друго. Базите на податоци не мора да се чуваат во компјутерот. На пр. телефонски именик, колекција на филмови на компакт дискови, адреси на пријатели итн. исто така се бази на податоци. Чување на бази на податоци во компјутерот овозможува нивна полесна и побрза обработка и добивање на саканата информација. База на податоци се користи за чување, организирање и пребарување на податоци. Постојат програми кои се специјализирани за работа со бази на податоци, како што се Access од пакетот Microsoft Office и Base од пакетот OpenOffice. Но и програмите за табеларно пресметување имаат можност да извршат некои едноставни операции со податоци во табела која може да се смета за едноставна база на податоци. Податоците од табелата можат да се сортираат, да се побараат податоци кои задоволуваат некој критериум и слично.

На пример, може да се креира база на податоци во која ќе се чуваат податоци за пријателите: име, презиме, адреса, телефон, релација (од клас, роднина и сл.). Во првиот ред се внесуваат наслови на колони, секој податок се внесува во посебна ќелија, податоците за секој пријател се внесуваат во нов ред.



Сл. 5. 41 Пример за табела како база на податоци

Основните елементи на секоја база на податоци се *поле*, *запис* и *име на поле*. Кога табелата претставува база на податоци, ќелијата претставува поле, редот претставува запис, а имињата на колоните претставуваат имиња на полиња.

За креирање на база на податоци доволно е во ќелиите да се внесат податоци на вообичаен начин, при што мора да се почитуваат следниве правила:

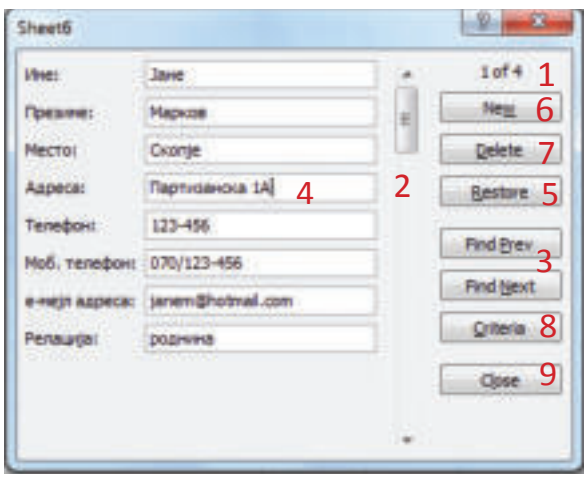
- Во првиот ред на базата мора да се внесат имиња на полиња (на пр. име, презиме, адреса);
- Не треба да има празен ред по редот со имиња на полињата, податоците се внесуваат веднаш во следниот ред;
- Секој запис мора да биде во посебен ред. Не смее да постои празен ред помеѓу записите;
- Ќелиите во една колона мора да содржат информации од ист тип, на пр. во колоната со наслов *Телефонски број* сите ќелии од оваа колона мора да содржат телефонски број, а не адреса или име. Ќелијата може да остане празна ако некоја информација не е позната, но тоа може подоцна да направи проблеми при сортирање на податоците па се препорачува да се внесе некој податок на пр. „непознато„ или „/“;
- Базата со податоци мора да биде само на еден работен лист.

5.4.1 Форма за податоци во MS Excel

Во базата можат да се додаваат, да се уредуваат и да се бришат податоци на вообичаен начин, но кога базата ќе стане голема ќе биде полесно да се користи форма преку која тоа ќе се направи на полесен начин.

Формата се повикува со кликнување на копчето *Form*  на картичката *Data*. Претходно мора да се кликне каде било во базата.

Откако ќе се кликне на копчето *Form* ќе се добие следнава форма:



Сл. 5. 42 Форма за додавање, за уредување и за бришење на податоци во база

1 Бројот на активниот запис се појавува во горниот десен агол;

2 За да се дојде до саканиот запис може да се користи лентата за лизгање

3 или копчињата *Find Prev* и *Find Next*;

4 За да се измени поле од запис, се кликнува во соодветно поле во формата и потоа се менува неговата содржина;

5 За да се поништат внесените измени во запис, се кликнува на копчето *Restore*;

6 За додавање на записи се кликнува на копчето *New*;

7 За бришење на активен запис се кликнува на копчето *Delete*;

8 За да се пребараат записи се кликнува на копчето *Criteria*, потоа во формата се внесува некој од познатите податоци или се задава услов за пребарување и се кликнува на истото копче кое ќе добие име *Form*;.

9 За затворање на формата се кликнува на копчето *Close*.

Задавање на услови за пребарување

При пребарување на записи за нумерички податоци се задаваат услови со користење на операторите за споредување (>, >=, =, <, <= и <>). За текстуалните податоци се користат знаците ѕвездичка (*) и прашалник (?). Ѕвездичката заменува повеќе знаци, а прашалникот заменува еден знак.

Чекор по чекор:

1. Отвори нова работна книга и зачувај ја со име Prodazba na tehnicka oprema! Креирај база на податоци како на сликата:

	A	B	C	D	E	F	G
1	Реден број	Производ	Категорија	Цена	Количина	Вработен	Град
2	1	Конфигурација Ѕенит	Компјутер	650	5	AA Computers	Скопје
3	2	Конфигурација Епсон	Компјутер	550	15	III Computers	Штип
4	3	Sony	Телевизор	400	3	AA Computers	Скопје
5	4	Nokia	Мобилен телефон	250	14	AA Mobile	Прилеп
6	5	HP	Печатач	300	5	III Mobile	Штип
7	6	Motorola	Мобилен телефон	320	6	AA Mobile	Прилеп
8	7	Samsung	Мобилен телефон	250	1	III Mobile	Штип
9	8	Тарси	Компјутер	500	12	CC Computers	Скопје
10	9	LG	Телевизор	450	12	III Computers	Прилеп
11	10	Epson	Печатач	280	21	III Computers	Прилеп
12	11	Конфигурација Тарси	Компјутер	700	5	III Computers	Прилеп
13	12	Sony	Монитор	190	6	III Computers	Штип
14	13	Philips	Монитор	220	9	AA Computers	Скопје
15	14	Конфигурација ППП	Компјутер	580	1	III Computers	Прилеп

2. На картичката *Data* кликни на копчето *Form*;
3. Со помош на формата додај го следниот запис: „31, Epson, печатач, 300, 12, AA Computers, Скопје“. Меѓу полињата во формата движи се со копчињата *Tab* (напред) и *Shift+Tab* (назад). По последното поле кликни на копчето *Enter* за да го внесеш записот во базата;
4. Најди и разгледај ги сите записи од категоријата компјутер:
   - Кликни на копчето *Criteria*;
   - Во полето „Категорија“ напиши „Компјутер“;
   - Кликни на копчето *Form*;
   - Кликнувај на копчето *Find Next* за да ги видиш сите записи кои го задоволуваат условот.

5. Најди и разгледај ги сите записи во кои количината е поголема од 10:
   - Кликни на копчето *Criteria*;
   - Во полето „Категорија“ избриши го условот „Компјутер“;
   - Во полето „Количина“ напиши „>10“;
   - Кликни на копчето *Form*;
   - Кликнувај на копчето *Find Next* за да ги видиш сите записи кои го задоволуваат условот.
6. Избриши го записот со реден број 6:
   - Кликни на копчето *Criteria*;
   - Доколку има поставено услови за пребарување, избриши ги;
   - Во полето „Нарачка број“ напиши 6;
   - Кликни на копчето *Form*;
   - Провери дали е покажан запис со редниот број 6, ако е покажан, кликни на копчето *Delete*.

5.4.2 Сортирање на податоци

Сортирањето на податоци значи податоците да се подредат според некој критериум. На овој начин податоците во табелата стануваат попрегледни. Податоците можат да се сортираат по азбучен редослед од А до Ш или обратно за текстуални податоци, од најголем до најмал или обратно за нумерички податоци, од најстар до најнов или обратно за датуми, а може да се креира и прилагоден попис.

Пр. 5. 18. Во следнава табела податоците се сортирани според полето „Вкупно поени“ од најголем до најмал.

	A	B	C	D	E	F
1	Презиме	Име	Основно училиште	Поени од тест	Поени од награди	Вкупно поени
2	Расиќ	Слава	Страшо Пинџур	75,00	3	78,00
3	Ангелова	Симона	Владо Тасевски	75,00		75,00
4	Мицевски	Петар	Аџо Шопов	68,45	5	73,45
5	Михајловски	Бобан	Аџо Шопов	72,88		72,88
6	Ангелова	Бојана	Петар Поп Арсов	64,08	5	69,08
7	Благоевска	Теодора	Мирче Ацев	67,30	1	68,30
8	Додева	Марија	Лазо Ангеловски	65,00	3	68,00
9	Николовска	Стефани	Лазо Ангеловски	64,75		64,75
10	Додева	Сања	Страшо Пинџур	64,40		64,40
11	Михајловски	Зоран	Лазо Ангеловски	58,08	5	63,08
12	Додева	Мирјана	Димитар Миладинов	62,00		62,00
13	Михајловски	Горан	Страшо Пинџур	48,45		48,45
14	Ангелова	Ивана	Лазо Ангеловски	45,10		45,10

Важно!

Пред сортирањето важно е да се позиционира каде било во базата или да се означат сите податоци кои ќе се сортираат, но НИКАКО не смее да се означи само колона во која се наоѓаат податоци. На тој начин се сортира само таа колона и се губи врска со останатите податоци. На пр. кога во претходната табела податоците би ги сортирале според колоната „Презиме“ и би ја означиле само таа колона, би го добиле следниот резултат со што податоците веќе не одговараат на вистинските податоци.

	A	B	C	D	E	F
1	Презиме	Име	Основно училиште	Поени од успех	Поени од награди	Вкупно поени
2	Ангелов	Славе	Страшо Пинџур	75,00	3	78,00
3	Ангелов	Симона	Владо Тасевски	75,00		75,00
4	Ангелов	Петар	Аџо Шопов	68,45	5	73,45
5	Блажевски	Бобан	Аџо Шопов	72,38		72,38
6	Додов	Бојана	Петар Поп Арсов	64,08	5	69,08
7	Додов	Теодора	Мирче Ацев	67,50	1	68,50
8	Додов	Марија	Лазо Ангеловски	65,00	3	68,00
9	Михајловски	Стефани	Лазо Ангеловски	64,75		64,75
10	Михајловски	Сања	Страшо Пинџур	64,40		64,40
11	Михајловски	Зоран	Лазо Ангеловски	58,08	5	63,08
12	Мицевски	Мирјана	Димитар Миладинов	52,00		52,00
13	Николовска	Гордана	Страшо Пинџур	49,45		49,45
14	Рогиќ	Ивана	Лазо Ангеловски	45,10		45,10

Сортирање по повеќе критериуми

Податоците во база можат да се сортираат по повеќе критериуми, на пр. доколку податоците се сортирани според презиме и има повеќе записи со исто презиме, тие ќе се сортираат и според име.

	A	B	C	D	E	F
1	Презиме	Име	Основно училиште	Поени од успех	Поени од награди	Вкупно поени
2	Ангелова	Сузана	Пенчо Поп Арсов	64,88	5	69,88
3	Ангелова	Милена	Лазо Ангеловски	65,10		65,10
4	Ангелова	Оливера	Видое Подгорец	65,80		65,80
5	Јаневски	Григорие	Мирче Ацев	67,80	1	68,80
6	Јаневски	Марија	Лазо Ангеловски	65,00	3	68,00
7	Јаневски	Мирјана	Димитар Миладинов	52,00		52,00
8	Јаневски	Сања	Страшо Пинџур	64,40		64,40
9	Мицевски	Бобан	Аџо Шопов	72,38		72,38
10	Мицевски	Гордан	Страшо Пинџур	49,45		49,45
11	Мицевски	Зоран	Лазо Ангеловски	58,08	5	63,08
12	Мицевски	Петар	Аџо Шопов	66,55	5	71,55
13	Николовска	Стефани	Лазо Ангеловски	64,75		64,75

Совет:

Пишувај со македонска поддршка за сортирањето да биде по азбучен редослед.

Сортирање на податоци во MS Excel

За да се сортираат податоци, се позиционира каде било во базата и на картичката Data се кликнува на копчето *Sort* по што ќе се отвори прозорецот *Sort*:

Сл. 5. 43 Прозорец за сортирање во MS Excel

1 Во паѓачката листа *Sort by* се избира наслов на колона според која се врши сортирање;

2 Во паѓачката листа *Criteria* се избира начин на сортирање: по растечки (*A to Z* за текстуални или *Smallest to Largest* за нумерички податоци) или по опаѓачки (*Z to A* за текстуални или *Largest to Smallest* за нумерички податоци) редослед;

3 Ако табела со податоци има наслови на колони тие не треба да бидат сортирани, па полето *My data has headers* треба да биде потврдено;

4 Нов критериум се додава со кликнување на копчето *Add Level*;

5 Се избира наслов на колона и се поставуваат критериуми исто како во првото ниво;

6 Критериум се брише со кликнување на копчето *Delete Level*;

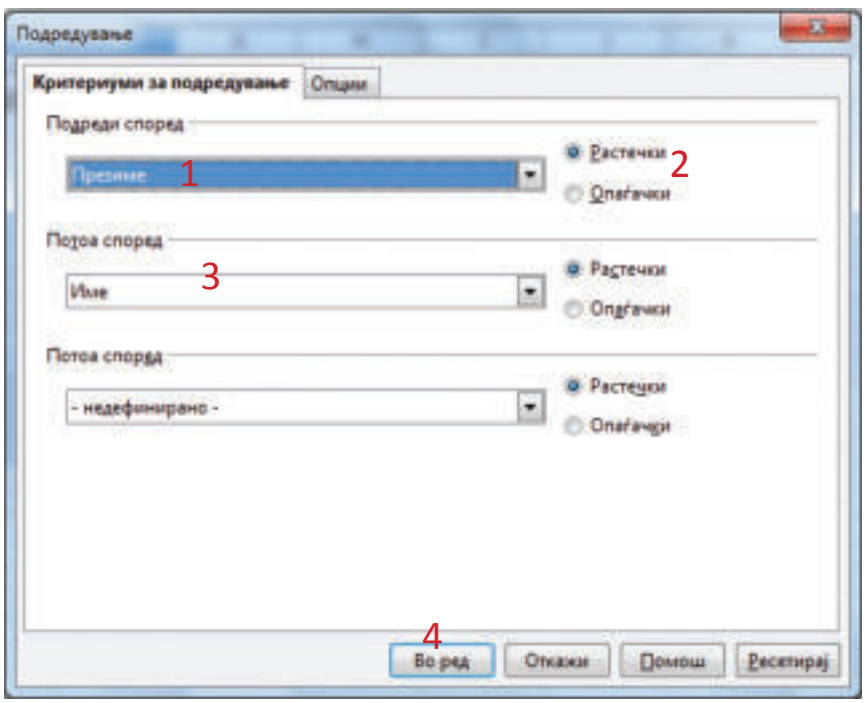
7 Кога ќе се постават критериуми за сортирање се кликнува на копчето *OK*.

Забелешка:

За сортирање податоци според првата колона во табелата, можат да се користат и копчињата - за сортирање по растечки редослед, или - за сортирање по опаѓачки редослед, од картичката Data.

Сортирање на податоци во Calc

За сортирање податоци, се позиционира каде било во базата и се повикува наредбата *Податоци→Подреди...* по што се отвора прозорецот *Подредување*:



Сл. 5. 44 Прозорец за сортирање во Calc

1 Во делот *Подреди според* од паѓачката листа се избира наслов на колоната според која се да врши сортирање;

2 Редоследот на сортирањето се избира со потврдување на едно од копчињата: *Растечки* или *Опаѓачки*;

3 За сортирање податоци по уште еден критериум, чекорите 1 и 2 се повторуваат во делот *Потоа според*;

4 Кога ќе се постават критериуми за сортирање се кликнува на копчето *Во ред*.

Забелешка:

За сортирање податоци според првата колона во табелата, можат да се користат копчињата - за сортирање по растечки редослед, или - за сортирање по опаѓачки редослед.

Чекор по чекор:

Креирај база на податоци со полиња: Име, Презиме и Оценка!

Име	Презиме	Оценка
Мирјана	Додева	3
Бобан	Михајловски	2
Петар	Мицевски	4
Симона	Трајкова	5
Сања	Панчева	4
Стефани	Николовска	5
Александар	Тошевски	4
Зоран	Рачиќевиќ	3
Славко	Бинов	2
Марија	Милошевска	3
Бојана	Стојчевска	5

Резултатите од тестот сортирај ги според оценатa, од највисоката до најниската. Ако повеќе ученици имаат иста оцена, сортирај ги според азбучен редослед.

Означи ги сите податоци освен насловите на колоните или кликни каде било во табелата!

MS Excel:

1. На картичката *Data* кликни на копчето *Sort*;
2. Во прозорецот *Sort*, во паѓачката листа *Sort by* избери Оцена, а во полето *Order* избери *Largest to Smallest*;
3. Кликни на копчето *Add Level*;
4. Во второто ниво, во паѓачката листа *Then by* избери Презиме, а во полето *Order* избери *A to Z*;
5. Кликни на копчето *OK*.

Calc:

1. Повикај ја наредбата *Податоци→ Подреди*;
2. Во прозорецот *Подредување*, во делот *Подреди според*, од паѓачката листа избери Оцена, и потврди го копчето *Опаѓачки*;
3. во делот *Потоа според*, од паѓачката листа избери Презиме, и потврди го копчето *Растечки*;
4. Кликни на копчето *Во ред*.

5.4.3 Филтрирање на податоци

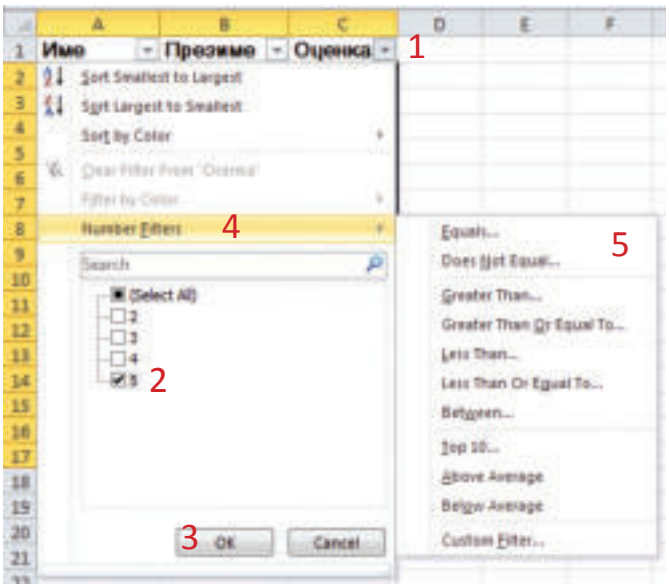
Филтрирањето на податоци овозможува на едно место да се видат сите податоци кои исполнуваат некој услов, на пр. сите ученици од прва година, сите нарачки од одреден купувач и слично. Филтрирање, всушност, значи прикажување само на одредени податоци додека други податоци кои не го исполнуваат условот се сокриени.

