АВТОР

м-р Андријана Томовска

ИНФОРМАТИКА

ЗА VII ОДДЕЛЕНИЕ ЗА ДЕВЕТГОДИШНО ОСНОВНО ОБРАЗОВАНИЕ

Автор: м-р Андријана Томовска

Издавач:Министерство за образование и наука на
Република Северна Македонија

Рецензенти: Проф.д-р Марика Апостолова Трпковска
Жаклина Наумовска
Сенар Фазлија

Илустратор: Елена Трпчевска

Лектор: Слаѓана Гиновска

Со одлука за одобрување на учебникот по предметот Информатика за VII одделени во деветгодишно основно образование бр.26- 521/1 од 16.03.2020 донесена од Националната комисија за учебници.

CIP - Каталогизација во публикација
Национална и универзитетска библиотека "Св. Климент Охридски", Скопје

373.3.016:004(075.2)=163.3

ТОМОВСКА, Андријана
Информатика за VII одделение за деветгодишно основно образование / Андријана Томовска. - Скопје : Министерство за образование и наука на Република Северна Македонија, 2020. - 162 стр. : илустр. ; 30 см

Библиографија: стр. 162. - Речник на клучни поими : (стручна терминологија): стр. 157-159

ISBN 978-608-226-864-4

COBISS.MK-ID 51870981

ПРЕДГОВОР

Согласно со наставната програма креирана и дефинирана од Бирото за развој на образованието и одобрена од страна на Министерството за образование и наука на Република Северна Македонија, креиран е овој учебник по Информатика за VII одделение за деветгодишно задолжително основно образование. Според официјалната наставна програма учебникот брои пет теми.

Во првата тема „Програма за табеларно пресметување“ учениците ќе се запознаат со работната околина на Microsoft Office Excel 2016 и ќе се оспособат за креирање и уредување табели, вршење на математички пресметки со примена на формули и функции и на крајот графички приказ на податоците. За практично увежбување на овие функционалности на програмата за табеларни пресметки креирани се соодветни прашања и практични задачи.

Следната тема која ќе се обработува е „Запознавање со информатичките концепти преку решавање логички натпреварувачки задачи“ во која учениците со помош на конкретни задачи ќе го развијат логичкото размислување, со цел изнаоѓање на соодветно решение. При решавањето на логичките натпреварувачки задачи учениците ќе ја осознаат врската помеѓу логичкото размислување и решавање со информатичките концепти и нивното место при работа со компјутерите.

Во третата тема „Напредно програмирање во визуелна околина“ ќе се направи линк со претходната тема, во која со помош на логиката ќе се креираат блокови од искази кои ќе извршуваат определена активност. На тој начин, применувајќи ја визуелната околина за програмирање Scratch ќе креираме едноставни програми, интерактивни игри, приказни и слично.

Четвртата тема е посветена на: „Програмирањето преку стандарден структурен програмски јазик“. За таа цел ќе ја користиме интегрираната околина за програмирање Code::Blocks и програмскиот јазик C++. Преку низа примери и практични задачи учениците ќе се стекнат со вештини за креирање на програми.

Во последната тема: „Онлајн живеење“ учениците ќе се оспособат за креирање веб-дневници, односно блогови со посебен акцент на почитување на правилата за етичко користење и безбедност на Интернет.

Во овој учебник практичните изработки се основата за усвојување и совладување на новите наставни единици и токму поради тоа изготвен е посебен дел со прашања и задачи кои ќе му овозможат на ученикот проширување на знаењата и вештините по наведените теми. Секоја наставна единица започнува со истакнување на новите клучни поими со

со кои ќе се среќаваат учениците при учењето, а завршува со запомни како рубрика кои укажуваат на најважните елементи од наставната единица. По секоја лекција наведни се прашања кои служат за повторување на наученото. Низ лекциите учениците ќе се среќаваат со потсетување, совети, забелешки кои се ставени во посебни рамки и се посебно графички означени. Во прилог, листа со значењата на графичките елементи кои се употребени во учебникот.

На крајот од секоја тема креирана е рубрика „Да повториме!, Да увежбаме!“, како посебен додаток со прашања и задачи по темите.

На крајот на учебникот креиран е мал речник со објаснувања на стручните поими и прилози кои се потребни при усвојувањето на наставните единици. Во прилог на учебникот креирано е CD со сите практични изработки дефинирани по фолдери на темите.

Вежбите, задачите, практичните изработки и прашањата изготвени се согласно со возраста на учениците во VII одделение во деветгодишното основно образование и согласно со нивните предзнаења стекнати во претходните учебни години во рамките на овој наставен предмет.

Авторот

СОДРЖИНА

Програма за табеларни пресметки

Вовед во програмата за табеларни пресметки
Програма за табеларни пресметки – Microsoft Office Excel 2016
Работа со податоци во табелата
Уредување табела
Форматирање табела
Автоматско пополнување податоци во табелата
Формули и функции во програмата за табеларни пресметки
Сортирање податоци
Креирање графикон
ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

1.Вовед во програмата за табеларни пресметки

Што ќе научиме?