Пр. 5. 19. Во следнава табела прикажани се само оние ученици кои на тестот добиле оцена 5:

	A	B	C
1	Име	Презиме	Оцени
5	Симона	Трајкова	5
7	Стефани	Николовска	5
12	Бојана	Стојчевска	5

Филтрирање на податоци во MS Excel

За да се примени филтер на податоци во база, се позиционира каде било во базата и на картичката *Data* се кликнува на копчето *Filter* по што во првиот ред на табелата во секоја колона ќе се појави стрелка за паѓачка листа:



Сл. 5. 45 Поставување на филтер на податоци во MS Excel

1 Се кликнува на стрелка во колона во која се наоѓа поле за кое се поставува услов, по што се отвора листа во која се прикажани сите вредности кои постојат во колоната;

2 Ако се бараат податоци со точно одредени вредности, тие вредности се потврдуваат во полињата за потврда;

3 и се кликнува на копчето *ОК*;

4 За поставување на дополнителен услов се избира филтер (за текст, за броеви, за датум);

5 Од дополнителната листа се бира соодветен оператор.

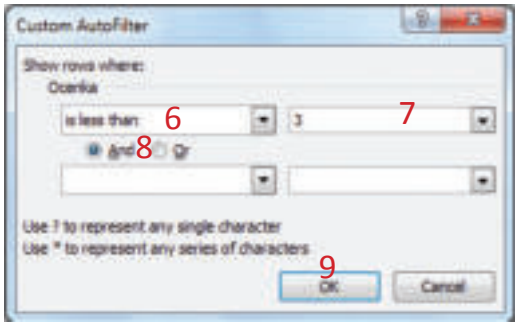
Се отвора дијалог прозорецот *Custom Auto Filter* во кој се довршува поставување на условот.

6 Во полето лево, доколку сакаме, може да се смени операторот;

7 Во полето десно се избира или се пишува вредност за условот;

8 Со копчињата *And* или *Or* можат да се креираат сложени услови;

9 Откако ќе се постави условот, се кликнува на копчето *ОК*



Сл. 5. 46 Поставување на дополнителни услови за филтрирање податоци

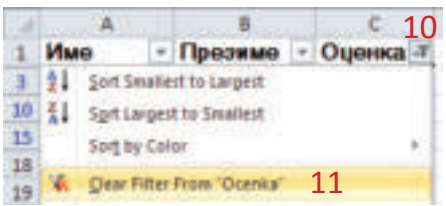
по што во табелата ќе бидат прикажани само оние податоци кои го задоволуваат условот.

	A	B	C
1	Име	Презиме	Оценка
3	Бобан	Михајловски	2
10	Славко	Бинов	2

Забелешка:

Можат да се постават повеќе филтри на различни колони.

10 За да се отстрани филтерот и повторно да се прикажат сите податоци, се отвора стрелката во колоната во која е поставен филтерот (на таа стрелка има икона)



11 и се избира *Clear Filter From "Име на колона"*.

За да се тргнат стрелките од колоните и да се прикажат сите податоци, повторно се кликнува на копчето *Filter* во картичката *Data*.

Чекор по чекор:

1. Отвори ја работната книга Prodazba na tehnicka oprema;
2. На картичката *Data* кликни на копчето *Filter*;
3. Кликни на стрелката до полето „Категорија“;
4. Во паѓачката листа потврди го изборот „Монитор“ (прво поништи ја потврдата во копчето *Select all*);
5. Кликни на стрелката до полето „Производ“;
6. Во паѓачката листа потврди го изборот „Sony“ (прво поништи ја потврдата во копчето *Select all*);
7. Кликни на копчето *ОК*;
8. Кликни на стрелката до полето „Производ“;
9. Во паѓачката листа кликни на опцијата *Clear Filter From "Производ"*;
10. Кликни на стрелката до полето „Категорија“;
11. Во паѓачката листа кликни на опцијата *Clear Filter From "Категорија"*;
12. Кликни на стрелката до полето „Цена“;
13. Во паѓачката листа кликни на опцијата *Number Filter*;
14. Од новата листа избери *Greather Than...;*
15. Во прозорецот *Custom AutoFilter*, во втората паѓачка листа напиши 400;

16. Кликни на копчето *ОК*;
17. Зачувај ја работната книга со име Prodazba na tehnicka oprema filter.

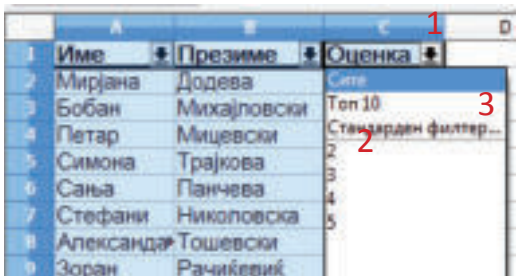
Филтрирање на податоци во Calc

За да се примени филтер на податоци во база, се позиционира каде било во базата и се повикува наредбата *Податоци→Филтер→Автоматски филтер* по што во првиот ред на табелата во секоја колона се појавува стрелка за паѓачка листа.

1 Се кликнува на стрелка во онаа колона во која се наоѓа поле за кое се поставува услов, по што се отвора листа во која се прикажани сите вредности кои постојат во колоната;

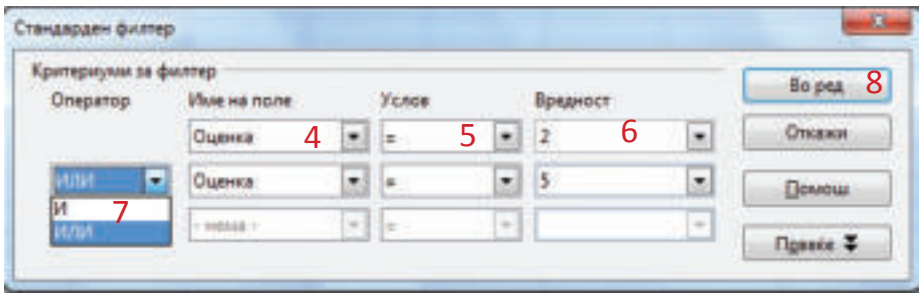
2 Ако се бараат податоци со точно одредени вредности, се кликнува на таа вредност во листата;

3 За да се постави дополнителен услов се кликнува на *Стандарден филтер*.



Сл. 5. 47 Поставување на автоматски филтер

Се отвора дијалог прозорец *Стандарден филтер* во кој се поставуваат услови за филтрирањето:



Сл. 5. 48 Поставување на стандарден филтер

Условите се поставуваат така што од паѓачките листи се избира

4 *Име на поле,*

5 *Услов* и

6 *Вредност* ;

7 Во паѓачката листа *Оператор* се избира еден од операторите *И/ИЛИ* со кои можат да се креираат сложени услови;

8 Откако ќе се постави условот, се кликнува на копчето *ОК*, по што во табелата ќе бидат прикажани само оние податоци кои го задоволуваат условот.

Забелешка:

Можат да се постават повеќе филтри на различни колони.

За да се отстрани филтер и повторно да се прикажат сите податоци, се кликнува на стрелката во колоната во која е поставен филтер (таа стрелка е со сина боја) и од паѓачката листа се кликнува на *Сите*.

За да се тргнат стрелките од колоните и да се прикажат сите податоци, се повикува наредбата *Податоци→Филтер...→Скриј автоматски филтер*.

Забелешка:

Кога ќе се повика наредба Податоци→Филтер...→Стандарден филтер, до имињата на полиња стрелките нема да се појават. Стандарден филтер се отстранува со наредбата Податоци→Филтер...→Отстрани филтер.

Чекор по чекор:

1. Отвори ја работната книга Prodazba na tehnicka oprema;
2. Повикај ја наредбата *Податоци→Филтер→Автоматски филтер*;
3. Кликни на стрелката до полето „Категорија“;
4. Во паѓачката листа кликни на податокот „Монитор“;
5. Кликни на стрелката до полето „Производ“;
6. Во паѓачката кликни на податокот „Sony“;
7. Кликни на стрелката до полето „Производ“;
8. Во паѓачката листа кликни на опцијата *Сите*;
9. Кликни на стрелката до полето „Категорија“;
10. Во паѓачката листа кликни на опцијата *Сите*;
11. Кликни на стрелката до полето „Цена“;
12. Во паѓачката листа кликни на опцијата *Стандарден филтер*;
13. Во прозорецот *Стандарден филтер*, во паѓачка листа *Име на поле* веќе е избрано полето „Цена“ (ако не е, избери го);
14. Во паѓачка листа „Услов“ избери го операторот *Поголемо од* (>);
15. Во паѓачка листа „Вредност“ напиши 400;
16. Кликни на копчето *Во ред*;
17. Зачувај ја работната книга со име Prodazba na tehnicka oprema filter.

Резиме

База на податоци е организирана колекција од податоци која се користи за чување, организирање и пребарување на податоци. Сортирање на податоци значи податоците да се подредат според некој критериум. Податоците во базата можат да се сортираат според повеќе критериуми. Филтрирањето значи прикажување само на одредени податоци додека други податоци кои не го исполнуваат условот се сокриени.

MS Excel:

На картичката *Data* се кликнува на копчето *Sort* по што се отвора прозорецот *Sort* во кој се бира поле според кое се сортира и се задаваат критериуми.

На картичката *Data* се кликнува на копчето Filter по што во првиот ред на табелата во секоја колона се појавува стрелка за *паѓачка* листа преку која се поставуваат критериуми за филтрирање.

Calc:

Се повикува наредбата *Податоци→Подреди...* по што се отвора прозорецот Подредување во кој се бира поле според кое се сортира и се задаваат критериуми.

Се повикува наредбата *Податоци→Филтер→Автоматски филтер* по што во првиот ред на табелата во секоја колона се појавува стрелка за паѓачка листа преку која се поставуваат критериуми за филтрирање.

Вештини што треба да ги усовршиш:

Да креираш база на податоци.

Да користиш форма за внесување, менување, бришење и пребарување на податоци во базата.

Да сортираш податоци во базата според еден или повеќе критериуми.

Да примениш филтер на податоци според одреден критериум.

Прашања:

1. Што е база на податоци?
2. Наведи пример за база на податоци?
3. Дали во програмите за табеларно пресметување може да се креира база на податоци?
4. Кои правила мора да се почитуваат при креирањето на базата на податоци?
5. Кои можности нуди форма за внесување на податоци во базата?
6. Како полесно ќе најдеш некој податок: со прелистување на базата или преку формата за внесување на податоци?
7. Што е сортирање на податоци?
8. Според колку критериуми податоците во базата можат да се сортираат?
9. Како се врши сортирање на податоци?
10. Што е филтер?
11. Како се врши филтрирање на податоци?
12. Како се отстранува поставен филтер во базата?

Задачи:

1. Креирај база на податоци за учениците од твојата паралелка!
   - Креирај база на податоци со следните полиња: Реден број во дневник, Презиме, Име на еден родител, Име, Датум на раѓање, Место на раѓање;
   - Во следниот ред внеси ги твоите податоци;
   - Со помош на форма внеси податоци за сите ученици од класот;
   - Најди и разгледај ги сите записи во кои името е исто како твоето име;
   - Најди и разгледај ги сите записи во кои презимето започнува со буквата А.
2. Сортирање на податоци!
   - Креирај база со следните полиња: Име, Презиме, Висина, Тежина. Внеси неколку записи. На работниот лист дај му име „Податоци“;
   - Податоците сортирај ги по азбучен редослед и копирај ги на друг работен лист на кој ќе му дадеш име „Список“;
   - На работниот лист „Податоци“ сортирај ги податоците според висина од најголем до најмал и копирај ги во нов работен лист на кој ќе му дадеш име „Висина“;
   - На работниот лист „Податоци“ сортирај ги податоците според тежина од најголем до најмал и копирај ги во нов работен лист на кој ќе му дадеш име „Тежина“;
   - Зачувај ја табелата.

3. Филтрирање на податоци!
   - Креирај база на податоци за упис на ученици во средно училиште (погледни го примерот 5.18);
   - Филтрирај податоци според критериумот: „Вкупно поени повеќе или еднакво на 65“. Податоците копирај ги во нов работен лист на кој ќе му дадеш име „Примени ученици“;
   - Во работниот лист „Примени ученици“ филтрирај податоци според критериумот: „Има поени од награди“ (потврди ги само вредностите 3 и 5). Податоците копирај ги во нов работен лист на кој ќе му дадеш име „Ученици со освоени награди“;
   - Во работниот лист „Примени ученици“ филтрирај податоци според критериумот: „Основното училиште е Страшо Пинџур“. Податоците копирај ги во нов работен лист на кој ќе му дадеш име „Учениците од ОУ Страшо Пинџур“.

5.5 Креирање извештаи – пивот табели

Изведени или пивот табели се користат за попрегледен распоред на податоци од една или од повеќе табели. Овие табели овозможуваат анализирање на голем број податоци и прикажување на информации според потребите. Како пример ќе земеме табела во која се води евиденција за полагање на курсеви за работа со компјутер:

	A	B	C	D	E	F
1	Курс	Кандидат	Време на полагање	Поени	Оцена	Да ли положил
2	windows	Ана Мицевска	јуни	90	5	да
3	windows	Јована Арсовска	јуни	74	3	да
4	windows	Маја Андова	јуни	85	4	да
5	windows	Марко Симиќ	јуни	92	5	да
6	windows	Мите Гоцев	јуни	68	3	да
7	windows	Селим Османи	јуни	87	4	да
8	word	Маја Андова	јуни	78	4	да
9	word	Марко Симиќ	јуни	37	1	не
10	excel	Маја Андова	јуни	92	5	да
11	windows	Јане Спировски	јули	69	3	да
12	word	Ана Мицевска	јули	88	4	да
13	word	Борче Костов	јули	98	5	да
14	word	Јане Спировски	јули	72	3	да
15	word	Јована Арсовска	јули	83	4	да
16	word	Марко Симиќ	јули	71	3	да
17	word	Мите Гоцев	јули	72	3	да
18	word	Селим Османи	јули	92	5	да
19	excel	Борче Костов	јули	100	5	да
20	excel	Јане Спировски	јули	35	1	не
21	excel	Јована Арсовска	јули	28	1	не
22	windows	Борче Костов	август	98	5	да
23	excel	Ана Мицевска	август	76	4	да
24	excel	Јане Спировски	август	65	2	да
25	excel	Јована Арсовска	август	78	4	да
26	excel	Марко Симиќ	август	82	4	да
27	excel	Мите Гоцев	август	64	2	да
28	excel	Селим Османи	август	68	3	да

Од вака дадена табела тешко е да се извлечат некои заклучоци, но со користење на пивот табела може да се дојде до корисни информации.

5.5.1 Пивот табела во MS Excel

Креирање на пивот табела

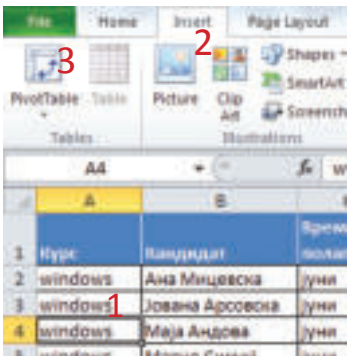
За да се креира пивот табела,

1 прво се позиционира, т.е. се кликнува каде било во табелата,

2 потоа се отвора картичката *Insert* и

3 се кликнува на копчето *Pivot Table* ,

4 по што се добива прозорецот *Create Pivot Table*.



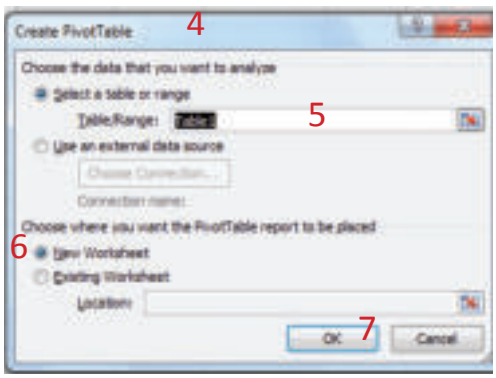
Сл. 5. 49 Креирање на пивот табела во MS Excel

Во прозорецот *Create Pivot Table*:

5 се означуваат сите податоци за кои се креира извештај, што се прикажува во полето *Table/Range* (ако се означени сите податоци од табелата во полето *Table/Range* пишува *Table1*),

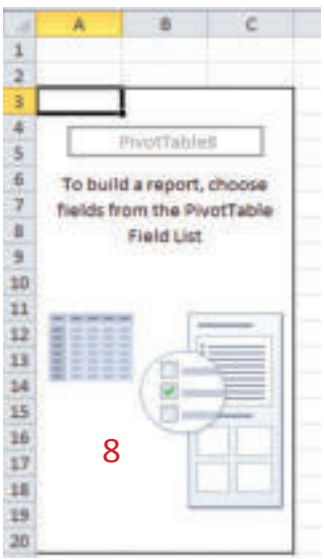
6 се избира полето *New Worksheet* (доколку има потреба пивот табела може да се креира и во истиот работен лист) и

7 се кликнува на копчето *OK*.



Сл. 5. 50 Прозорец за креирање на пивот табела

8 Изгледот на празна пивот табела е прикажан на следнава слика:



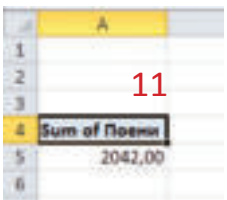
Сл. 5. 51 Празна пивот табела

Од десната страна на работниот екран се прикажани имиња на колоните во кои се напишани нашите податоци. Со едноставно повлекување на колоните во полињата под нив се добива анализа на податоците по различни критериуми.

9 За да се видат вкупно освоените поени на сите кандидати, по сите курсеви и во сите месеци, во полето *Value* се повлекува колоната „Поени“,

10 по што полето за потврда пред името на колоната поени ќе биде потврдено.