Да креираме табели со различен тип на податоци и низи од пресметки со примена на формули и функции, уредени со алатките за форматирање и графички претставени со графикони.

Луѓето во секојдневието честопати се соочуваат со најразлични податоци кои се поставени во табела. На пример: наставниците во училиштата креираат табели за да ги следат и забележуваат постигнувањата на учениците по некој предмет, да можат да пресметаат просечен успех по ученик и воопшто на целото одделение, да можат да прават споредби на постигнувањата по тромесечија, да изработат распоред на часови и сл. Вработените во училишната библиотека може да креираат табела со податоци за книгите со кои располага училиштето. Учениците ќе можат да ја употребуваат оваа програма за различни училишни и вонучилишни цели, како на пример: креирање табела со податоци на учениците од одделението, креирање телефонски именик, креирање табела за редовност во училиште, преглед на редовноста на домашните задачи, прибирање и обработка на податоци при креирање на проектна задача, графичко претставување на прибраните податоци и сл. Родителите можат да ја употребуваат оваа програма за водење евиденција за приходите во семејството, месечните трошоци и сл.

Според тоа, целта на изучувањето на темата „Програма за табеларни пресметки **MS Office Excel 2016** е работа со табеларни податоци, вршење пресметки според определени функции и креирани формули, како и графички приказ на податоците.
Учениците при изучувањето на оваа тема ќе се оспособат за:

- креирање табеларен документ;
- внесување податоци од различен формат;
- уредување/форматирање табела;
- вршење пресметки употребувајќи формули и функции;
- сортирање податоци во табела;
- филтрирање податоци во табела;
- креирање графикони од различен тип за претставување на податоци.

Покрај теоретското усвојување на поимите и материјалот во целина, посебен акцент ќе ставиме на практичната работа. За таа цел, потребно е на компјутерот да се инсталира канцелариското пакет Microsoft Office 2016.

Слика 1: Икона на Microsoft Excel 2016

Клучни поими

работна тетратка, работен лист, табела, колона, редица, ќелија, последователни/непоследователни ќелии, активна ќелија, формула, функција, графикон.

1.1 Програма за табеларни пресметки – Microsoft Office Excel 2016

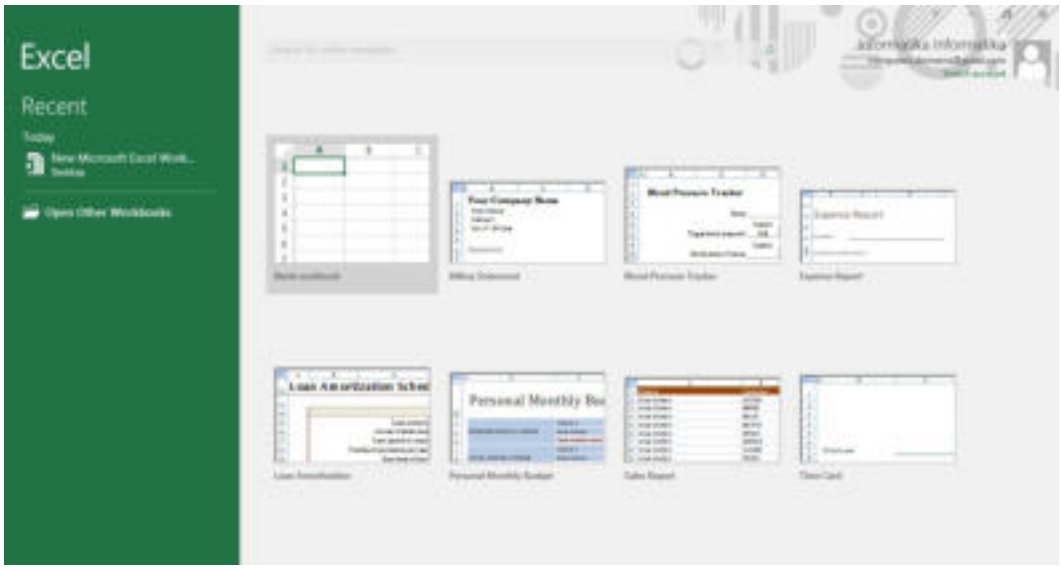
Што ќе научиме?

Поимот табела го запознавме минатата учебна година при изучување на темата „Програма за обработка на текст“. Табела претставува збир на податоци кои се организирани во вертикални делови кои се викаат колони и хоризонтални делови кои се викаат редици. Секоја колона и редица се пресекуваат и нивниот пресек се нарекува ќелија или клетка.

Апликацијата **Microsoft Office Excel 2016** претставува дел од канцелариското пакет на Microsoft Office 2016 и овозможува претставување на податоците на табеларен начин, уредување табели, вршење табеларни пресметки и графички приказ на податоците, креирање на прегледни и професионални извештаи и сл.

1.1.1 Стартување на програмата. Елементи на работниот прозорец

За креирање на еден табеларен документ ја стартуваме програмата Microsoft Office Excel 2016 преку старт менито и избор на соодветната икона која ја претставува програмата. Притоа, се појавува следниот прозорец:



Слика 1: Стартување нов документ во Excel 2016

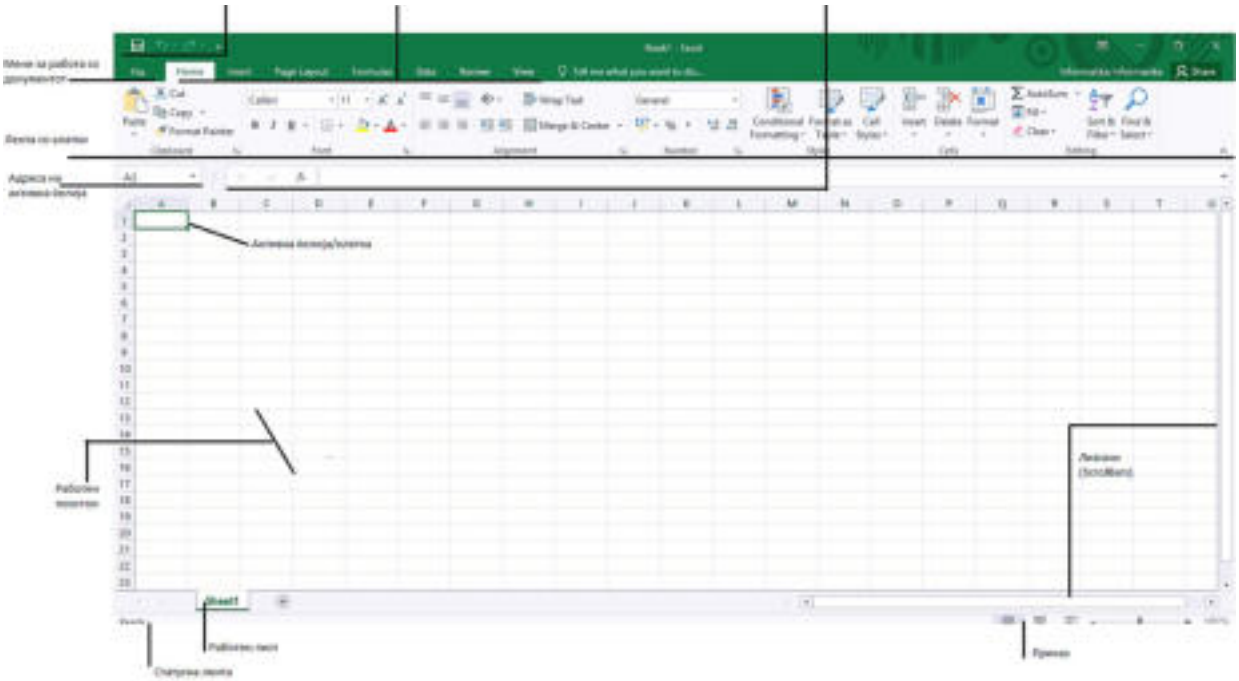
Овој прозорец се дели на два дела. На левата страна на прозорецот прикажана е листа со документи кои корисникот неодамна ги отворил и имаат својство на линкови, односно со кликнување на името на документот веднаш се активира тој документ. Од десната страна на прозорецот корисникот може да отвори Blank Workbook, односно нов или празен работен документ или, пак, да избере готов образец (Template) на документ за преработка или доработка.

Да се потсетиме!

Што се документи? Како уште се викаат документите? Што е карактеристично за нив?

Документите кои се креираат во програмата MS Excel 2016 се викаат работни тетратки (Workbook) и добиваат наставка на .xlsx. Секоја работна тетратка се состои од работни листови (Worksheet). Секој работен лист се состои од табели. Вертикалните делови на табелата се викаат колони и се означуваат со букви (A, B, C.....Z, AA, AB, AC...), а хоризонталните делови се викаат редици и се означуваат со бројки (1,2,3...). Ќелија е местото каде што се внесуваат податоците. Секоја ќелија има име кое е составено со буквата на колоната и бројот на редот во која се наоѓа. На пример: A1, B5, D16 и сл. Секој работен лист се состои од 256 колони и 65 536 редици.

При избор на **Blank Workbook** се активира работниот прозорец на програмата **Microsoft Office Excel 2016**. На следната слика прикажани се деловите на работниот документ на програмата.



Слика 2: Работен прозорец на MS Excel 2016

1.**Quick Access Toolbar** – е лента за брз пристап до наредбите кои најчесто се користат. Истата може да се модифицира според потребите на корисникот.

2.**Лента со менија (Menu Bar)** – е мени-лентата која претставува групирање на командите според нивните задачи, всушност, лента со наредби за соодветна картичка.

3.**Лента со алатки** - се појавува или менува зависно кое мени е избрано.

4.**Адреса на активна ќелија (Name Box)** – претставува поле за приказ на името на ќелијата која е селектирана, односно ја прикажува активната ќелија.

5.**Formula Bar** – е лента за внесување формули и функции, приказ на податоците или содржината на активната ќелија.

6.**Активна ќелија** – е ќелијата во која корисникот внесува податоци.

7.**Sheet Tabs** – е лента со работни листови.

8.**Scroll Bar** – се ленти за движење низ работниот прозорец: хоризонтална и вертикална.