11 Се добива следниот извештај:



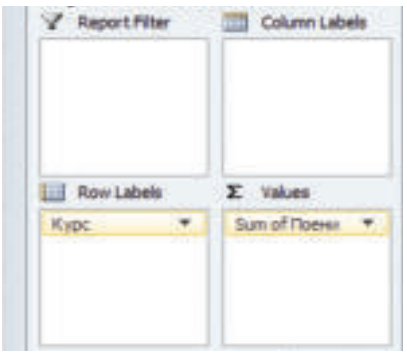
Сл. 5. 52 Пивот табела за збирот на поените

Сл. 5. 53 Поставување полиња за пивот табела за збирот на поените

За да се види по кои курсеви колку вкупно поени се освоени, во полето *Row Labels* се повлекува колоната „Курс“ по што се добива следниот извештај:

Row Labels	Sum of Поени
windows	663,00
word	691,00
excel	688,00
Grand Total	2042,00

Сл. 5. 54 Пивот табела за освоени поени по курсеви



Сл. 5. 55 Полиња за пивот табела за освоени поени по курсеви

За да се видат вкупно освоените поени за секој кандидат, во полето *Row Labels*, наместо колоната „Курс“ се повлекува колоната „Кандидат“ (се повлекува назад или се поништува полето за потврда пред колоната „Курс“) по што се добива следниот извештај:

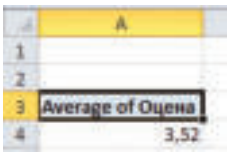
Row Labels	Sum of Поени
Ана Мицевска	254,00
Борче Костов	296,00
Јане Спировски	241,00
Јована Арсовска	263,00
Маја Андова	255,00
Марко Симиќ	282,00
Мите Гоцев	204,00
Селим Османи	247,00
Grand Total	2042,00

Сл. 5. 56 Пивот табела за освоени поени по кандидат

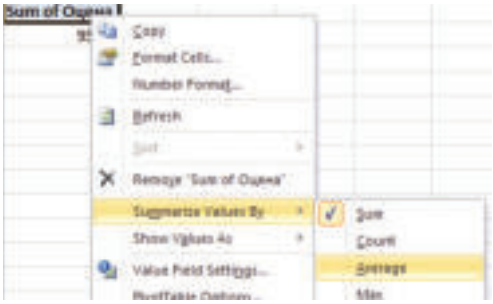
За љубопитните:

Пилот табелите нудат големи можности за креирање на многу различни извештаи. Доколку сакаме, на полињата може да се постави филтер. Доволно е да се кликне на стрелката до името на полето и да се изберат вредности за кои се креира извештај.

За да се види просечната оцена, во полето *value* се повлекува колоната „Оцена“. Потоа со десно копче од глувчето се кликнува на полето *Sum of Оцена* и од паѓачката листа се избира *Summarize Values By.* На крај од новата листа се избира *Average*, по што се добива бараниот податок.



Сл. 5. 57 Пивот табела за просечна оцена

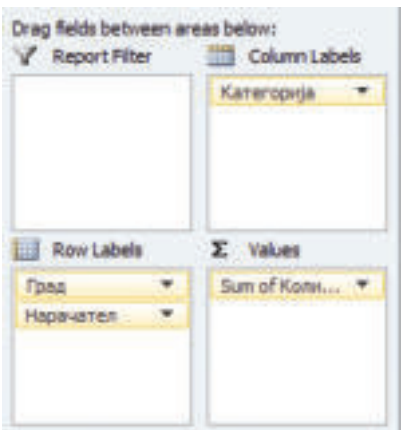


Сл. 5. 58 Добивање на пивот табела за просечна оцена

Чекор по чекор:

Креирај извештај во вид на пивот табела!

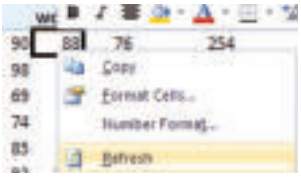
1. Отвори ја работната книга Prodazba na tehnicka oprema;
2. Кликни каде било во табелата;
3. Кликни на картичката *Insert*;
4. Кликни на копчето *Pivot Table*, ќе се отвори прозорецот *Create Pivot Table*;
5. Веќе се означени сите податоци од табелата (вклучувајќи ги и имињата на колоните), рангот на означените податоци се гледа во полето *Table/Range*; Доколку не се означени податоците, означи ги;
6. Веќе е потврдена опцијата *New Worksheet*, доколку не е, потврди ја;
7. Кликни на копчето *OK*, се појавува празна пивот табела;
8. Повлечи ги колоните во полињата како што е покажано на сликата десно;
9. Ќе добиеш табела како на сликата долу.



3	Sum of Количина	Column Labels					
4	Row Labels	Компјутер	Мобилен телефон	Монитор	Печатач	Тонер	Grand Total
5	Битола	18	1	8	6		33
6	DB Computers	18		8			26
7	DB Mobile		1		6		7
8	Прилеп	6	22		21	12	61
9	DD Computers	6			21	12	39
10	AA Mobile		22				22
11	Скопје	13		4		1	24
12	CC Computers	13					13
13	AA Computers	5		4		1	11
14	Grand Total	41	23	12	27	13	121

Освежување на пилот табела

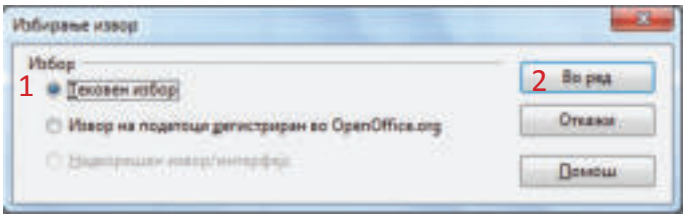
Често пати се случува откако ќе се креира пивот табела изворните податоци да се изменат. Тогаш е потребно пивот табелата да се освежи. Се кликнува со десното копче каде било во пивот табелата и од паѓачкото мени се избира *Refresh*.



5.5.2 Пивот табела во Calc

Креирање на пивот табела

За да се креира пивот табела, се повикува наредбата *Податоци→Пилот за податоци→Стартувај...* по што се добива прозорецот *Избирање извор*

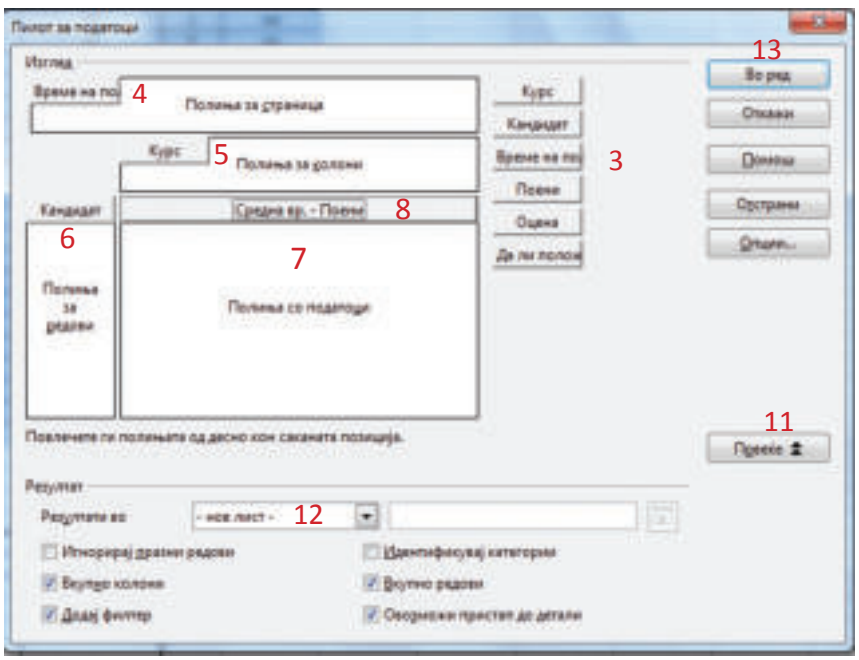


Сл. 5. 59 Прозорецот Избирање извор

1 во кој се потврдува изборот *Тековен извор*

2 и се кликнува на копчето *Во ред.*

Се отвара прозорецот *Пилот за податоци*:



Сл. 5. 60 Прозорецот Пилот за податоци

3 Во средишниот дел се прикажани имиња на полињата од базата кои се повлекуваат десно до соодветните полиња:

4 *Полиња за страница* – полиња според кои ќе можат да се филтрираат податоци

5 *Полиња за колони* – полиња чии вредности ќе се прикажат во колони

6 *Полиња за редови* – полиња чии вредности ќе се прикажат во редови

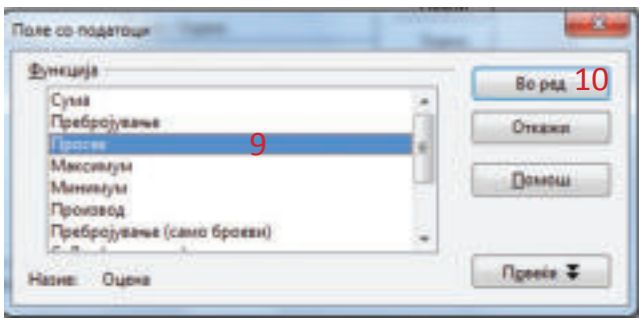
7 *Полиња со податоци* – полиња чии вредности ќе се прикажат како податоци во табелата.

За вредностите во полињата со податоци ќе се прикаже и збирот, средната вредност или вредност пресметана според некоја друга функција. Функцијата може да се смени со:

8 два пати кликнување на името на полето во *Полиња со податоци*, по што се добива прозорецот *Поле со податоци*, во кој:

9 во листата се кликнува на саканата функција и

10 се кликнува на копчето *Во ред*;



Сл. 5. 61 Прозорецот Поле со податоци

Во прозорецот *Пилот за податоци* (сл. 5.60)

11 се кликнува на копчето *Повеќе* по што се отвора долниот дел на прозорецот.

12 За пивот табелата да се прикаже во нов лист, од паѓачката листа *Резултат во* се избира *нов лист*;

13 На крај се кликнува на копчето *Во ред*

14 по што се добива извештајот.

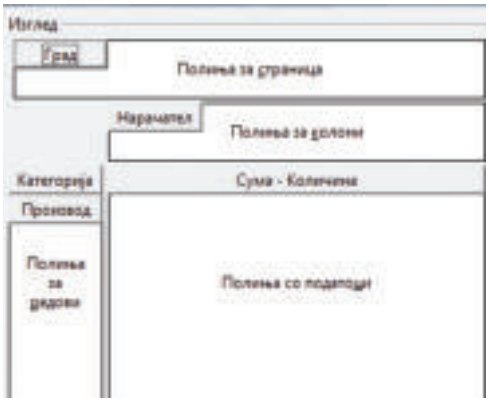
	A	B	C	D	E
1	Филтер				
2	Време на полагање	- сите -	14		
3					
4	Просек - Поени	Курс			
5	Кандидат	excel	windows	word	Вкупно Резултат
6	Ана Мицевска	76	90	88	84,67
7	Борче Костов	100	98	98	98,67
8	Јане Спировски	50	69	72	60,25
9	Јована Арсовска	53	74	83	65,75
10	Маја Андова	92	85	78	85
11	Марко Симиќ	82	92	54	70,5
12	Мите Гоцев	64	68	72	68
13	Селим Османи	68	87	92	82,33
14	Вкупно Резултат	68,8	82,88	76,78	75,63

Сл. 5. 62 Извештај во вид на пивот табела

Чекор по чекор:

1. Отвори ја работната книга prodazba na tehnicka oprema;
2. Кликни каде било во табелата;

3. Повикај ја наредбата *Податоци→Пилот за податоци→Стартувај..*;
4. Во прозорецот *Избирање извор*, потврди *Тековен избор* и кликни на копчето *Во ред*;
5. Во прозорецот *Пилот за податоци* повлечи ги колоните во полињата како што е покажано на следната слика:

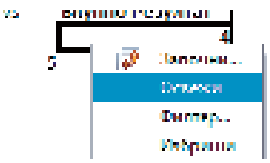


6. Ќе добиеш табела како на сликата долу:

Град	сите						
Сума - Количина		Наредник					
Категорија	Производ	HP Compaq	HP Mobile	CC Compaq	DD Compaq	AA Compaq	AA Mobile	Вкупно Резултат
Компјутер	Fujitsu			12				12
	Конфигурација DELL				1			1
	Конфигурација Fujitsu				5			5
	Конфигурација Lenovo	18						18
	Конфигурација Siemens					5		5
Мобилен т	Motorola						11	11
	Nokia						14	14
	Samsung		1					1
Монитор	Phillips					9		9
	Sony	11						11
Печатач	Epson				21			21
	HP		6					6
Телевизор	LG				12			12
	Sony					3		3
Вкупно Резултат		29	7	12	39	17	22	126

Освежување на пилот табела

Често пати се случува откако ќе се креира пивот табела изворните податоци да се изменат. Тогаш е потребно пивот табелата да се освежи. Се кликнува со десното копче каде било во пивот табелата и од паѓачкото мени се избира *Освежи*.



Резиме:

Изведени или *пивот табели* се користат за попрегледен распоред на податоци од една или повеќе табели.

MS Excel:

За да се креира пивот табела, на картичката *Insert* се кликнува на копчето *Pivot Table*, по што се добива прозорецот *Create Pivot Table*. Со едноставно повлекување на колоните во соодветни полиња се добива анализа на податоците по различни критериуми.

Calc:
За да се креира пивот табела, се повикува наредбата *Податоци→Пилот за податоци→Стартувај...* Во прозорецот *Пилот за податоци* во средишниот дел се прикажани имиња на полињата од базата кои се повлекуваат десно до соодветните полиња: *Полиња за страница, Полиња за колони, Полиња за редови* и *Полиња со податоци.*

Вештини што треба да ги усовршиш:
Да креираш различни извештаи во вид на пивот табела.

Прашања:

1. Што е изведена или пивот табела?
2. Како се креира пивот табела?
3. Зошто се погодни извештаите во вид на пивот табела?

Задачи:

1. Креирај база на податоци за освоени награди на натпревари за средните училишта за последните 3 години со следните полиња: Година, Натпревар по предмет, Училиште, Ученик, Освоено место, Освоени поени.
   - Креирај извештај за вкупно освоени поени по училиште;
   - Креирај извештај за вкупно освоени поени по натпревар;
   - Креирај извештај за вкупно освоени поени по година.

5.6 Заштита и валидација на податоци

5.6.1 Заштита на податоци

Податоците во табелите можат да се заштитат од случајно или намерно менување од страна на други корисници. При тоа може да се заштити цела работна книга, одредени работни листови или само одредени ќелии на работниот лист со или без лозинка.

Отклучување на ќелии во MS Excel

Кога ќе се отвори нов документ сите ќелии се заклучени. Тоа значи дека сите ќелии ќе бидат заштитени кога ќе се примени заштита на работен лист или на работна книга. За да се дозволи менување на содржина на некои ќелии, тие ќелии треба да се отклучат пред да се примени заштита на работен лист или на работна книга.

Забелешка:
Ќелиите се заклучуваат или отклучуваат според потреба. Се заклучуваат оние ќелии чија содржина се заштитува од менување, а тоа се најчесто ќелии кои содржат формули и функции. Исто така, може да се сокрие изгледот на формули и функции во некои ќелии.

За да се отклучат ќелии:

- се означуваат ќелии или опсег на ќелии,
- на картичката *Home*, во групата *Cells*, се кликнува на копчето *Format*,
- во паѓачката листа се кликнува на копчето *Locked* за да се деактивира.

Сокривање на формули и функции во MS Excel

За да се сокријат формули или функции во одредени ќелии:

1 се означуваат ќелии или опсег на ќелии,

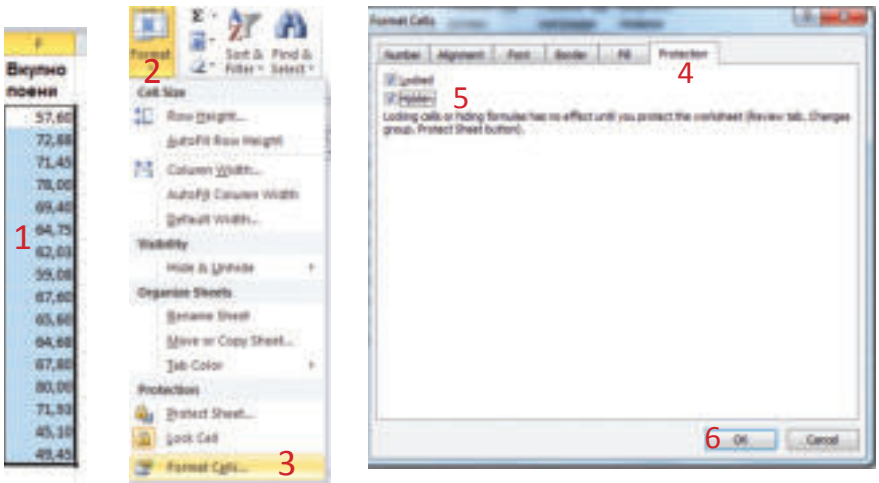
2 на картичката *Home*, во групата *Cells*, се кликнува на копчето *Format*,

3 во паѓачката листа се кликнува на копчето *Format Cells*,

4 во прозорецот *Format Cells* се кликнува на картичката *Protection*,

5 се потврдува полето за потврда *Hide*,

6 се кликнува на копчето *OK*.



Сл. 5. 63 Сокривање на формула или функција

Забелешка:

Во овој прозорец, исто така, можат да се отклучат или заклучат ќелии.

Важно!

Заштита на заклучени ќелии и на ќелии со сокриени формули нема да се примени доколку не се заклучи работниот лист.

Заштита на работен лист во MS Excel

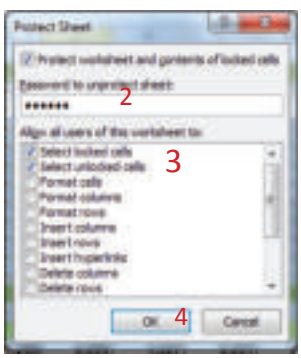
За да се заштити работен лист:

1 во картичката *Review*, во групата *Change*, се кликнува на копчето *Protect Sheet*,

2 во прозорецот *Protect Sheet*, во полето *Password to unprotect sheet* се пишува лозинката,

3 во листата *Allow all users of this worksheet to* се избираат акции кои ќе им бидат дозволени на сите корисници да ги извршат,

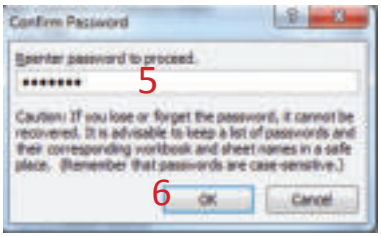
4 се кликнува на копчето *OK*.



Сл. 5. 64 Заштита на работен лист

5 Лозинката се потврдува во прозорецот *Confirm password*

6 и се кликнува на копчето *ОК*.