9.**Статусна лента (Status Bar)** – се наоѓа на долниот дел од прозорецот и истата може да се прилагодува за приказ на повеќе опции, од кои повеќето му даваат на корисникот информации за тековниот лист.

10.**Мени File** – ги содржи основните наредби за работа со документот во Excel.

1.1.2 Работа со документи во Excel

Постапката за работа со документите во Excel е сосема иста како и во другите програми на канцелариският пакет на Microsoft Office, односно сите се задаваат преку менито File. Но, ајде да се потсетиме на основните команди и операции со документи:

Активност/операција	Значење на активноста/операцијата
New	Отворање нов документ
Open	Отворање постоечки документ
Save	Зачувување документ
Save As	За креирање нова копија од документот на друга локација или со друго име
Print	Печатење документ
Share	За споделување на документот со повеќе корисници или, пак, да се испрати преку е-маил
Export	За менување на типот на документот или конвертирање на документот во .pdf или .xps формат
Close	За затворање на документот

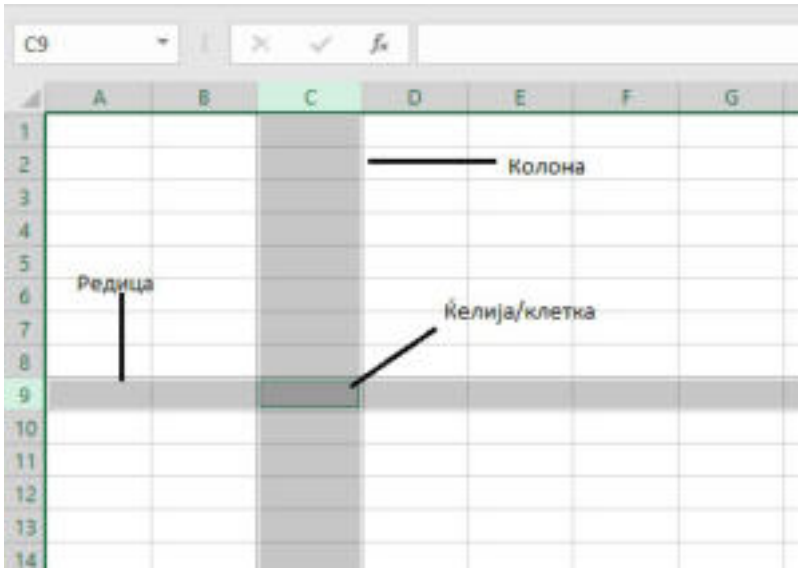
Табела 1: Активности/операции со работниот документ во Excel 2016

Забелешка!

За пресекот на колоните и редиците во овој учебник ќе го користиме зборот ќелија. Иако, во различна литература можат да се сретнат и под името ќелија и под името клетка.

1.1.3 Работа со основните елементи на работниот лист

На следниот прозорец прикажани се основните елементи на работниот лист: табела, редица, колона и ќелија:



Слика 3: Основни елементи на работниот прозорец

За да започнеме со внесување податоци, најпрво означуваме ќелија и во неа ги внесуваме податоците. Таа ќелија која е означена и подготвена за внес на податоците се вика активна ќелија. На пример:

Задача

Креирајте фолдер на работната површина на компјутерот. Именувајте го со вашето име и презиме. Во овој фолдер ќе ги зачувувате вашите практични изработки.

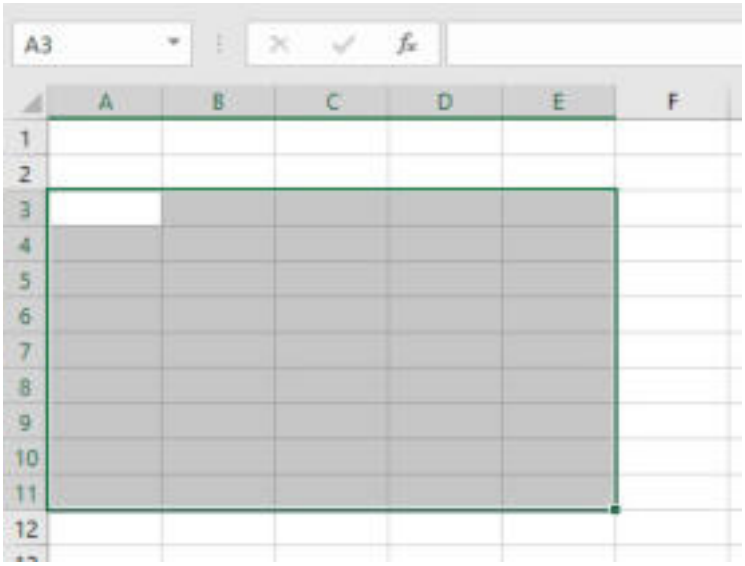
Отворете нов работен документ.
Кликнете во ќелија A1 и внесете податок: вежба 1.
Кликнете во ќелија A2 и внесете податок: вашето име.
Кликнете во ќелија B2 и внесете го вашето одделение.
Зачувајте го документот со име „**Vezba1.xlsx**“ во фолдерот кој е креиран на работната површина.

При работа со елементите во работниот лист секогаш треба да ги селектираме или означиме истите. Селектирање или означување, значи да прикажеме врз кои елементи ќе вршиме активности, како на пример: избор на фонт, големина на фонт, боја на фонт, рамки и сл., како и копирање, отсекување, бришење и др.

Една ќелија се означува со кликнување врз неа. Зависно од потребата некогаш можеме да селектираме повеќе ќелии одеднаш. **Соседните ќелии** ги селектираме така што кликнуваме на првата ќелија која сакаме да биде означена, го држиме копчето **Shift** од тастатурата која

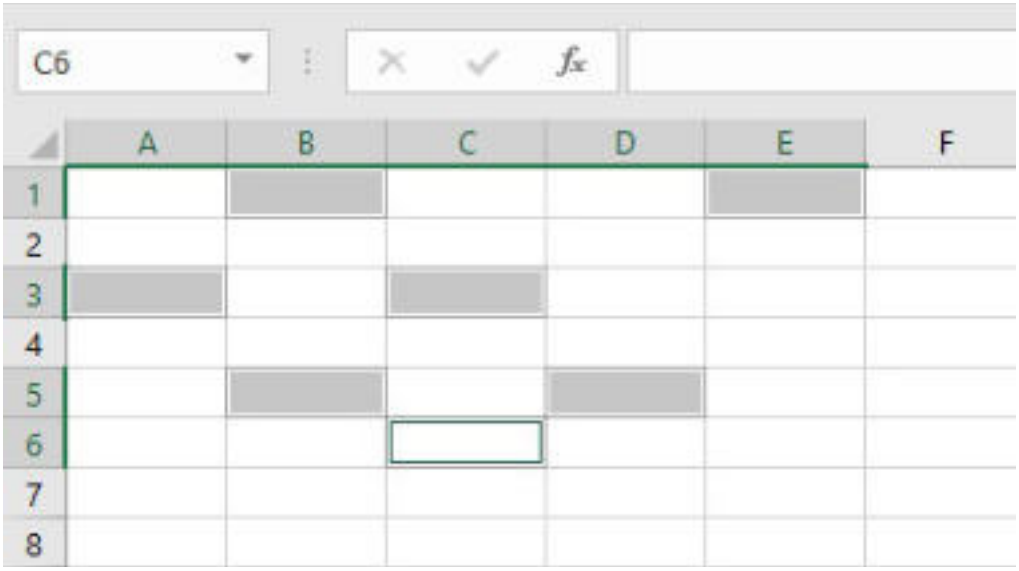
сакаме да биде означена, го држиме копчето **Shift** од тастатурата и кликнуваме на последната ќелија или, пак, со кликнување на првата ќелија со глувчето и влечеме до последната ќелија. На тој начин сме означиле ранг од ќелии.

Рангот од ќелии претставува збир на последователни или соседни ќелии и се означуваат на следниот начин, на пример: A3:E11.



Слика 4: Селектирање последователни/сосоедни ќелии

Непоследователни или **несоседни ќелии**, можеме да ги селектираме така што кликнуваме на ќелијата, го држиме **Ctrl** копчето од тастатурата, па кликнуваме на следна ќелија итн.



Слика 5: Селектирање несоседни ќелии

За движење низ работниот документ ќе ги употребуваме лентите за движење (Scrool Bars) кои се наоѓаат на десната страна и во долниот дел на работниот прозорец, како и копчињата од тастатурата:

Копче од тастатура	Движење
Enter или ↓	Една ќелија подолу
↑	Една ќелија погоре
←	Една ќелија налево лево
→	Една ќелија надесно десно
Ctrl + ←	Првата ќелија во редицата
Ctrl + →	Последната ќелија во редицата
Home	Почеток на редот
End	Крај на редот
Ctrl + Home	Почеток на табелата
Ctrl + End	Крај на табелата

Табела 2: Движење низ работниот прозорец

Вежба 1

Во документот од вежба 1, во работниот лист Sheet1 да се селектираат ќелиите B4:F8 и со употреба на алатката Fill Color, од менито Home, да се обојат со сина боја. На работниот лист дајте му име: соседни. Промените кои се направени во документот да се зачуваат.

Во истиот документ додадете нов работен лист. Именувајте го работниот лист во несоседни. Селектирајте ги ќелиите: A2, C2, E2, G2, B3, D3, F3, A4, C4, E4, G4, B5, D5, F5, A6, C6, E6, G6, B7, D7, F7. Промените кои се направени во документот да се зачуваат.

Запомни!

Програмата **Microsoft Office Excel 2016** се употребува за претставување на податоците во табела, за вршење пресметки и графички приказ на податоците. Документите кои се креираат во Excel 2016 се викаат работни тетратки (Workbook) и имаат наставка .xlsx. Работните тетратки или документите содржат работни листови (Sheet1, Sheet2...). Работните листови содржат табели. Табела е збир податоци кои се организирани во

вертикални делови кои се викаат колони и хоризонтални делови кои се викаат редици. Секоја колона и ред се пресекуваат и нивниот пресек се нарекува ќелија или клетка. Најчести активности или операции со работниот документ се: отворање нов документ, отворање постоечки документ, зачувување документ и печатење документ.