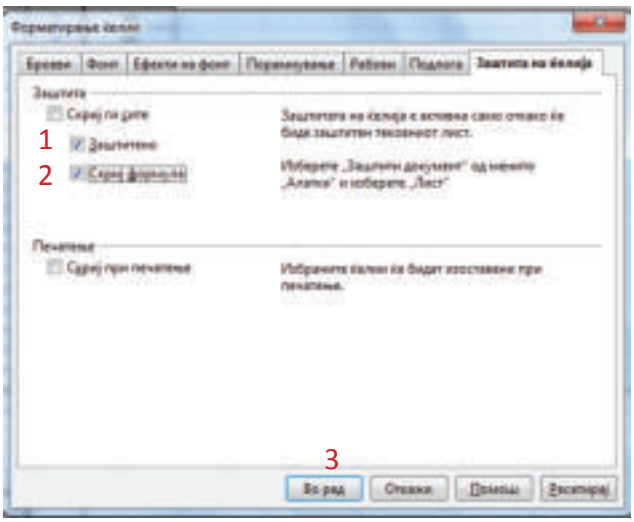
Сл. 5. 65 Внесување и потврдување на лозинка

Забелешка:

Лозинка не мора да се постави, но тогаш секој ќе може да ја отстрани заштитата на работниот лист и да ги смени податоците.

Заштита на ќелии и сокривање на формули во Calc

За да се заштитат ќелии и/или да се сокрие формула, ќелиите или опсегот на ќелии се означуваат и се повикува наредбата *Форматирај→Ќелии* по што се појавува прозорецот *Форматирање ќелии*.



Сл. 5. 66 Прозорец Форматирање ќелии

Во прозорецот *Форматирање ќелии*, на картичката *Заштита на ќелија*:

1 се потврдува опцијата *Заштитени* за ќелиите да се заштитат од менување на содржина и/или

2 се потврдува опцијата *Скриј формула* за формула да не се гледа во ќелиите.

3 На крај се кликнува на копчето *Во ред.*

Важно:

Заштита на ќелии и сокривање на формули нема да се примени доколку не се заштити работниот лист.

Заштита на работен лист во Calc

За да се заштити работен лист се повикува наредбата *Алатки→Заштити документ→Лист...*, по што се појавува прозорецот Заштита на лист, во кој

1 во полето *Лозинка* се внесува лозинка,

2 во полето *Потврда* се потврдува лозинката

и

3 се кликнува на копчето *ОК*.

Сл. 5. 67 Внесување и потврда на лозинка

На заштитен работен лист заштитата се отстранува со истата наредба *Алатки→Заштити документ→Лист...*

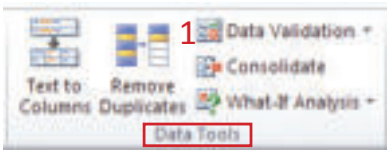
5.6.2 Валидација на податоци

Валидација на податоци значи проверка на податоците кои се внесуваат во ќелии. Проверката може да се однесува на тип на податоци, на пр. смее да се внесе само нумерички податок, или на вредности, на пр. само позитивни броеви, одреден опсег на датуми и сл. Контрола на внесување на податоци може да се изврши и со паѓачки листи со што внесувањето на податоци се ограничува со избор од листата.

Валидација на податоци во MS Excel

Валидација на податоци се поставува на следниот начин:

Се означуваат ќелии за кои се поставува валидација,



1 потоа на картичката *Data*, во групата *Data Tools*, се кликнува на копчето *Data Validation*;

2 Во дијалог прозорецот *Data Validation* се кликнува на картичката *Setting*, потоа

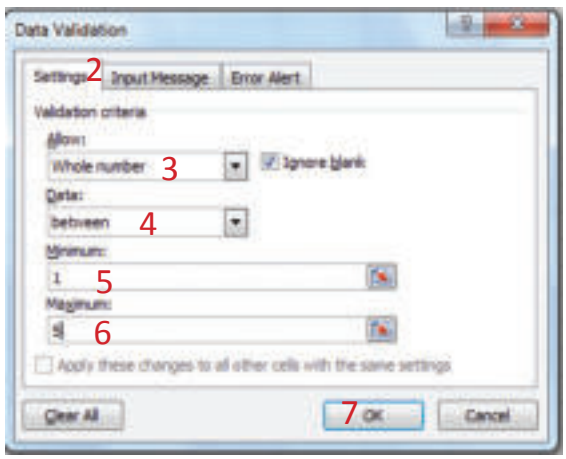
3 во листата *Allow* се избира тип на податоци кој може да биде цел број, децимален број, листа, датум време, текст.

4 во листата *Data* се избира оператор, на пр. помало, поголемо, помеѓу итн.

Зависно од изборот на операторот се појавуваат полињата

5 *Minimum* и/или

6 *Maximum* во кои се пишува соодветна вредност,



Сл. 5. 68 Прозорец за валидација на податоци

7 на крај се кликнува на копчето *OK*.

Важно!

Валидација на податоци не може да се постави доколку работната тетратка е заштитена.

Чекор по чекор:

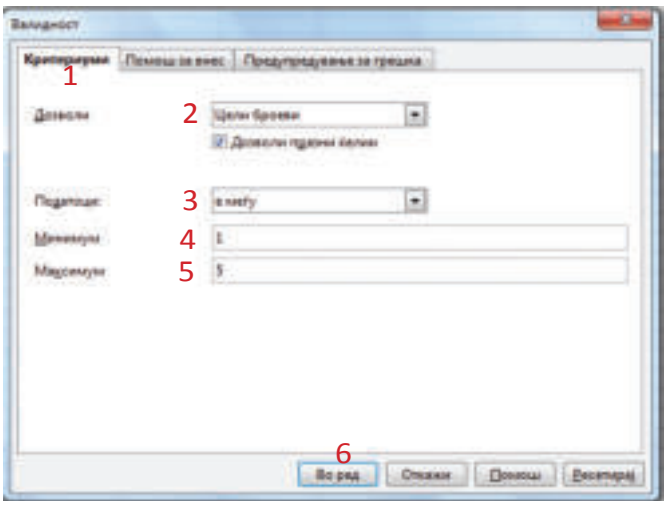
1. Отвори нова работна книга;
2. Работниот лист Sheet1 преименувај го со име Оцени;
3. Во работниот лист Оцени креирај база на податоци со следните полиња: Име, Презиме и Оцена;
4. Означи ја колоната Оцена;
5. На картичката *Data*, во групата *Data Tools*, кликни на копчето *Data Validation*;
6. Во дијалог прозорецот *Data Validation* кликни на картичката *Setting*;
7. Во листата *Allow* избери тип *Whole Number* (цели броеви);
8. Во листата *Data* избери *between* (помеѓу);
9. Во полето *Minimum* напиши 1;

10. Во полето *Maximum* напиши 5;
11. Внеси податоци во табелата, обиди се да внесеш невалиден податок за оцена;
12. Зачувај ја работната книга со име Валидација.

Валидација на податоци во Calc

Валидација на податоци се поставува на следниот начин:

Се означуваат ќелии за кои се поставува валидација, потоа се повикува наредбата *Податоци→Валидност...*;



Сл. 5. 69 Прозорец за валидација на податоци

1 Во дијалог прозорецот *Валидност* се кликнува на картичката *Критериуми*, потоа

2 во листата *Дозволи* се избира тип на податоци кој може да биде цел број, децимален број, листа, датум време, текст итн.

3 во листата *Податоци* се избира оператор, на пр. помало, поголемо, помеѓу итн.

Зависно од изборот на операторот се појавуваат полињата

4 *Минимум* и/или

5 *Максимум*

во кои се пишува соодветна вредност,

6 на крај се кликнува на копчето *Во ред*.

Важно!

Валидација на податоци не може да се постави доколку работната тетратка е заштитена.

Чекор по чекор:

1. Отвори нова работна книга;
2. Работниот лист Лист1 преименувај го со име Оцени;
3. Во работниот лист Оцени креирај база на податоци со следните полиња: Име, Презиме и Оцена;
4. Означи ја колоната Оцена;

5. Повикај ја наредбата *Податоци→Валидност...*;
6. Во дијалог прозорецот *Валидност* кликни на картичката *Критериуми*;
7. Во листата *Дозволи* избери *Цели броеви*;
8. Во листата Податоци избери *е меѓу;*
9. Во полето *Минимум* напиши 1;
10. Во полето *Максимум* напиши 5;
11. Внеси податоци во табелата, обиди се да внесеш невалиден податок за оцена;
12. Зачувај ја работната книга со име Валидација!.

КОМПЈУТЕРСКИ МРЕЖИ И ИНТЕРНЕТ

Клучни зборови

- Компјутерска мрежа
- Работни станици
- Peer-to-Peer мрежно работење
- Клиент/сервер мрежно работење
- Периферни уреди во мрежа
- Хаб
- Рутер
- Интернет
- WWW (Веб)
- HTTP
- URL
- Пребарувач
- Веб 2.0
- Семантички веб
- Социјална мрежа
- Веб социјална мрежа
- Приватност и заштита на личните податоци

6.1 Компјутерски мрежи

Веднаш по појава на првите компјутери се размислувало за комуникација помеѓу компјутерите, односно за размена на податоци без користење на надворешни мемории, туку со директна врска (on line). За оваа цел компјутерите се поврзуваат во компјутерски мрежи.

Компјутерска мрежа претставува збир од два или повеќе компјутери кои се поврзани преку комуникациски медиум и кои меѓусебно можат да комуницираат и да делат ресурси. Мрежите се користат за пренос како на дигитални така и на аналогни податоци кои мора да бидат прилагодени кон соодветни системи за пренос. Податоците можат да се пренесуваат во реално време (говор, видео и сл.) или во одложено време (електронска пошта, пренос на датотеки и сл.)

Компјутерите се поврзуваат во компјутерски мрежи за:

- заедничко користење на софтверски (апликации, игри и сл.) и на хардверски ресурси (дискови, печатачи, скенери и други уреди),
- заедничко користење на датотеки (документи, слики, музика итн.),
- размена на податоци и комуникација помеѓу корисниците,
- заедничка работа на корисниците на некои проекти.

6.1.1 Видови на компјутерски мрежи

Поделба според големината и распространетост

Според големината и функцијата која ја имаат, компјутерските мрежи се поделени во три основни групи:

- локални мрежи LAN (Local Area Networks)
- регионални (градски) мрежи MAN (Metropolitan Area Networks)
- глобални мрежи WAN (Wide-Area Networks).

Локални мрежи

Локална компјутерска мрежа претставува основа на секоја мрежа. Оваа мрежа е просторно ограничена – поврзаните компјутери се наоѓаат на релативно мал простор, како што е зграда или комплекс од згради. Во локалните мрежи секој компјутер има можност да пристапи до податоци и уреди на некој друг компјутер во мрежата. На овој начин повеќе компјутери можат да користат скапа опрема како што се ласерски печатачи или плотери, како и самите податоци. Корисникот на еден компјутер одлучува кои ресурси ќе се делат во мрежата.

Како преносни медиуми кај локалните мрежи се користат кабли или безжична врска кога се работи за безжична локална мрежа WLAN (Wireless Local Area Network). Безжични локални мрежи имаат помала брзина на пренос на податоци но обезбедуваат поголема мобилност на корисниците.

Регионални мрежи

Во регионални компјутерски мрежи се поврзуваат компјутери кои се наоѓаат на територија на еден град или регион. Како преносен медиум се користи некој од расположиви видови на јавен пренос (телефонски линии, сателитски врски и оптички врски).

Глобални мрежи

Во глобална компјутерска мрежа се поврзуваат компјутерите од целиот свет. Оваа мрежа, всушност, претставува мрежа од повеќе локални и регионални мрежи и овозможува комуникација помеѓу мрежи кои не се на ист географски простор.

Интернетот е најдобриот пример за овој вид на мрежи иако за него повеќе се користи терминот *глобална мрежа*. Глобалната мрежа ги поврзува сите мрежи во една целина која овозможува користење на податоци и ресурси низ целата планета.

Поделба според начинот на поврзување

Во повеќето мрежи се користат кабли за поврзување на уредите, притоа тие можат да бидат поврзани со единствен кабел кој ги поврзува сите уреди или пак каблите да го поврзуваат секој уред со централна локација. Каблите се направени од бакарни жици, а се користат и други видови на медиуми како што се стакло или пластика.

Набрзо по појавувањето на преносните сметачи (notebook), повеќето корисници имале потреба од безжично поврзување кон Интернет или локалните мрежи. Сметачите се опремуваат со радио примопредаватели со краток досег. За безжичен пренос се користат радиобранови и микробранов пренос.

Според начинот на поврзување, компјутерските мрежи се делат на:

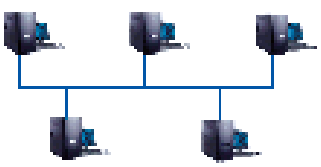
- жичани компјутерски мрежи и
- безжични компјутерски мрежи.

6.1.2 Топологија на компјутерските мрежи

Топологија претставува геометриски распоред на компјутерите во мрежата. Најчестите топологии се топологија на заедничка магистрала, топологија на ѕвезда и топологија на прстен.

Топологија на заедничка магистрала

Топологија на заедничка магистрала поврзува компјутери во една линија со еден заеднички кабел. Предностите на оваа топологија се во тоа што лесно може да се додаде нов компјутер во мрежата и што бара помалку кабли. Недостатоците се во тоа што целата мрежа ќе се исклучи доколку дојде до прекин во кабелот и тешко е да се идентификува каде настанал проблемот. Ова не е добро решение за поврзување на повеќе компјутери.



Сл. 6. 1 Топологија на заедничка магистрала

Топологија на ѕвезда

Во топологија на ѕвезда се користи централна компонента која овозможува поврзување на компјутери заради меѓусебна комуникација. Централната компонента најчесто е разводна кутија (hub), или комутатор (switch). Предноста на топологијата на ѕвезда е што одговорноста се фокусира на централната компонента, а недостатокот е што се потребни многу кабли за поврзување. Расипувањето на централната компонента доведува до прекинување на работата на мрежата. Ако се расипе еден компјутер останатите компјутери нормално ќе работат во мрежата.

Сл. 6. 2 Топологија на ѕвезда

Топологија на прстен

Кај топологија на прстен компјутерите се поврзуваат во круг со еден кабел. Секој компјутер комуницира директно и единствено со соседните компјутери. Мрежа во форма на прстен може да биде еднонасочна и двонасочна. Во еднонасочна мрежа секој компјутер комуницира само со еден сосед, додека во двонасочна мрежа секој компјутер комуницира со двата соседи.

Сл. 6. 3 Топологија на прстен

6.1.3 Архитектура на компјутерските мрежи

Компјутерите можат да се поврзат според две основни архитектури: мрежа од рамноправни компјутери (peer-to-peer) и мрежи базирани на сервери (клиент-сервер архитектура).

Мрежа на рамноправни компјутери

Во мрежа на рамноправни компјутери (peer-to-peer, или P2P), сите компјутери имаат еднакви можности и одговорности, односно секој компјутер може да биде и клиент и сервер. Не постои главен компјутер кој ги надгледува останатите компјутери и кој ќе контролира како податоците се делат во мрежата. Компјутерите се поврзани меѓу себе со цел да делат датотеки, ресурси и пристап до Интернет. Ова е практично за поврзување на помал број на компјутери, најчесто во домашни услови, каде секој компјутер работи како независна *работна станица* која чува податоци на сопствен хард диск, но податоците може и да ги дели со сите останати компјутери во мрежата.

Сл. 6. 4 Мрежа на рамноправни компјутери

Корисникот на секој компјутер одлучува кои податоци од неговиот компјутер можат да се делат во мрежата. Мрежи од овој вид се нарекуваат и *работни групи* (Work-group). Софтверот за мрежи со рамноправни компјутери е вграден во поголемиот дел на модерните оперативни системи, како што се Windows и Mac OS.

Мрежи базирани на сервери

Кај овие мрежи еден или повеќе јазли имаат улога на сервер – компјутер кој ги опслужува сите други компјутери кои имаат улога на клиенти. Клиентите обично се активни корисници кои праќаат барања и чекаат додека тие не се исполнат. Серверот е пасивен, тој чека на барања кои ги исполнува и ги праќа назад до корисникот. Сервер може да биде било кој компјутер, но тоа се обично моќни машини со добри конфигурации и карактеристики од причина што тие во исто време треба да опслужат многу клиенти и целата комуникација се одвива преку него. Тоа можат да бидат персонални компјутери со големи можности па сè до големи централни компјутери.

Сл. 6. 5 Клиент-сервер мрежа

Клиентите со серверот се поврзани преку локална или регионална компјутерска мрежа. Клиент-сервер мрежите покрај основниот оперативен систем бараат и дополнителен мрежен оперативен систем (NOS – Network Operating System).

6.1.4 Вмрежување на персоналните компјутери

Персоналните компјутери (десктоп и преносни) можат да работат самостојно или во мрежа поврзани со други компјутери.

Компјутерот кој работи самостојно мора на својот диск да ги има инсталирано сите апликации потребни на корисникот. Исто така сите податоци треба да бидат зачувани на дискот или на некоја друга надворешна меморија. Доколку има потреба од користење на дополнителни периферискиединици, како што се печатач, скенер, звучник и други, тие мора да бидат поврзани на самиот компјутер.

Компјутерот кој е поврзан во мрежа се нарекува *работна станица*. Кога се работи со вмрежен компјутер, можат да се разменуваат податоци и информации со други компјутери во вид на пораки и датотеки, или може да се делат периферискиединици и да се користат мрежни апликации.

Периферни уреди и додатна опрема во мрежа

Мрежите, особено домашните, можат да бидат многу пософистицирани бидејќи постојат многу други додатни и периферни уреди кои можат да се приклучат во мрежа. На сликата десно е даден пример на една добро опремена домашна мрежа.

Сл. 6. 6 Пример на домашна компјутерска мрежа

Во оваа мрежа централното место го зазема безжичен рутер кој го поврзува десктоп компјутерот со серверот на мрежата. Освен компјутерот, во мрежата се поврзани и лаптоп, печатач, таблет, смартфон, читач на книги и опрема за следење на ТВ и музички програми преку Интернет.

Поврзувањето главно е безжично, но се користат и кабли за поврзување на десктоп компјутер со рутер, со печатач и со сервер. Рутерот сите единици ги поврзува една со друга и со Интернет.

Резиме

Компјутерска мрежа претставува збир од два или повеќе компјутери кои се поврзани преку комуникациски медиум и кои меѓусебно можат да комуницираат и да делат ресурси.

Според големината и функцијата која ја имаат компјутерските мрежи се поделени во три основни групи: *локална мрежа, регионална* и *глобална мрежа*.

Според начинот на поврзување, компјутерските мрежи се делат на *жичани компјутерски мрежи* и *безжични компјутерски мрежи*.

Топологија претставува геометриски распоред на компјутерите во мрежата. Најчестите топологии се *топологија на заедничка магистрала, топологија на ѕвезда* и *топологија на прстен*.

Компјутерите можат да се поврзат според две основни архитектури: *мрежа од рамноправни компјутери* и *мрежи базирани на сервери*.

Персоналните компјутери можат да работат самостојно или во мрежа поврзани со други компјутери. Компјутер кој е поврзан во мрежа се нарекува *работна станица*. Во мрежите, особено домашните, постојат многу други периферни единици кои можат да се приклучат во мрежа.