Прашања

1.Зошто служи програмата Excel 2016?
2.Како се викаат документите кои се креирани во програмата за табеларни пресметки?
3.Кои се елементите на работниот прозорец на програмата за табеларни пресметки?
4.Како се означени хоризонталните делови во активниот документ во програмата за табеларни пресметки?
5.Како се вика пресекот на колона и редица во програмата за табеларни пресметки?

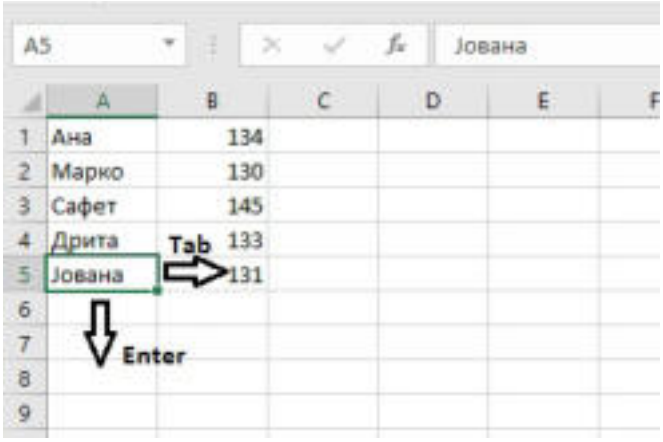
1.2 Работа со податоци во табелата

Забелешка!

Алфабетските податоци и алфа-нумеричките податоци се подредуваат на левата страна на ќелијата, додека нумеричките податоци се подредуваат во однос на десната страна на ќелијата. Секако, со алатките за форматирање можеме да ги подредуваме по желба.

За да креираме табела во програмата за табеларни пресметки, најпрво треба да внесеме податоци. Податоците во табелата се внесуваат во активната ќелија, а потоа продолжуваме со внесување така што ќе ги употребиме копчињата од тастатурата за движење низ работниот прозорец кои ги прикажавме во претходната наставна единица.

На пример, за да креираме листа на ученици и нивната висина ги извршуваме следните чекори:
1.кликнете на ќелијата A1 и внесете име на ученик;
2.притиснете **Enter** од тастатурата за да одите во ќелијата A2;
3.повторно внесете име и повторете ја постапката сè додека не внесете десет (10) имиња;
4.кога ќе го внесете последното име, кликнете на копчето **Tab** од тастатурата кое ќе ве однесе една ќелија надесно. Таму внесете ја висината на ученикот со бројка;
5.со помош на копчињата за движење низ работниот документ продолжете да внесувате податоци и за останатите ученици во листата;
6.зачувајте го документот во фолдерот на работната површина со име „**Vezba2.xlsx“.**



Слика 1: Внесување податоци во табела

Од практичната задача можеме да забележиме дека податоците кои ги внесуваме во активната ќелија се појавуваат и во **Formula Bar,** односно **лентата за формули**. Со кликнување врз која било ќелија, која содржи податок, адресата на ќелијата се појавува во **Name Box,** додека, пак, содржината на ќелијата се прикажува во **Formula Bar.**

1.2.1 Типови податоци

Податоците кои ги внесуваме во табелата можат да бидат **алфабетски, нумерички и алфа-нумерички**. Алфабетските податоци содржат само букви, односно текст, нумеричките податоци се низи од броеви, додека, пак, алфа-нумеричките се комбинација од букви/текст и бројки

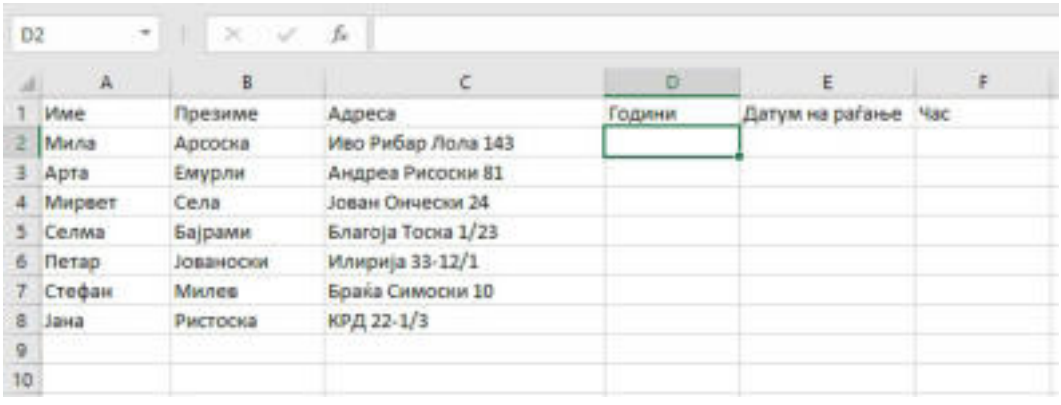
.

Алфабетски и алфа-нумерички податоци

За внесување на текстуални податоци **(алфабетски)** и комбинација од текст и броеви **(алфа-нумерички)** ќе ја изработиме следната задача. Во документот „**Vezba2.xlsx**“, да додадеме нов работен лист во кој ќе креираме табела: основни податоци за учениците. Преименувајте го работниот лист со име **„Податоци“**.

1.Во ќелиите A1, B1, C1, D1, E1, F1 и G1 ќе креираме насловни ќелии: име, презиме, адреса, години, датум на раѓање и час.

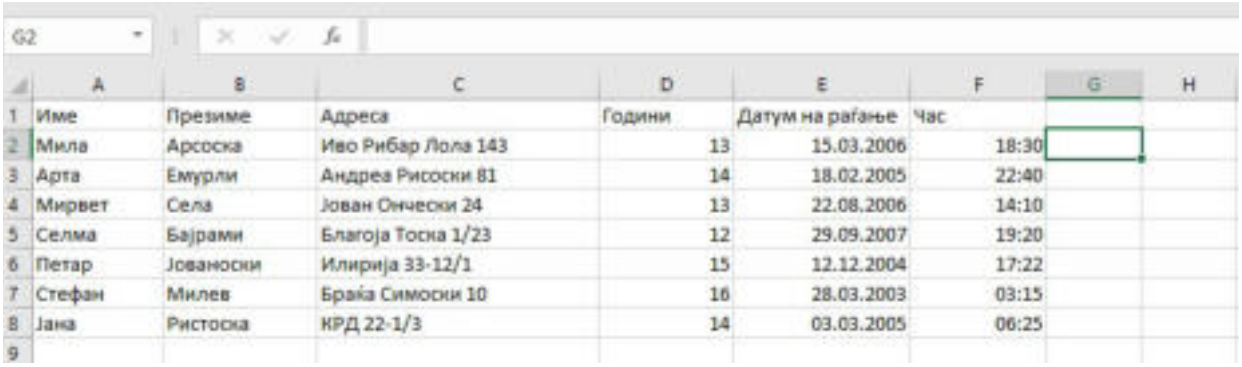
2.Употребувајќи ги копчињата за движење низ работниот документ, внесете ги податоците во колоните: име, презиме, адреса.



Слика 2: Внесување алфабетски и алфа-нумерички податоци

Нумерички податоци

Кога зборуваме за бројчани (нумерички) податоци, можеме да утврдиме дека постојат различни **типови бројни податоци**, како на пример: цели броеви, позитивни и негативни броеви, датум, час, валута и сл. Ајде да ја дополниме табелата **„Податоци“**. По внесувањето на сите податоци да ги зачуваме промените во документот.



	A	B	C	D	E	F	G	H
1	Име	Презиме	Адреса	Години	Датум на раѓање	Час		
2	Мила	Арсоска	Иво Рибар Лола 143	13	15.03.2006	18:30		
3	Арта	Емурли	Андреа Рисоски 81	14	18.02.2005	22:40		
4	Мирвет	Села	Јован Ончески 24	13	22.08.2006	14:10		
5	Селма	Бајрами	Благоја Тоска 1/23	12	29.09.2007	19:20		
6	Петар	Јованоски	Илирија 33-12/1	15	12.12.2004	17:22		
7	Стефан	Милев	Браќа Симоски 10	16	28.03.2003	03:15		
8	Јана	Ристоска	КРД 22-1/3	14	03.03.2005	06:25		
9								

Слика 3: Внесување нумерички податоци

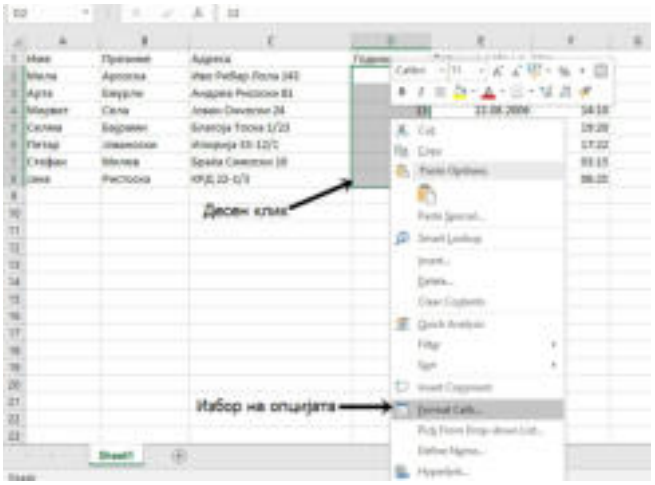
Оттука, можеме да забележиме дека при внесување датум денот го одвоивме од месецот со точка (.). Исто и месецот го одвоивме од годината со точка. При внесување датум можеме да користиме и коса црта (/) или, пак, само црта (-). Тогаш Excel овие податоци ги препознава како датуми. При внесување час, часовите ги одделуваме од минутите и од секундите со знакот две точки (:).