Прашања:

1. Што е компјутерска мрежа? Колку најмалку компјутери ја сочинуваат мрежата?
2. Кои се придобивките од поврзување на компјутерите во мрежа?
3. Што е LAN, а што WAN?
4. Ако компјутерите само физички се поврзани, подразбира ли тоа дека податоците на нив се достапни на сите компјутери во мрежата ?
5. Што е топологија на компјутерска мрежа?
6. Кои се предностите, а кои се недостатоците на топологија со заедничка магистрала?
7. Кои се предностите, а кои се недостатоците на топологија на ѕвезда?
8. Во топологија на прстен, еден компјутер со кои други компјутери може да комуницира?
9. Кои се две основни архитектури на компјутерските мрежи?
10. Дали мрежа на рамноправни компјутери е погодна за WAN мрежите?
11. Во клиент-сервер мрежата, од што зависи дали компјутерот ќе биде клиент или сервер?
12. Што е работна станица?
13. Направи споредба помеѓу компјутер што работи самостојно и работна станица!
14. Освен компјутерите, кои други додатни и периферни уреди можат да бидат поврзани во мрежа?

6.2 Мрежна дистрибуција

За да се поврзат компјутери и други единици (печатач, скенер, мобилен телефон итн.) во компјутерска мрежа и во неа да се овозможи дистрибуција на податоци потребни се:

- комуникациски уреди
- комуникациски канали (медиуми за пренос на податоци)
- соодветен мрежен софтвер.

6.2.1 Комуникациски уреди

Компјутерските мрежи користат различни видови на уреди за комуникација преку кои компјутерите се поврзуваат на комуникациски медиум.

Мрежната картичка е уред кој го поврзува компјутерот со локалната мрежа. Улогата на мрежната картичка е да претвора дигиталните сигнали од компјутерот во сигнал погоден за пренос преку мрежата и обратно. Тоа е електронска картичка која се вградува во компјутерот и на која има приклучни точки (порти) на кои можат да се прикачат мрежни кабли.

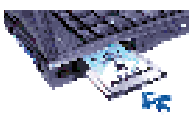
Сл. 6. 7 Мрежни картички

Постојат и безжични мрежни картички кои наместо порти за кабли имаат антени за сигнал кој се пренесува по безжичен пат. Мрежната картичка често се нарекува и NIC (Network Interface Card), мрежен интерфејс или мрежен адаптер.

Модем е уред кој дозволува корисниците да праќаат дигитални информации преку аналогните средства за комуникација, како што се телефонските врски. Тој врши конверзија на дигитални сигнали во аналогни сигнали (модулација) и обратно (демодулација). Преку модемот компјутерите се поврзуваат на глобалната мрежа – Интернет. Модемот може да биде екстерен кој од надвор се приклучува на компјутерот и интерен, кој се вградува во компјутерот во вид на картичка. Во поново време се користи USB модем.

Сл. 6. 8 Модеми: екстерен, интерен и USB

PC картичка (PC Card) може да има мрежен адаптер, модем или и едно и друго. Обично се користи кај преносливите компјутери.



Сл. 6. 9 PC картичка

Хаб (hub) и свич (switch) се уреди преку кои група на корисници се поврзуваат и им се овозможува работа во мрежа. Обично се користат како централна компонента во локалните мрежи.

Хаб е, всушност, разводна кутија на која постојат повеќе конектори. На секој конектор се поврзува по еден кабел преку кој се поврзува еден компјутер или сервер. Секој сигнал кој пристигнува преку еден кабел тој го умножува и го праќа понатаму преку повеќе кабли до останатите уреди во мрежата.

Сл. 6. 10 Хаб

Свич или склопка (комутатор) е уред сличен на хаб кој исто така има повеќе конектори на кои се поврзува по еден кабел преку кој се поврзува еден компјутер или сервер. Се разликува од хаб по тоа што сигналот кој ќе го добие не го праќа до сите уреди туку само до уред за кој тој сигнал е наменет.

Сл. 6. 11 Свич

Препорака:

Погледни ја разликата помеѓу хаб и свич на следнава адреса: http://www.direct-systems.com/support/switchvshub.php

Рутер (router) е уред за насочување на податоци на нивниот пат низ мрежата. Тој овозможува повеќе локални мрежи да се поврзат во една мрежа. Кога рутер ќе добие сигнал упатен до некој компјутер, тој сигналот понатаму го праќа до мрежа во која бараниот компјутер е поврзан. Ако не може директно да пристапи до саканата мрежа, сигналот го праќа до друг рутер кој е на патот до мрежата. Рутери можат да бидат жични и безжични.

Сл. 6. 12 Рутер

6.2.2 Комуникациски канали

Комуникациски канал е медиум преку кој се пренесуваат информации. Тоа може да биде кабел кој претставува физичка врска помеѓу два компјутери – краевите на кабелот се поврзуваат со мрежните картички. Компјутерите можат да се поврзат и со безжичен пат – тогаш се потребени приемник и антена.

Од медиумот зависи брзината на пренос на податоци како и најголемата можна оддалеченост помеѓу два уреди. Брзината на пренос се мери со бројот на пренесени битови во секунда (bps – bit per second) или со поголемите единици kbps, Mbps итн.

Жичен пренос

Кабловските врски овозможуваат најголемата безбедност и најголемата брзина на пренос. Постојат три вида на жични канали: вплетени кабли, коаксијални кабли и оптички кабли.

Вплетените кабли (twisted wire pairs) се состојат од две изолирани жици вплетени една со друга. Чувствителни се на надворешни влијанија, па често се користат повеќе парови жици со што се намалува мешање на електрични сигнали кои патуваат низ кабелот и оние од надворешните извори. Парови на жици понекогаш се обвиткуваат со метална обвивка со што дополнително се намалува влијание од надворешните извори. Ваквите кабли првобитно, а и денес, се користат за телефонските комуникации. Вплетените кабли се ефтини и релативно бавни.

Сл. 6. 13 Вплетени кабли

Коаксијалните кабли се состојат од бакарна жица во средина, завиткани со изолациски материјал, а сето тоа е обвиткано со проводник кој штити од надворешните електрични влијанија. Коаксијалните кабли можат да бидат тенки и дебели. Се користат за поврзување на компјутери и други уреди на поголеми далечини (до неколку стотини метри) и обично имаат поголема ширина на канали од вплетените кабли и со тоа имаат и побрз пренос на податоци.

Сл. 6. 14 Коаксијални кабли

Оптичките кабли се прават од стаклени или пластични влакна преку кои се пренесува светлосен сигнал кој го генерира ласер или LED диода (light-emitting diode). Околу влакната се наоѓа обвивка која рефлектира светлина. Ова е најсигурен вид на пренос на податоци. Други видови медиуми лесно можат да се „прислушкуваат“ додека со оптичките кабли тоа не е можно. Важната особина на оптичките кабли е и тоа што светлосните сигнали кои патуваат преку него се имуни на електрични и магнетни влијанија од надвор. Преносот на податоци преку оптичките кабли е многу брз.

Сл. 6. 15 Оптички кабли

Безжичен пренос

Безжична врска се користи кога не е можно да се поврзе кабел или кога компјутерите се на голема оддалеченост и поврзување со кабли би било скапо. Најчесто користени безжични типови на пренос се: микробранови, инфрацрвени и радио бранови.

Микробранови се медиум со кој податоци и информации се праќаат преку воздух во вид на високофреквентни радио сигнали. Микробранови сигнали се движат по права линија – помеѓу предавател и приемник не смее да има физичка пречка, па во микробранови системи на Земјата се поставуваат репетитори на сигналите. Во сателитските микробранови системи комуникација се одвива со праќање на радио сигнали од радио станица на Земјата до сателит во Земјината орбита, од каде тие понатаму се праќаат до други станици на Земјата. На микробрановите сигнали влијаат лоши временски услови што може да доведе до послаб прием на сигналот.

Сл. 6. 16 Интернет преку сателит

Инфрацрвените зраци и порано се користеле кај далечинските управувачи на ТВ приемници, видео или стерео уреди. Сега се користат и за безжични локални мрежи. Инфрацрвените сигнали патуваат по права линија. Рефлектори на плафонот ги одбиваат сигналите кои ги праќа еден компјутер до приемник кој се наоѓа во друг компјутер. Лошата страна на ваков вид на пренос е небезбедност затоа што не може да се контролира кој се може да го фати сигналот, освен ако тој не е криптиран.

Пренос преку *радио сигнали* е кога пренос на податоци и информации се одвива преку фреквенции изнајмени од јавните радио мрежи. Денес преносните (laptop) компјутери користат радио примопредаватели за комуникација со локална компјутерска мрежа.

6.2.3 Мрежен софтвер

Збир на програмите кои поддржуваат работа во мрежа се нарекува комуникациски софтвер. Него го сочинуваат протоколи (правила според кои се врши комуникација во мрежа) и оперативни системи кои се во директна комуникација со хардверот на компјутерот и имаат поддршка за комуникацискиот хардвер.

Мрежниот софтвер ги обезбедува следните функции:

- поставување на параметрите за работа во мрежа,
- вклучување на компјутерот во мрежа,
- работа на корисниците во мрежа,
- сигурносните мерки,
- административните работи и помош на корисникот.

6.2.4 Мрежни додатоци

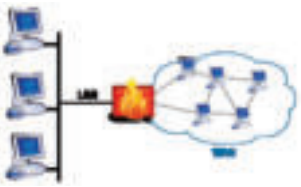
Давател на интернет услуги

Суштината на поврзување на интернет е поврзување со некој рутер кој е дел од интернетот. Таков рутер најчесто имаат универзитетите, истражувачките центри или комерцијалните компании кои се нарекуваат даватели на интернет услуги или *интернет провајдери* (ISP – Internet Service Providers). Тие функционираат на различни

нивоа на комуникации. Локален давател на интернет услуги обезбедува поврзување до некој од националните или регионалните даватели на интернет услуги кои имаат голем капацитет и поседуваат сопствена мрежа.

Огнен ѕид (Firewall)

Огнениот ѕид е хардверски уред или софтвер кој се поставува помеѓу локална мрежа и интернет. Намената на огнениот ѕид е да ги штити податоците во мрежата од неавторизираните корисници со блокирање и забрана на пристап според правила кои корисникот ќе ги дефинира.



Сл. 6. 17 Огнениот ѕид

Со поставување на огнен ѕид помеѓу два сегмента во мрежа можат да се контролираат права на пристап на поедини корисници до поедини делови на мрежата.

Резиме

За да се поврзат компјутери и други ресурси во компјутерска мрежа потребни се: *комуникациски уреди, комуникациски канали* и соодветен *мрежен софтвер*.

Компјутерските мрежи користат различни видови на уреди за комуникација: мрежните картички, PC картички, модем, хаб, свич и рутер.

Компјутерите во мрежа можат да се поврзат со кабли или по безжичен пат. За поврзување со кабли се користат вплетени, коаксијални и оптички кабли. Најчесто користените бежични типови на пренос се: микробранови, инфрацрвени и радио бранови.

Програми кои подржуваат работа во мрежа се нарекуваат *комуникациски софтвер*. Огнен ѕид е хардверски уред или софтвер кој ги штити податоците во мрежата од неавторизираните корисници.

Прашања:

1. Што е потребно за мрежна дистрибуција?
2. Наброј уреди за пренос на податоци!
3. Која е улогата на мрежната картичка?
4. Какви можат да бидат мрежните картички?
5. Што е PC картичка и каде се користи?
6. Што е модем? Која е неговата улога во компјутерските мрежи?
7. Какви видови модеми постојат?
8. За што се користат хаб и свич?
9. Која е разлика помеѓу хаб и свич?
10. Што е рутер? Која улога ја има рутерот во мрежата?
11. Какви видови на рутери постојат?
12. Што се комуникациски канали и какви тие можат да бидат?
13. Од што зависи брзината на пренос на податоци во мрежите?
14. Кој пренос на податоци е побезбеден и побрз, жичен или безжичен?
15. Со што се мери брзината на пренос на податоци?
16. За колку секунди се пренесува 30 MB со брзината на пренос од 2 Mbps ?
17. Наброј медиуми за пренос на податоци!
18. Какви кабли се користат за мрежна дистрибуција?

19. Кои кабли се најбезбедни?
20. Кои се најчесто користени безжични типови на пренос?
21. Во кој тип на мрежа се користат микробранови, а во кој тип инфрацрвените зраци?
22. Како се користат радиобранови во компјутерските мрежи?

6.3 Интернет

Интернет е глобална мрежа во која се поврзани милиони компјутерски мрежи од целиот свет. Содржините на Интернет се:

- електронска пошта (E-mail)
- разговори во реално време (IRC Internet Relay Chat)
- протокол за пренос на датотеките (FTP File Transfer Protocol)
- WWW – World Wide Web и други.

6.3.1 Историја и развој на Интернет

Историјата на Интернет започнува кон крајот на 50-тите години од минатиот век кога почнало да се размислува за комуникациска мрежа во која би се поврзале повеќе компјутери и би разменувале податоци меѓу себе. Првата компјутерска мрежа, наречена ARPANet е создадена во 1969. година во рамки на истражувачките проекти на американскиот секретаријат на одбрана. Оваа мрежа е осмислена како воен проект, но набргу мрежата почнала да се користи и за мирнодопски намени. Таа ги поврзувала американските научни и академски институции и на неа биле поврзани четири компјутери.

За љубопитните:

Првата ARPANet врска е воспоставена на 29. октомври 1969 год. помеѓу IMP на UCLA и SRI. Во декември 1969 год. сите четири компјутери биле меѓусебно поврзани.

Придобивките од размена на податоци во ова мрежа биле јасни па се продолжило со работа на овој проект. Во седумдесетите години веќе се поврзани 15 јазли и 23 компјутери, а 1973 година е поврзан и првиот јазол надвор од границите на САД. Воведен е и нов мрежен протокол NCP (Network Control Protocol), кој овозможувал полесно вмрежување. За овој период се врзува и појава на e-mail, mailig листа и news групите.

ARPANet 1983. година се раздвојува на воената мрежа MILNet (MILitary Network) и на ARPANet. Во разни држави започнуваат да се развиваат академски и комерцијални мрежи. Во осумдесеттите години од дотогашниот NCP протокол се поминува на нов протокол за контрола на преносот (TCP/IP – Transmission Control Protocol/Internet Protocol). Во ова време почнува да се користи терминот Интернет кој се однесува на мрежа која користи TCP/IP протокол. На крајот на осумдесеттите години на Интернетот било поврзано околу 100.000 компјутери од околу дваесетина држави – САД, Канада, Јапонија, Мексико и поголемиот дел од западноевропските држави.

На почетокот на деведесеттите години ARPANet веќе не постои, а се воведуваат нови услуги и протоколи: WAIS (Wide Area Information Servers), Gopher и World Wide Web (WWW). WWW подоцна станува најпознатата и најкористената услуга на Интернет. Кон

крајот на деведесеттите доаѓа до комерцијализација на Интернет.Сè повеќе компании поставуваат свои веб-страници, а 1994 година се појавува првата online „продавница". Се развиваат нови технологии и услуги, како што се пребарувачи на Интернет, електронското работење, пренос на слика и на звук во реално време, Интернет телефонија итн. Во тоа време на Интернет се поврзани преку 56 милиони компјутери и има околу 200 милиони корисници.

Интернетот настанал спонтано и претставува феномен – тоа е мрежа која нема сопственик. Ниедна институција не е сопственик на мрежата како целина, поедини институции имаат сопственост само врз дел од комуникациските врски и опрема. Денес Интернетот го користат неколку милијарди луѓе, а со употреба на безжичните технологии бројот на корисниците секојдневно расте.

6.3.2 Функционирање на Интернет

Интернет технологијата се развивала постапно. Прво се започнало со пренесување на датотеки, со електронската пошта, потоа со протокол за контрола на преносот TCP/IP (Transmission Control Protocol/Internet Protocol), со мрежните вести, со WWW (World Wide Веб) и со различни сервиси (Archie, Gopher…). Понатаму се појавуваат системи за безбедна наплата преку мрежа, мултимедија и сервисите како што се chat, форуми и различни online заедници. Сервисите се појавуваат и исчезнуваат со развојот на технологијата. Порано се користеле сервиси како што се Archie и Gopher со кои можело да се види што се наоѓа на некој FTP сервер. Но, со појавата на WWW овие сервиси станале непотребни.

WWW денес е дел од секојдневниот живот. Луѓето користат Интернет за различни потреби – да читаат вести и книги, да комуницираат со пријателите и со соработниците, да се информираат, да купуваат, за игра и забава и за многу други работи. WWW е сервис на Интернет кој содржи веб-страници меѓусебно поврзани со линкови и организирани во веб-локации кои се наоѓаат на некој сервер. Програмите кои го пребаруваат WWW и пронаоѓаат страници според зададени критериуми се нарекуваат веб-пребарувачи (Search engine). Прегледување на овие страници се врши со посебни програми наречени веб-прелистувачи (Веб browser). Веб-прелистувачите овозможуваат лесно добивање на информации без потреба да се знаат сложени протоколи кои се наоѓаат во позадината.

Протоколи и адреси на Интернет

Со секоја активност на Интернет управуваат протоколи. Компјутерите комуницираат така што разменуваат одредени податоци, при тоа протоколите ги дефинираат форматот и редоследот на податоците, како и акциите кои се преземаат по праќањето или по примањето на податоци и на пораки.

Протокол за контрола на преносот/интернет протокол TCP/IP

Протокол за контрола на преносот/интернет протокол (TCP/IP – Transmission Control Protocol/Internet Protocol) е еден од основните Интернет протоколи кој го користат речиси сите компјутерски мрежи кои се поврзани на Интернет, а се користи и во локалните мрежи. Други протоколи, на пр. протокол за електронска пошта или за пренос на веб-страници работат врз основа на овој протокол. TCP/IP се користи за пренесување на податоци помеѓу компјутерите поврзани во мрежата. Тој, исто така, овозможува поврзување на различни видови компјутери, сервери и периферискиуреди, но и на разни оперативни системи.

TCP/IP протокол се состои од два дела. Првиот дел, TCP, податоците ги дели на помали пакети заради полесен и посигурен пренос и на одредишната адреса повторно ги спојува во оригинална датотека. Вториот дел, IP, е основниот протокол со кој се дефинира начинот на кој на секој компјутер му се доделува единствена IP адреса со помош на која тој лесно може да се идентификува. Основната функција на овој протокол е да ги проследи податоците од почетната до крајната IP адреса преку патека која ја одредува рутер.

Адреса на интернет-протокол (IP адреса)

Секој компјутер кога ќе се приклучи во Интернет мрежата добива единствена IP адреса која се состои од 4 бајти. IP адреса, во бинарен запис може да биде, на пр. 10100000.01100011.00110011.00010100. Оваа адреса преведена во декаден запис е 160.99.54.20. (Броевите помеѓу точките се од 0 до 255. Зошто?) Броевите во IP адресата ја одредуваат локацијата на компјутерот во мрежата. Првиот дел во адресата определува дел на мрежата (држава или регион во држава), а вториот дел го дефинира самиот компјутер.

За љубопитните:

IP адресите ги доделува организацијата ICANN (Internet Corporation for Assigned Names and Numbers – www.icann.org) преку организацијата IANA (Internet Assigned Number Authority – www.iana.org). Постои одреден опсег на адреси кои се означени како приватни и не се поврзани со Интернет.

Адресата изразена со броеви не е погодна за памтење, па се воведуваат симболички адреси. Преведување на IP адреса во симболичка адреса се врши системот за именување на домени DNS (Domain Name System). DNS е база на податоци во која се зачувани сите IP адреси и соодветни симболички адреси и по барање на корисникот тие се преведуваат една во друга. DNS постојано се ажурира затоа што IP адресите на некои корисници се менуваат (динамички IP), а и бројот на адресите непрекинато се зголемува.

Симболичка адреса

Симболичка адреса е единствено определена и најчесто се состои од четири или пет кратенки раздвоени со точки, на пр. www.pmf.ukim.edu.mk. Таа секогаш има форма: *ime_na_servis.ime_na_domen*, каде *ime_na_servis* може да биде:

- *www* – кратенка за world wide web сервис, систем на меѓусебно поврзани хипертекст документи
- *smtp* – кратенка за сервисот за е-пошта
- *ftp* – кратенка за сервисот за пренос на датотеките и други.

ime_na_domen се состои обично од две или три (најмногу 127) кратенки раздвоени со точки. Овие кратенки се нарекуваат нивоа на името на доменот (domain levels).

Првото ниво може да биде кратенка за држава кај национални домени (на пр. mk, rs, uk, bg итн.) или кратенка за област на делување кај меѓународни домени. Најпознатите кратенки за област на делување се:

- .edu – едукативни веб-локации
- .com – комерцијални веб-локации
- .gov – веб-локации на Владините институции
- .net – веб-локации на администратори на мрежи
- .org – веб-локации на непрофитни организации

Секое следно ниво на доменот е подниво на претходниот домен (хиерархиска организација) Последното ниво е име на серверот и тоа е единствено име. На пр. домените *pmf.ukim.edu.mk* и *feit.ukim.edu.mk* имаат исти имиња на првите две нивоа, додека имињата на серверите се различни.

За љубопитните:

Првата симболичка адреса е symbolic.com.

Протокол за пренос на датотеките FTP

Протокол за пренос на датотеките – FTP (File Transfer Protocol) е еден од најстарите протоколи. Се користи за праќање и за примање на датотеки преку мрежите кои ги поддржуваат TCP/IP протоколите, независно од оперативниот систем кој се наоѓа на компјутерите. Бидејќи FTP е и апликација, се смета и за еден од сервисите на Интернет.

Протокол за пренос на хипертекст HTTP

Протокол за пренос на хипертекст – HTTP (HyperText Transfer Protocol) е протокол кој го користат сите сервери и прелистувачи за да комуницираат меѓусебно. Со овој протокол веб-страниците се пренесуваат од серверот до веб-прелистувачот на клиентот. Прелистувачот чита податоци и ги презентира на корисникот. Веб-страници се хипертекстови напишани во хипертекст маркирачкиот јазик – HTML (HyperText Markup Language). Доколку веб-страница која е вчитана во веб-прелистувач содржи линкови, корисникот може со кликнување на некој од линковите да отвори некој друга страница.

Униформен локатор на ресурси URL

Поврзаните веб-страници и други ресурси можат да бидат на ист сервер или на различни сервери оддалечени со илјадници километри. Процесот на насочување од еден до друг сервер е едноставен и јасен за корисникот. Линковите препознаваат единствени ресурси на мрежата преку униформен локатор на ресурси URL (Uniform Resourse Locator). Секој ресурс на Интернет, исто како и секој компјутер, има своја единствена URL адреса. Структура на URL адресата е:

ime_na_protokol://ime_na_domen/pateka

Ime_na_protokol е кратенка која укажува на протокол кој се користи при пристапување до документот:

- http укажува на протокол за пренос на хипертекст HTTP
- ftp укажува на протокол за пренос на датотеките FTP
- mailto укажува на протокол за пренос на е-пошта MTP (Mail Transfer Protocol)

Името на протоколот е придружено со ознаката :// која е составен дел на адреса на секоја веб-страница. Во повеќето веб-прелистувачи името на протоколот не мора да се напише.

Со *ime_na_domen* е означен серверот на кој ресурсот се наоѓа.

Pateka е патека до ресурсот на серверот, на пр. *http://www.mon.gov.mk/mk/dokumentimon/documents/soopstenie.doc* е URL адреса на документот *soopstenie.doc* на серверот чиј домен е *www.mon.gov.mk/mk*.

Протокол на безбеден пренос SSL

SSL (Secure Socket Layer) е протокол за безбедна комуникација помеѓу клиент и сервер. Безбедна верзија на HTTP е HTTPS и има поддршка за енкрипција и утврдување на идентитет. Енкрипција се користи за спречување на натрапници да прислушкуваат

комуникација која содржи чувствителни информации (број на кредитна картичка и слично). Утврдување на идентитет се користи за точно да се знае дали на другиот крај навистина е оној кој треба да ги прими податоците.

Резиме

Првата компјтерска мрежа, наречена ARPANet настанала во 1969 год. Во осумдесеттите години почнува да се користи терминот Интернет кој се однесува на мрежа која користи TCP/IP протокол.

Со секоја активност на Интернет управуваат *протоколи*.

TCP/IP е еден од основните Интернет протоколи кој го користат речиси сите компјутерски мрежи кои се поврзани на Интернет. Секој компјутер кога ќе се приклучи во Интернет мрежата добива единствена *IP адреса* која се состои од 4 бајти. Адреса изразена со броеви не е погодна за памтење, па се воведуваат *симболички адреси*. Преведување на IP адреса во симболичка адреса го врши *DNS сервис*.

FTP е протокол кој се користи за праќање и примање на датотеки преку мрежите.

HTTP е протокол за пренесување на веб-страници.

Секој документ на Интернет, исто како и секој компјутер, има своја единствена *URL адреса*.

Прашања:

1. Која е првата компјутерска мрежа и кога е создадена?
2. Колку компјутери на почетокот биле поврзани во првата компјутерска мрежа?
3. Со кој протокол е заменет протоколот NCP?
4. Кога за првпат е употребен терминот Интернет?
5. Колку компјутери денес се поврзани на Интернет?
6. За што најчесто луѓето го користат Интернетот?
7. Што е WWW?
8. Како се нарекуваат програми за пронаоѓање на веб-страници на Интернет?
9. Со кои програми се разгледуваат веб-страници на Интернет?
10. Што е протокол? Кои протоколи на Интернет најчесто се користат?
11. Што е IP адреса и каква форма таа има?
12. Зошто е воведена симболичка адреса?
13. Каква форма има симболичка адреса?
14. Што се: DNS, FTP, HTTP, HTML, URL и SSL?

6.4 Напредни веб технологии

6.4.1 Веб 1.0?

Од самиот почеток веб е замислен како место каде поединци ќе можат да објавуваат различни содржини кои заинтересираните корисници ќе можат да ги најдат и прегледаат. Во првобитниот веб (кој сега се нарекува Веб 1.0) не постоела комуникација, креаторите на содржините поставувале содржини, а други корисници можеле

само да ги прегледуваат. Услугите на Веб 1.0, на пр. Hotmail или Yahoo се однесувале само на читање на електронската пошта и читање садржини на статичките веб-локации. Откако ќе пронајдат соодветен портал, корисниците можеле да пронајдат информации кои ги поставил сопственик на порталот и да ги прочитаат или преземат, но немало можност корисник да постави свои содржини, како што се слики, видео записи или документи.

6.4.2 Веб 2.0?

Под терминот Веб 2.0 се мисли на развој на веб по 2000 година, кога дошло до промени во начинот на користење на веб. Основните карактеристики на Веб 2.0 се отвореност, слобода и колективна интелигенција. Веб 2.0 од една страна е социјален елемент кој на корисниците им овозможува социјализација преку мрежата, од друга страна е финансиски елемент кој со понудените сервиси дава простор за пропаганда и проучување на навики на потрошувачите, т.е за заработка.

Интернетот се развил во платформа која преку низа алатки на корисниците им овозможува да учествуваат во креирање на веб-содржини од различен вид (на пр. online дневник во форма на блог, лична презентација на www.myspace.com) и директна размена на информации со други корисници (на пр. да разменуваат слики преку веб-локации како што е Picasa или видео записи преку веб-локации како што е Youtube).

Она што го карактеризира Веб 2.0 е следново:

- корисниците можат да користат апликации преку својот веб-прелистувач,
- корисниците се тие на кои податоците им припаѓаат и кои имаат контрола врз нив,
- структурата на веб ги поттикнува корисниците да учествуваат во креирање на веб-содржини или апликации,
- унапредени графички опкружувања во однос на т. н. Веб 1.0.

Примери за Веб 2.0 се социјални мрежи, блогови со коментари на посетителите, „вики" веб-локации, веб-локации за размена на видео записи (video sharing sites), веб-локации за размена на фотографии и документи, веб апликации, фолксономија (folksonomy) и други интерактивни веб-локации.

Online заедници

Виртуелна или online заедница е група на луѓе чија комуникација се одвива преку Интернет. Ваквите заедници се резултат на социјалното умрежување (social networking) на Интернет. Социјално умрежување означува активно учество во виртуелни заедници, односно збир на корисници со заеднички интереси собрани околу некој интернетски сервис (блогови, форуми итн.). Најпопуларните социјални мрежи се Facebook и MySpace.

Блог е термин кој се однесува на личен дневник пишуван на веб со обратно хронолошки подредени содржини. Со терминот блог се врзуваат и термините блогосфера – заедница на корисници на Интернет кои учествуваат во креирање на блогови, блогер – автор на блог и блогирање – зачестено пишување на блогови и коментирање на туѓи блогови. Постојат и мобилни блогови прилагодени на пишување и читање преку мобилните телефони и Podcast или аудиоблогови зачувани како звучна датотека.

Значајно место во социјалната интеракција заземаат и форумите – јавно дискутирање за некои теми преку Интернет, како и instant messaging или chat – размена на пораки во реално време.

Фолксономија и тагирање

Ознака или *таг* (tag) се однесува на текст кој опишува значење или структура на некоја содржина на Интернет (слика, блог, профил, веб-страница и друго). Оваа едноставна техника претставува основа на моќ и популарност на новите веб алатки и услуги кои секојдневно се развиваат и на кои корисниците пишуваат, ги менуваат и ги опишуваат содржините.

Облак или *збир од тагови* (tag cloud) често се користи како визуелен приказ на тагови кои некоја веб-страница или услуга ги користат. При тоа таговите кои почесто се користат се прикажани со поголем фонт или се на некој друг начин нагласени, а се подредени според абецедата или според азбуката. Избирање на една ознака од збирот на ознаките корисникот ќе го одведе до содржините кои се означени со таа ознака.

На овој начин се добива нова категоризација на веб-содржините наречена *фолксономија* (folksonomy) што значи категоризирање (taxonomy) од страна на „народ“ (folks). Таксономија е систем на категоризирање кој го дефинираат стручњаци, додека кај фолксономија се работи за категоризација која настанала спонтано од милиони обични корисници преку тагови. Кога корисниците поставуваат линк кон некоја содржина на веб или поставуваат некоја датотека, тие ја опишуваат како други корисници би можеле поедноставно да ја пронајдат. Најпознатата веб-локација категоризирана на овој начин е Wikipedia.

Сервиси за објавување и размена на содржини

На Интернет постојат многу сервиси на кои корисниците можат да остават свои датотеки (фотографии, документи, аудио и видео записи итн.) Flickr.com е комбинација на сервис на Интернет за објавување на дигитални фотографии и социјален веб. YouTube е сличен сервис за објавување, прегледување, размена и коментирање на видеозаписи.

6.4.3 Семантички веб или Веб 3.0

Најголемата вредност на концептот Веб 2.0 е тоа што тој претставува прв чекор кон интелигентен или т.н. семантички веб кој веќе се развива. Веќе постојат сервиси кои овозможуваат различни области да се следат од едно место, на пр. RSS читач (Rich Site Summary). Со добро избрани локации и тагови за следење може да се постигне ништо што е значајно од избраните области да не пропуштиме.

Семантички веб означува веб кој ќе претставува надградба на постоечкиот Веб 2.0, а во центар ќе го постави корисникот. Тој ќе се базира врз податоци и информации кои ќе доаѓаат од страна на корисниците и ќе се приспособува кон нивните потреби. На пр. корисниците кои за пребарување често ги користат клучните зборови „веб дизајн“ ќе добиваат повеќе информации и реклами врзани со веб дизајнот. Пребарување на содржините ќе генерира локални резултати според географската положба на корисникот и ќе понуди веб-локации за кои смета дека корисникот сака да ги види.

Придобивките од семантичкиот веб се големи. До податоците ќе може да се пристапи од кое било место. Ова е овозможено со користење на „паметните“ телефони (smartphone) итн. облак (cloud) апликации.

Не постои точна дефиниција ниту за Веб 2.0 ниту за Веб 3.0. Веб 3.0 е само идеја и никој не знае што тој точно ќе донесе. Многумина под употреба на Веб 3.0 подразбираат креација и користење на интернет профили креирани врз податоците зачувани во историја на пребарувањата. Со користење на вакви профили би било можно содржините да се прилагодат кон корисникот и кон неговите интересирања. На пример, пребарување на информации ќе биде различно од корисник до корисник а ќе се темели на нивните поранешни пребарувања. Тоа значи дека овој сервис ќе ги знае сите барања и навики на корисниците и содржините ќе ги организира во согласност со нив.

Препорака:

Погледни ги следните видео записи: како работи Интернет на http://www.warriorsofthe.net/movie.html и Веб 3.0 видео на http://vimeo.com/11529540

Резиме

Во Веб 1.0 не постоела комуникација, креаторите на содржините поставувуваат содржини, а други корисници можат само да ги прегледуваат.

Веб 2.0 на корисниците им овозможува да учествуваат во креирањето на веб-содржини од различен вид и директна размена на информации со други корисници. Примери за Веб 2.0 се социјални мрежи, блогови со коментари на посетители, „вики" веб-локации, веб-локации за размена на видео записи, веб-локации за размена на фотографии и документи, веб апликации, фолксономија и други интерактивни веб-локации.

Семантички веб или Веб 3.0 означува веб кој ќе се базира врз податоци и информации кои ќе доаѓаат од страна на корисниците и ќе се приспособува кон нивните потреби.

Прашања:

1. Како корисниците доаѓале до информации на Веб 1.0?
2. Дали во Веб 1.0 постоела можност корисник да постави свои содржини, како што се слики, видео записи или документи?
3. Како корисниците учествуваат во креирање на содржини на Веб 2.0?
4. Кои се карактеристики на Веб 2.0?
5. Преку кои сервиси луѓето се поврзуваат во онлајн заедници?
6. Дали имаш креирано свој блог? Ако да, преку кој портал?
7. Посети ги сервисите sites.google.com, wikispaces.com, aboutme.com, wordpress.com и слично. Разгледај какви можности тие даваат за креирање на сопствена веб-локација.
8. Што е таг?
9. Што е фолксономија?
10. Наброј неколку сервиси на кои можеш да оставиш фотографии, аудио или видео фајлови и слични содржини!
11. Што е семантички веб?
12. Дали можеш да направиш разлика помеѓу Веб 2.0 и Веб 3.0?
13. Дали знаеш што се App апликации? Посети docs.google.com и креирај документ.
14. Регистрирај се на https://live.com и разгледај кои сервиси се во понуда.

6.5 Социјални мрежи

Социјалните мрежи (social networks) овозможуваат комуникација помеѓу корисници и претставуваат една форма на дружење. Една од најпопуларните социјални мрежи е Facebook, каде корисник може да постави своја слика, да објави приватни податоци, да опише свои интереси, да развива тематски дискусии и сл. Корисникот може во своја група да прими или да исклучи поедини членови. Можно е да се иницираат одредени политички, хуманитарни или други акции. Слична намена има и Twitter.

Интернет форум е услуга на Интернет која овозможува размена на мислења помеѓу членови со користење на веб-прелистувач. Сите пораки кои корисникот ќе ги напише и ќе ги прати на форум ги гледаат сите други корисници на форумот. Тоа е како огласна табла (message board) на која членови оставаат пораки. Се работи за расправи по некоја тема која ја иницира некој од членовите на форумот. Форумот обично е поделен во неколку теми заради полесно снаоѓање. Членовите можат да останат анонимни. За да се стане член на форум потребно е зачленување, главно бесплатно, со пријава од тип корисничко име/лозинка. Зачленувањето е поврзано со адреса на електронска пошта на корисникот заради идентификација. За услугата на форумот да не се злоупотреби и да се почитуваат правилата на прифатливото однесување, се грижат администраторите и модераторите. Нив ги именува сопственикот на форумот, а тоа обично се членови кои доброволно се пријавуваат. Тие можат да исклучат поедини членови и да избришат недолични содржини.

6.5.1 Правила за добро однесување во комуникација преку Интернет

Во електронската комуникација лицата кои комуницираат обично не се гледаат, туку разменуваат пишувани пораки. Некои луѓе мислат дека тогаш можат да се однесуваат грубо и невоспитано, но треба да се знае дека секоја комуникација има свои правила кои треба да се знаат и да се почитуваат. Од друга страна е исто така личност која има чувства и не треба да се каже нешто што може да ја повреди или налути.

Почитувај ги следните правила:

- Секогаш претстави се.
- Биди љубезен.
- Почитувај туѓи мислења.
- Почитувај туѓа приватност.
- Никогаш не пишувај само со големи букви, тоа значи дека викаш.
- Кога одговараш на прашање, реченицата започни ја со име на лицето на кое му одговараш, особено ако во разговор учествуваат повеќе лица.
- Не користи двосмислени зборови и изјави.
- Не користи сарказам, други не можат да го видат изразот на твоето лице и пораката можат погрешно да ја разберат.
- Во пораките додај соодветен „смајли“ за појасно да изразиш што чувствуваш .
- Треба да знаеш дека она што ќе го напишеш останува трајно забележено и да тоа можат да го прочитаат и други лица освен оние на кои пораката им е наменета.
- Кога користиш кратенки провери дали тие точно се напишани. Кратенките треба

да бидат општо познати како друго лице би можело правилно да ги разбере.

- На крајот од разговор уредно одјави се.

Општи правила за добро однесување во комуникација на Интернет се нарекуваат Нет етика или Нетика (Netiquette – скратено од Internet etiquette).

6.5.2 Приватност и безбедност на Интернет

Во време на современите технологии секој може да се изрази преку Интернет и да постави свои содржини кои можат да бидат добронамерни и точни или злонамерни и неточни. Она што ги карактеризира овие мрежи е ширење на информации, но и дезинформации, со огромна брзина.

Корисниците мора да бидат свесни дека покрај корисни информации постојат измами, лаги и различни форми на манипулации. Доколку информациите не се проверуваат лесно може да се случи некои корисници да бидат изманипулирани и измамени. Измамените корисници можат да загубат материјални и морални вредности.

Исто така, преку Интернет се пренесуваат лични податоци, се водат доверливи деловни и приватни разговори, се пренесуваат доверливи воени и државни информации, се вршат финансиски трансакции и уште многу работи. Секогаш постои опасност овие информации да ги види некој кој може да ги злоупотреби. На корисниците може да им биде загрозена приватноста и репутацијата, да им бидат украдени пари од банкарски сметки, онеспособен компјутерот, дури и да им биде загрозена безбедноста.

Препорака:

Погледни како низ годините се менува приватноста на Facebook:
http://mattmckeon.com/facebook-privacy/

Компјутерскиот криминал сè повеќе е во подем. Злонамерните поединци, организираните хакерски групи, па и компјутерските криминалци можат да:

- заразат компјутер со злонамерен софтвер за да украдат идентитет или да ги следат навиките на корисниците,
- да украдат лозинки за пристап до банковни сметки и да извршат кражба,
- да направат проблем во работата на компјутерот со помош на вируси и друг злонамерен софтвер,
- да преземаат контрола врз компјутер и да го користат за напад на други корисници,
- да ги наведат корисниците да посетат некоја лажна веб-локација и таму да остават лични податоци,
- да упаднат во нечија мрежа и бесплатно да користат туѓа интернет конекција,
- да украдат и да користат туѓи налози за електронска пошта и друго.

Мерки на заштита

За да се спречи злоупотреба на податоци и да се обезбеди нивна заштита на Интернет, неопходно е да се применат соодветни мерки на заштита. Постојат и правни мерки на заштита, но компјутерскиот криминал многу тешко се открива.

Мерките кои сами можеме да преземаме за да се заштитиме од земање на податоци на измама се следните:

- заштитен софтвер: антивируси и огнениот ѕид,
- ажурирање на оперативен система, блокирање на несакана е-пошта (spam) и користење на нови верзии на интернет прелистувачи,
- редовно правење на резервни копии (back-up),
- доколку не мора, компјутерот немој да го користиш како администратор.

Заштита на идентитет и приватност

Првиот чекор во заштитата на личните информации е отворање на безбеден профил:

- Внимателно избери информации кои во него ќе ги наведеш.
- Лозинките секогаш држи ги во тајност.
- Слики праќај само на лица на кои им веруваш.
- Научи како да блокираш пораки од непознати луѓе.
- Игнорирај непознати луѓе кои сакаат да остварат комуникација со тебе.
- Не прифаќај состаноци со непознати, а доколку прифатиш состанок извести ги родителите.
- Ако ништо не работиш на Интернет, исклучи се.
- Преку веб камера комуницирај само со познати лица, камерата исклучи ја веднаш по престанок на комуникација.

Запомни!

На Интернет нема опцијата undo – тоа што си го направил останува забележано. Доколку, на пример, оставиш своја слика на некоја од социјалните мрежи, понатаму немаш контрола кој и како сликата ќе ја користи.

Онлајн пазарот и банкарството се сигурни и безбедни доколку се придржуваме на некои едноставни препораки. Доколку купуваш преку Интернет биди сигурен/на дека се работи за доверлива компанија:

- Провери дали компанијата навистина постои во реалниот свет.
- Провери дали веб-локацијата е безбедна, т. е. дали во URL адресата протоколот е означен со https:// и дали постои ознака на сигурносен катанец (padlock image).
- Провери дали постојат јасни правила во врска со приватноста и враќањето на производите.

Научи да препознаваш лажни веб-локации и реклами кои можат да доведат до тоа да бидеш измамен:

- обично се ветуваат големи награди, на пр. добивка на лотарија,
- се остава лажен впечаток за итноста,
- има необични и непотребни детали за привлекување на внимание,
- се бара плаќање однапред или давање на лични податоци.

Внимавај!

Содржините како што се keygene, crack и многу други кои бесплатно можат да се преземаат од Интернет најчесто се заразени со некој вид на злонамерен софтвер.

Некои содржини на Интернет направени се така што можат да одбегнат и најсовремена антивирусна заштита. Никогаш не отворај прилози во електронска пошта која доаѓа од непознати лица, но внимавај и со прилози во пораки од познати лица бидејќи некогаш пораките се пратени без нивното знаење или ни сами не знаат што се наоѓа во прилог на нивната порака.

Постојат многу веб-локации кои бараат регистрација. На таквите веб-локации се собираат податоци за сите корисници на нивните услуги кои подоцна можат да се користат за праќање на рекламни пораки. Но тие сервери можат да бидат нападнати од страна на хакери кои тие податоци можат да ги злоупотребат.

Многу веб-локации земаат податоци од корисниците по пат на т. н. колачиња (cookies) без нивното знаење. Најдобар начин за заштита од ваков вид на собирање информации е колачињата да се исклучат во веб-прелистувачот. Преку менито за помош (Help) информирај се како да ги исклучиш колачињата на твојот веб-прелистувач.

Резиме

Социјалните мрежи овозможуваат комуникација помеѓу корисниците и претставуваат една форма на дружење. Интернет форум е услуга на Интернет која овозможува размена на мислења помеѓу членови со користење на веб-прелистувач.

Комуникацијата преку Интернет има свои правила кои треба да се знаат и да се почитуваат. Општите правила за добро однесување во комуникацијата на Интернет се нарекуваат *Нетика*

Покрај корисни информации на Интернет постојат измами, лаги и различни форми на манипулации. Исто така, преку Интернет се пренесуваат лични податоци, се водат доверливи деловни и приватни разговори, секогаш постои опасност овие информации да ги види некој кој може да ги злоупотреби.

За да се спречи злоупотреба на податоци и да се обезбеди нивна заштита на интернет неопходно е да се применат соодветните мерки на заштита.

Прашања:

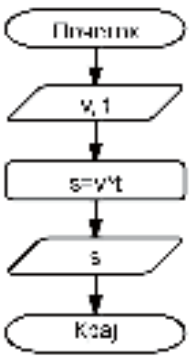
1. Што се социјални мрежи?
2. Кои социјални мрежи ги познаваш? Дали користиш некоја социјална мрежа?
3. Што е форум? Како се станува член на форум?
4. Кој се грижи за почитување на правилата за однесување на форум?
5. Како треба да се однесуваш во комуникација преку Интернет?
6. Наведи неколку правила на добро однесување при Интернет комуникација?
7. Што е Нетика? Како настанал зборот Нетика?
8. Дали корисниците на Интернет секогаш се добронамерни?
9. Дали сите информации на Интернет се точни?
10. Дали приватност на Интернет е загрозена?
11. Како можат да се злоупотребат личните податоци и доверливите информации?
12. Што е компјутерски криминал?
13. Што можат да направат хакери и компјутерски криминалци?
14. Како може да се заштити приватност на Интернет?
15. Дали купувањето преку Интернет е безбедно?
16. Како ќе препознеш веб-локации кај кои податоците се заштитени и сигурни?

Решенија

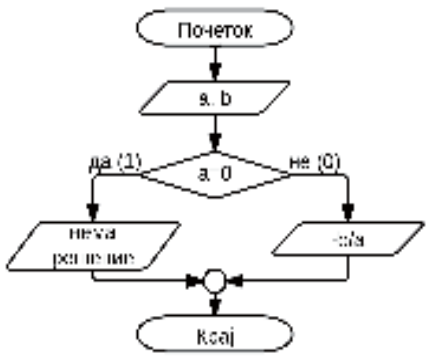
4.1 Поим за алгоритми и програми

4.1.1 Алгоритми

2. почеток
   читај t, v;
   s=v*t;
   печати s;
   крај



3.



4.2 Програмирање и програмски јазици

4.2.5 Извршување и изглед на готови пример програмски кодови

1.
```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      cout<<"Zdravo!"<<endl;
      cout<<"Kako si?"<<endl;
      system("PAUSE");
      return 0;
}
```

2.
```
#include<cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int god;
      cout<<"Hello world!"<<endl;
      cout<<"I'm C++"<<endl;
      system("PAUSE");
      return 0;
}
```

3. Направена е логичка грешка: треба 100*61.5.
4. Грешките се означени со црвената боја:

```
#include<cstdlib>
#include <iostream>
using namespace std;
int main()
{
      Cout<<"Kako si?"<<endl;
      cout<<"Mislis deka programiranjeto e tesko?">>endl;
      cout<<"Ne grizi se!";
      cout<<"Nabrgu ke razberes mnogu poveke";
      system("PAUSE");
      return(0)
}
```

4.3 Програма со редоследна структура

4.3.2 Искази. Исказ за приказ на екран

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int god;
    cout<<"Zdravo!"<<endl;
    cout<<"Kakov prekrasen den!"<<endl;
    cout<<"Odam na prosetka."<<endl;
    system("PAUSE");
    return 0;
}
```

4.4.2 Типови на променливи

Прашања:

7. Исправна е декларација под а).
9. Ќе се прикаже: bcb.
10. Ќе се прикаже: 1.

Задачи:

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int kolicina=24;
    float cena=12.5;
    cout<<"kolicina, cena"<<endl;    //a)
    cout<<kolicina<<"    "<<cena<<endl; //b)
    cout<<"kolicina="<<kolicina<<" cena="<<cena<<endl;//v)
    system("PAUSE");
    return(0);
}
```

2.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a, b, P;
    a=0.5;
    b=3.2;
    P=a*b;
    cout<<"Ploshtinata na pravoagolnikot so stranite a= ";
    cout<<a<<" i b= "<<b<<" iznesuva P= "<<P<<endl;
    system("PAUSE");
    return 0;
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float r=2.5;
    cout<<"Plostinata na krugot so radiusot";
    cout<<" r= "<<r<<" iznesuva "<<3.14*r*r<<endl;
    system("PAUSE");
    return(0);
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a,P, L;
    a=8.5;
    P=a*a;
    cout<<"Plostinata na kvadratot so stranata a = "<<a;
    cout<<" iznesuva "<<P<<endl;
    cout<<" a negoviot perimetar e "<<L<<endl;
    system("PAUSE");
    return 0;
}
```

5.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int alfa=30;
    cout<<"Agolot na vrvot iznesuva ";
    cout<<180-2*alfa<<" stepeni"<<endl;
    system("PAUSE");
    return(0);
}
```

6.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a,b;
    a = 1;
    b=-18;
    cout<<"Koeficinta na ravenkata e "<< b/a;
    system("PAUSE");
    return 0;
}
```

4.5 Дополнителни специфики на јазикот

4.5.2 Техника за објаснувања за податоците кои се очекуваат од корисникот

Прашања:

7. 2, 3, 1

Задачи:

1.
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a,b,P,L;
    cout<<"Vnesi gi stranite na pravoagolnikot:";
    cin>>a>>b;
    P=a*b;
    L=2*a+2*b;
    cout<<"Plostinata na pravoagolnikot e "<<P;
    cout<<" a negoviot perimetar e "<<L<<endl;
    system("PAUSE");
    return 0;
}
```

2.
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a,P,L;
    cout<<"Vnesi gi stranite na kvadratot:";
    cin>>a;
    P=a*a;
    L=4*a;
    cout<<"Plostinata na kvadratot e "<<P;
    cout<<" a negoviot perimetar e "<<L<<endl;
    system("PAUSE");
    return 0;
}
```

3.
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    const float PI = 3.14;
    float r;
    cout<<"Vnesi go radiusot:";
    cin>>r;
    cout<<"Plostinata na krugot e "<<PI*r*r;
    cout<<" a negoviot perimetar e "<<2*PI*r<<endl;
    system("PAUSE");
    return 0;
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float cena, kolicina, iznos, kusur;
    cout<<"Cenar ";
    cin>>cena;
    cout<<"Kolicina: ";
    cin>>kolicina;
    cout<<"Iznos: ";
    cin>>iznos;
    kusur=iznos-cena*kolicina;
    cout<<"Kusur: "<<kusur<<endl;
    system("PAUSE");
    return(0);
}
```

4.5.3 Дополнителни спецификации на јазикот C++ (прв дел)

Прашања:

1. а) 385/100=3 б) 385%10=5
   в) 385/10%10=8 г) 385%100/10= 8
2. а) 4 б) 4.5
3. 2*(a+b) 1./3*x (a+b)/2.
4. а) 6 б) 12
5. а) 64 б) 4 в) 4
6. а) a-=9 б) x/=(y+1) в) a*=((b-c)+5))
7. а) 7 б) 6

Задачи:

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a,b;
    float sr_vred;
    cout<<"Vnesi dva celi broja: ";
    cin>>a>>b;
    sr_vred = (a+b)/2.;
    cout<<"Sredna vrednost na "<<a;
    cout<<" i "<<b<<" e "<<sr_vred;
    system("PAUSE");
    return 0;
}
```

2.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n,a,b,c;
    cout<<"Vnesi trocifren broj: ";
    cin>>n;
    a=n/100;
    b=n/10%10;
    c=n%10;
    cout<<"Na brojot "<<n<<"prvata cifra e "<<a;
    cout<<", vtorata cifra e "<<b<<", tretata cifra e "<<c;
    system("PAUSE");
    return(0);
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int sec,cas,min;
    cout<<"Vnesi vreme vo sekundi: ";
    cin>>sec;
    cout<<sec<<" sekundi se ";
    cas=sec/3600;
    sec=sec%3600;
    min=sec/60;
    sec=sec%60;
    cout<<cas<<" casovi, "<<min<<" minuti i ";
    cout<<sec<<" sekundi."<<endl;
    system("PAUSE");
    return 0;
}
```

4.5.4 Дополнителни спецификации на јазикот C++ (втор дел)

Прашања:

2. 68
4. Нема разлика

Задачи:

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    string ime, prezime;
    cout<<"Vnesi ime: ";
    cin>>ime;
    cout<<"Vnesi prezime: ";
    cin>>prezime;
```

```
11      cout<<prezime<<" "<<ime;
12      system("PAUSE");
13      return 0;
14  }
```

4.6 СТРУКТУРА ЗА ИЗБОР ОД ДВЕ МОЖНОСТИ

4.6.2 Структура (исказ) избор од две можности

Прашања:

5. а) 1 б) 0 в) 0 г) 1 д) 0
6. а) 1 б) 1 в) 0 г) 1
11. а) Brojot 5 e pomal od 10 б) Brojot 15 e pogolem od 10
12. а) 10 б) 5
13. а) 8 б) 3 в) 0

Задачи:

1.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       int a;
7       cout<<"Vnesi cel broj: ";
8       cin>>a;
9       if (a%2==0)
10           cout<<"Brojot "<<a<<" e paren";
11      else
12           cout<<"Brojot "<<a<<" e neparen";
13      system("PAUSE");
14      return 0;
15 }
```

2.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       int a,b;
7       cout<<"vnesi dva broja: ";
8       cin>>a>>b;
9       if (a<b)
10           cout<<a<<"<"<<b;
11      else
12           cout<<b<<"<"<<a;
13      system("PAUSE");
14      return 0;
15 }
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a;
    cout<<"Strana na kvadratot: a=";
    cin>>a;
    if (a>0)
        cout<<"Perimetar na kvadratot e "<<4*a;
    else
        cout<<"Vrednosta na a mora da e pozitivna!";
    system("PAUSE");
    return 0;
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a,b;
    cout<<"Vnesi go namalenikot: ";
    cin>>a;
    cout<<"Vnesi go namalitelot: ";
    cin>>b;
    if (a>=b)
        cout<<a<<" - "<<b<<" = "<<a-b;
    else
        cout<<"Razlikata "<<a<<"-"<<b<<" ne postoi vo N";
    system("PAUSE");
    return 0;
}
```

5.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a;
    cout<<"Vnesi eden broj: ";
    cin>>a;
    cout<<"Apsolutnata vrednost";
    cout<<" na brojot "<<a<<" e ";
    if (a>=0)
        cout<<a;
    else
        cout<<-a;
    system("PAUSE");
    return 0;
}
```

6.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a, b, c, s;
    cout<<"Vnesi tri broevi ";
    cin>>a>>b>>c;
    s=0;
    if (a>0) s=s+a;
    if (b>0) s=s+b;
    if (c>0) s=s+c;
    cout<<"s="<<s;
    system("PAUSE");
    return 0;
}
```

7.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a, b, c, p, n;
    cout<<"Vnesi tri broevi: ";
    cin>>a>>b>>c;
    if (a>b)
        p=a;
    else
        p=b;
    if (p>c)
        n=p;
    else
        n=c;
    cout<<"Najgolemiot broj e"<<n;
    system("PAUSE");
    return 0;
}
```

8.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a, b, c, t;
    cout<<"Vnesi tri broja: ";
    cin>>a>>b>>c;
    if (a>b) {
        t=a; a=b; b=t;
    }
    if (a>c) {
        t=a; a=c; c=t;
    }
    if (b>c) {
        t=b; b=c; c=t;
    }
```

```
18      cout<<a<<", "<<b<<", "<<c;
19      system("PAUSE");
20      return 0;
21  }
```

9.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      float a, b, c;
7      cout<<"Vnesi tri broja: ";
8      cin>>a>>b>>c;
9      if (a+b>c&&a+c>b&&b+c>a)
10         cout<<"Moze da se formira triagolnik";
11     else
12         cout<<"Ne moze da se formira triagolnik";
13     system("PAUSE");
14     return 0;
15 }
```

10.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      int n,c1,c3;
7      cout<<"Vnesi tricifren broj: ";
8      cin>>n;
9      c3=n%10;
10     c1=n/100;
11     if (c1==c3)
12         cout<<n<<" e palindrom";
13     else
14         cout<<n<<" ne e palindrom";
15     system("PAUSE");
16     return 0;
17 }
```

4.6.3 Техника на вгнездување на искази

Прашања:

5. a = 1.44; 7. 1

Задачи:

1.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      int a;
7      cout<<"Vnesi broj: ";
8      cin>>a;
```

```
    if (a>0)
    {
        if (a%2==0)
            cout<<a<<" e pozitiven i e paren";
        else
            cout<<a<<" e pozitiven i e neparen";
    }
    else
        cout<<a<<" e negativen";
    system("PAUSE");
    return 0;
}
```

2.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b;
    float x;
    cout<<"Vnesi gi parametrite: "<<endl;
    cout<<"a= ";
    cin>>a;
    cout<<"b= ";
    cin>>b;
    if (a==0)
    {
        if (b==0)
            cout<<"Ravenkata ima beskonecno resenija";
        else
            cout<<"Ravenkata nema resenija";
    }
    else
    {
        x= -(float)b/a;
        cout<<"Resenieto e x= "<<x;}
    system("PAUSE");
    return 0;
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a, b, c;
    cout<<"Vnesi tri broja: ";
    cin>>a>>b>>c;
```

```
    if (a+b>c&&a+c>b&&b+c>a)
    {
        if(a==b && a==c)
            cout<<"Triagolnikot e ramnostran";
        else
        {
            if (a==b || a==c || b==c)
                cout<<"Triagolnikot e ramnokrak";
            else
                cout<<"Triagolnikot e raznostran";
        }
    }
    else
        cout<<"Ne moze da se formira triagolnik";
    system("PAUSE");
    return 0;
}
```

4.7 Структура за избор од повеќе можности

Прашања:

2. Bravo!
3. Не постои разлика.

Задачи:

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int den;
    cout<<"Vnesi reden broj na denot: ";
    cin>>den;
    if (den==1)
        cout<<"Denes e ponedelnik!";
    else if (den==2)
        cout<<"Denes e vtornik!";
    else if (den==3)
        cout<<"Denes e sreda!";
    else if (den==4)
        cout<<"Denes e cetvrtok!";
    else if (den==5)
        cout<<"Denes e petok!";
    else if (den==6)
        cout<<"Denes e sabota!";
    else if (den==7)
        cout<<"Denes e nedela!";
    else
    cout<<"Pogresen podatok!";
    system ("PAUSE");
    return 0;
}
```

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int den;
    cout<<"Vnesi reden broj na denot: ";
    cin>>den;
    switch (den)
    {
        case 1:
            cout<<"Denes e ponedelnik!";
        break;
        case 2:
            cout<<"Denes e vtornik!";
        break;
        case 3:
            cout<<"Denes e sreda!";
        break;
        case 4:
            cout<<"Denes e cetvrtok!";
        break;
        case 5:
            cout<<"Denes e petok!";
        break;
        case 6:
            cout<<"Denes e sabota!";
        break;
        case 7:
            cout<<"Denes e nedela!";
        break;
        default:
        cout<<"Pogresen podatok!";
    }
    system ("PAUSE");
    return 0;
}
```

2.

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    char ocenka;
    cout<<"Vnesi opisna ocenka: ";
    cin>>ocenka;
    if (ocenka=='A')
        cout<<"Imas 5!";
    else if (ocenka=='B')
        cout<<"Imas 4!";
    else if (ocenka=='C')
        cout<<"Imas 3!";
    else if (ocenka=='D')
        cout<<"Imas 2!";
    else if (ocenka=='E')
        cout<<"Imas 1!";
    else
        cout<<"Pogresen podatok!";
    system("PAUSE");
    return 0;
}
```

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    char ocenka;
    cout<<"Vnesi opisna ocenka: ";
    cin>>ocenka;
    switch (ocenka)
    {
        case 'A':
            cout<<"Imas 5!";
        break;
        case 'B':
            cout<<"Imas 4!";
        break;
        case 'C':
            cout<<"Imas 3!";
        break;
        case 'D':
            cout<<"Imas 2!";
        break;
        case 'E':
            cout<<"Imas 1!";
        break;
        default:
            cout<<"Pogresen podatok!";
}
    system("PAUSE");
    return 0;
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int mesec;
    char prestapna;
    cout<<"Vnesi reden broj na mesecot: ";
    cin>>mesec;
    switch (mesec)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            cout<<"Mesecot ima 31 den";
        break;
        case 4:
        case 6:
        case 9:
        case 11:
```

```
            cout<<"Mesecot ima 30 dena";
        break;
        case 2:
            cout<<"dali godinata e prestapna? (d/D)";
            cin>>prestapna;
            if (prestapna == 'd' || prestapna == 'D')
                cout<<"Mesecot ima 29 denovi";
            else
                cout<<"Mesecot ima 28 denovi";
        break;
        default:
            cout<<"Pogresen podatok!";
    }
    system("PAUSE");
    return 0;
}
```

4.8 Основна структура за повторување

4.8.1 и 4.8.2 Структури за повторување на циклус до исполнување на услов

Прашања:

5. а) 2 4 6 8 10 б) ништо нема да се прикаже
6. а) 2 4 6 8 10 б) 20
7. а) еднаш б) шест пати
8.

```
int x=1;
while (x<=10)
{
        cout<<"*";
        x++;
}
```

9.

```
2   do                                  7   (proizvod <100000);
5   proizvod = proizvod *x;             3   {
4   cin<<x;                             1   proizvod=1;
6   while                               8   }
```

Задачи:

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i, n;
    float s;
    cout<<"n= ";
    cin>>n;
    s=0;
    i=1;
    while (i<=n)
    {
        s+=1./i;
        i++;
    }
    cout<<"s= "<<s;
    system("PAUSE");
    return 0;
}
```

2.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
      int i, m, n, s, brojac;
      cout<<"Od koj broj da sobiram? ";
      cin>>m;
      cout<<"Do koj broj da sobiram? ";
      cin>>n;
      brojac=s=0;
      i=m;
      while (i<=n)
      {
            if (i%2==0)
            {
                  brojac=brojac+1;
                  s=s+i;
            }
            i++;
      }
      cout<<"brojac= "<<brojac<<endl;
      cout<<"s= "<<s<<endl;
      system("PAUSE");
      return 0;
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
      int i, n, broj, bp, bn;
      cout<<"Kolku broevi ke vnesesh? ";
      cin>>n;
      bp=bn=0;
      i=1;
      while (i<=n)
      {
            cout<<"Vnesi broj: ";
            cin>>broj;
            if (broj>0)
                  bp=bp+1;
            else
                  bn=bn+1;
            i++;
      }
      cout<<"ima "<<bp<<" pozitivni";
      cout<<" i "<<bn<<" negativni broevi";
      system("PAUSE");
      return 0;
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i, broj, b, s;
    b=s=0;
    i=1;
    while (s<100)
    {
        cout<<"Vnesi broj: ";
        cin>>broj;
        s+=broj;
        b++;
    }
    cout<<"Vneseni se "<<b<<" broevi";
    system("PAUSE");
    return 0;
}
```

5.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n, cifri;
    cout<<"Vnesi broj! ";
    cin>>n;
    cout<<"Brojot "<<n<<" ima ";
    cifri=0;
    while (n!=0)
    {
        n/=10;
        cifri++;
    }
    cout<<cifri<<" cifri";
    system("PAUSE");
    return 0;
}
```

n	cifri	n != 0
12345/10=1234	1	1
1234/10=123	2	1
123/10=12	3	1
12/10=1	4	1
1/10=0	5	0

6.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n;
    cout<<"Vnesi broj! ";
    cin>>n;
    cout<<"Cifrite na brojot ";
    cout<<n<<" nanazad se ";
    while (n>0)
    {
        cout<<n%10<<", ";
        n/=10;
    }
    system("PAUSE");
    return 0;
}
```

7.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int broj, najgolem;
    cout<<"Vnesi broj! ";
    cin>>broj;
    najgolem=broj;
    do
    {
        cout<<"Vnesi broj! ";
        cin>>broj;
        if (broj>najgolem)
        najgolem=broj;
    }
    while (broj!=0);
    cout<<"Najgolemiot broj e "<<najgolem;
    system("PAUSE");
    return 0;
}
```

Објаснување:

На променливата `najgolem` (најголемиот број) и доделуваме вредност на првиот внесен број. Засега тој број е најголем. За секој следен број што ќе се внесе проверуваме дали тој број е поголем од до сега најголемиот број (`najgolem`). Ако е поголем тогаш тој број се зачувува во променливата `najgolem`.

4.9 Останати структури за повторување

4.9.1 Структура за повторување на циклус со броење на циклусите

Прашања:

2. а) 10 пати б) 5 пати в) 11 пати г) 6 пати

3. а) 10 б) ништо (бесконечен циклус)

4. а) AnaAna б) ништо (бесконечен циклус)

5.
```
for (int i=1;i<=10;i++)
    if(i<5 && i!=2 )
        cout<<'x';
```

6.
```
for (int n = 100;n>0;n=n-10)
cout<<'x';
```

Задачи:

1.
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n;
    for (n=100;n<=999;n++)
        if(n%10==0)
            cout<<n<<",";
    system("PAUSE");
    return 0;
}
```

2.
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float x, stepen;
    int i, n;
    cout<<"Vnesi gi x i n! ";
    cin>>x>>n;
    stepen=1;
    for (i=1;i<=n;i++)
        stepen = stepen * x;
    cout<<stepen<<endl;
    system("PAUSE");
    return 0;
}
```

3.
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float broj;
    int i, bp, bn;
    bp=bn=0;
    for (i=1;i<=10;i++)
    {
        cout<<"Vnesi broj! ";
        cin>>broj;
```

```
            if (broj>0)
                bp = bp+1;
            if (broj<0)
                bn = bn+1;
        }
        if (bp==bn)
        {
            cout<<"Vneseniot e ist broj na pozitivni";
            cout<<" i na negativni broevi";
        }
            else if (bp>bn)
            cout<<"Ima poveke pozitivni broevi";
        else
            cout<<"Ima poveke negativni broevi";
        system("PAUSE");
        return 0;
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i, m, n, l, b;
    b=0;
    cout<<"Vnesi broj! ";
    cin>>m;
    cout<<"Vnesi broj! ";
    cin>>n;
    if (m>n)
    {
        l=m; m=n; n=l;
    }
    for (i=m;i<=n;i++)
    if (i%2==0)
        b = b+1;
    cout<<"Od "<<m<<" do "<<n<<" ima ";
    cout<<b<<" parni broevi";
    system("PAUSE");
    return 0;
}
```

5.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int brojac,n;
    cout<<"Vnesi prirodni broj: ";
    cin>>n;
    cout<<"Delitelite na brojot "<<n<<" se: ";
    for (brojac=1;brojac<=n/2;brojac++)
    {
        if(n%brojac==0)
            cout<<brojac<<", ";
    }
```

```
15      cout<<n<<endl;
16      system("PAUSE");
17      return 0;
18  }
```

4.9.2 Дополнителни структури за повторување

Задачи:

1.
```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       int broj,n;
7       cout<<"Broj\tDeliteli\n";
8       cout<<"                              ";
9       for (n=10;n<=100;n++)
10      {
11           cout<<"\n"<<n<<"\t";
12           for (brojac=1;brojac<=n/2;brojac++)
13           {
14                if(n%brojac==0)
15                     cout<<brojac<<", ";
16           }
17           cout<<n;
18      }
19      system("PAUSE");
20      return 0;
21 }
```

2.
```
1  #includes <iostream>
2  #includes <cstdlib>
3  using namespace std;
4  int main()
5  {
6       int broj, s, brojac,n;
7       brojac=s=0;
8       for (n=1;n<=10;n++)
9       {
10           cout<<"Vnesi broj ";
11           cin>>broj;
12           if (broj<0)
13           {
14                cout<<"Vnesenoto e negativen broj!\n";
15                break;
16           }
17           s+=broj;
18           brojac++;
19      }
20      cout<<"Vneseni se "<<brojac;
21      cout<<" broevi, nivniot zbir e "<<s;
22      system("PAUSE");
23      return 0;
24  }
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n, i;
    cout<<"Vnesi n: ";
    vnes: cin>>n;
    if (n<1)
    {
        cout<<"Vnesi broj pogolem od 1 ";
        goto vnes;
    }
    for (i=1;i<=n;i++)
        cout<<i<<", ";
    system("PAUSE");
    return 0;
}
```

4.11. Задачи за талентираните ученици

4.11.1. Линеарна структура

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int k;
    cout<<"Vnesi ja dolzinata na prvata slica: ";
    cin>>k;
    cout<<"Vkupnata dolzina na sticite iznesuva ";
    cout<<6*k+2*20+2*40;
    system("PAUSE");
    return(0);
}
```

2.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int t;
    cout<<"Vnesi go brojot na triagolnicite: ";
    cin>>t;
    cout<<"Treba da se napravat ";
    cout<<t+2<<" jazli";
    system("PAUSE");
    return(0);
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int c1, c2, c3;
    cout<<"c1= ";
    cin>>c1;
    cout<<"c2= ";
    cin>>c2;
    cout<<"c3= ";
    cin>>c3;
    cout<<"Prviot. patnik treba da plati ";
    cout<<c1/3.<<" denari"<<endl;
    cout<<"Vtoriot patnik treba da plati ";
    cout<<c1/3.+(c2-c1)/2.<<" denari"<<endl;
    cout<<"Tretiot patnik treba da plati ";
    cout<<c1/3.+(c2-c1)/2.+c3-c2<<" denari"<<endl;
    system("PAUSE");
    return(0);
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n, b1000, b500, b100, b50, b10;
    cout<<"Kolku denari? n= ";
    cin>>n;
    cout<<"%a "<<n<<" denari bankomatot ke isplati ";
    b1000=n/1000;
    n=n%1000;
    b500=n/500;
    n=n%500;
    b100=n/100;
    n=n%100;
    b50=n/50;
    n=n%50;
    b10=n/10;
    cout<<b1000+b500+b100+b50+b10<<" banknoti.";
    system("PAUSE");
    return(0);
}
```

4.11.2. Разгранета структура

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int d, c1, c2, m, c;
    cout<<"Dimenzija na sobata d= ";
    cin>>d;
    cout<<"Dimenzii na masata c1 i c2: ";
    cin>>c1>>c2;
```

```
11      cout<<"Rastojanie od masata do gradovite: ";
12      cin>>m;
13      if (c1>c2)c=c1;
14      else c=c2;
15      if(c+d-2*m)
16          cout<<"moze";
17      else
18          cout<<"ne moze";
19      system("PAUSE");
20      return(0);
21  }
```

2.

```
1   #include <iostream>
2   #include <cstdlib>
3   using namespace std;
4   int main()
5   {
6       int x;
7       cout<<"x= ";
8       cin>>x;
9       if (x%5!=0)
10          cout<<x/5*5<<" "<<x/5*5+5;
11      else
12          cout<<x/5*5-5<<" "<<x/5*5+5;
13      system("PAUSE");
14      return(0);
15  }
```

3.

```
1   #include <iostream>
2   #include <cstdlib>
3   using namespace std;
4   int main()
5   {
6       int ora;
7       cout<<"ora= ";
8       cin>>ora;
9       if (ora ==0) cout<<"12 AM";
10      if (ora==12) cout<<"12 PM";
11      if (ora>=1 && ora<=11)
12          cout<<ora<<" AM";
13      if (ora>=13 && ora<=23)
14          cout<<ora-12<<" PM";
15      system("PAUSE");
16      return(0);
17  }
```

4.11.3. Циклични структури

1.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i, n, l;
    int vkupno=0;
    cout<<"n= ";
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"l= ";
        cin>>l;
        if(l%4==0)
            vkupno+=l/4;
        else
            vkupno+=(l/4+1);
    }
    cout<<vkupno;
    system("PAUSE");
    return(0);
}
```

2.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i, n, p, poeni;
    int vkupno=0;
    cout<<"n= ";
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"poeni= ";
        cin>>poeni;
        vkupno+=poeni;
    }
    cout<<"potrebni poeni ";
    cin>>p;
    if (vkupno>p)
        cout<<"da";
    else
        cout<<"ne";
    system("PAUSE");
    return(0);
}
```

3.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n, p;
    p=0;
    cout<<"n= ";
    cin>>n;
    while (n>0)
    {
        p=p*10+n%10;
        n=n/10;
    }
    cout<<p;
    system("PAUSE");
    return(0);
}
```

4.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b, d;
    d=0;
    cout<<"a= ";
    cin>>a;
    cout<<"b= ";
    cin>>b;
    while (a>0 && b>0)
    {
        if (a>b)
            a--b;
        else
            b--a;
        d++;
    }
    cout<<d;
    system("PAUSE");
    return(0);
}
```

5.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int m, n, a;
    cout<<"m= ";
    cin>>m;
    cout<<"n= ";
    cin>>n;
    a=m*n;
    while (m!=n)
    {
        if (m>n)
            m--n;
        else
            n--m;
    }
    cout<<a/m;
    system("PAUSE");
    return(0);
}
```

Додаток А

Ознака	Опис	Големина	Опсег
bool	две вредности – точно (true) и погрешно (false)	1 бајт	true и false
char	може да се користи за чување знаци или цели броеви	1 бајт	-128 до 127, но може да биде и 0-255 (кај некои архитектури). Кога сакате да чувате цели броеви, секогаш користете usigned char или signed char.
signed char	исто како char, но гарантирано може да чува и негативни броеви	1 бајт	-128 до 127
unsigned char	исто како char, но гарантирано може да чува само позитивни броеви	1 бајт	0 до 255
short short int signed short signed short int	цели броеви (позитивни и негативни)	2 бајти	-32768 до 32767
unsigned short unsigned short int	позитивни цели броеви	2 бајти	0 до 65535
int signed int	цели броеви (позитивни и негативни)	4 бајти	-2147483648 до 2147483647
unsigned unsigned int	позитивни цели броеви	4 бајти	0 до 4294967295
long long int signed long signed long int	цели броеви (позитивни и негативни)	4 бајти	-2147483648 до 2147483647
unsigned long unsigned long int	позитивни цели броеви	4 бајти	0 до 4294967295
long long long long int signed long long signed long long int	цели броеви (позитивни и негативни)	8 бајти	-9223372036854775808 до 9223372036854775807
unsigned long long unsigned long long int	позитивни цели броеви	8 бајти	0 до 18446744073709551615
float	децимални броеви	4 бајти	околу 7 точни цифри
double	децимални броеви, двојна прецизност	8 бајти	околу 15 точни цифри
long double	децимални броеви	10 (или 12) бајти	околу 22 точни цифри

Користена литература:

- Burks, Arthur W.; Goldstine, Herman; von Neumann, John (1947),Preliminary discussion of the Logical Design of an Electronic ComputingInstrument , Princeton, NJ: Institute for Advanced Study, retrieved 2008-05-18
- Raul R., Ulf H, 2000, The First Computers, history and architecture, MITPress, Cambridge, Massachusetts, London, England
- Operating System Concepts by Abraham Silberschatz, Peter B. Galvin and Greg Gagne (Jul 29, 2008)
- The Tech Contracts Handbook: Software Licenses and Technology Services Agreements for Lawyers and Businesspeople
- Microsoft Office 2010 Plain and Simple (Plain & Simple) by Katherine Murray
- C++ Primer Plus (5th Edition) by Stephen Prata
- The Internet Book: Everything You Need to Know About Computer Networking and How the Internet Works (3rd Edition) by Douglas E. Comer.