1.2.2 Формат на бројни (нумерички) податоци

Во табелите, при внесување на податоците можеме да избереме различен формат на податоци. Изборот на формат на податоците го изведуваме на следниот начин:

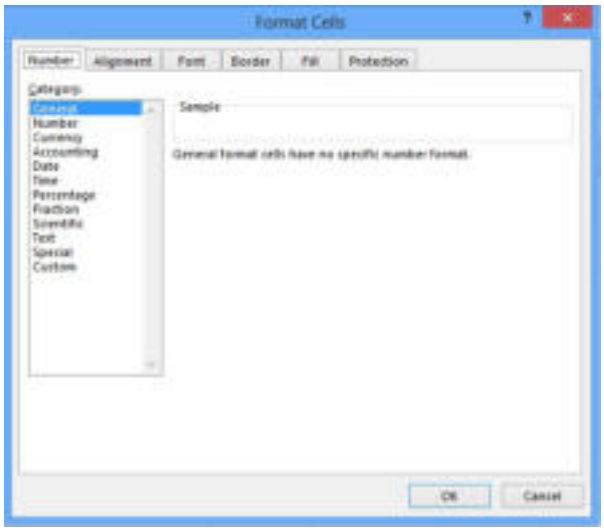
1.ги селектираме податоците чиј формат сакаме да го измениме;
2.десен клик врз селектираните ќелии и ја избираме опцијата **Format Cells.**

Притоа се појавува следниот прозорец:



Слика 4: Избор на опцијата Format Cells

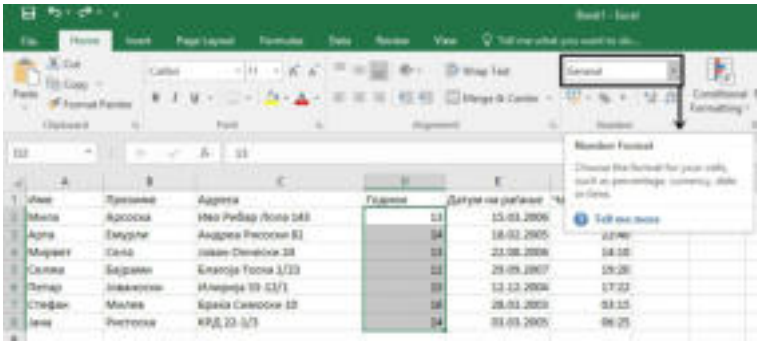
По изборот на опцијата Format Cells се појавува прозорецот од кој го избираме јазичето **Number,** а потоа избираме категории:



Слика 5: Прозорецот на Format Cells

1. **General** – општ формат.
2. **Number** – цели и децимални броеви, позитивни и негативни броеви.
3. **Currency** – формат на валута: денари, евра, долари, лири, фунти и сл.
4. **Accounting** – ги прикажува знаците на валутата од левата страна на вредноста и прикажува цртичка (-) при вредност нула (0).
5. **Date** – формати на датум.
6. **Time** – формати на време.
7. **Percentage** – приказ на формат на процент.
8. **Fraction** – формати на дропки.
9. **Scientific** – приказ на број во експоненцијален формат.
10. **Text** – низи од текст.
11. **Special** – специјални/посебни формати.
12. **Custom** – избор на дополнителни формати на податоци.

Освен преку прикажаната постапка за избор на различен формат на бројни податоци можеме да го употребуваме и менито Home и опцијата Number Format, како што е прикажано на следната слика:



Слика 6: Избор на формат на податоци преку менито Home

Ајде да избереме формат на датумот: дд.мм.гг и формат на време, односно час во кој ќе се прикажува часот и минутите, како и опцијата претпладне/попладне. Зачувајте ги промените во документот.

Запомни!

При креирање табела во програмата за табеларни пресметки податоците се внесуваат во активната ќелија. Постојат различни видови податоци: алфабетски, нумерички и алфанумерички. Нумеричките податоци можат да имаат различен формат: цел број, децимален број, позитивен и негативен број, датум, време, валута и сл. Форматите на податоците ги избираме преку опцијата Format Cells при десен клик со глувчето или, пак, преку опцијата Number Format од менито Home.

Прашања

1.Во која ќелија се внесува податок?
2.Кое копче или копчиња се кликнуваат по внесување на податок во ќелијата?
3.Набројразлични формати на бројни податоци!

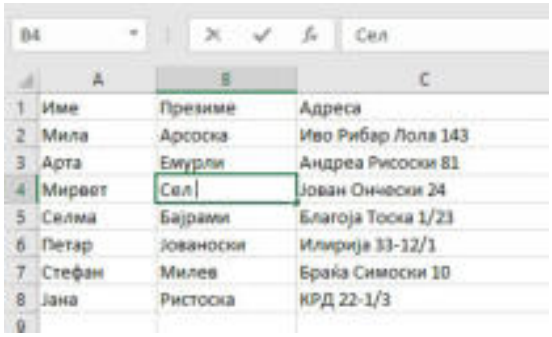
Експоненцијален формат значи заменување дел од бројот со E + n, во кој E (експонент) помножен со претходниот број од 10 на n-тиот степен. На пример, 2-децимални научен формат прикажува 12345678901 како 1.23E + 10, што е 1,23 пати 10 на 10-тиот степен.

1.3 Уредување табела

Кога креираме табела во програмата за табеларни пресметки на прв впечаток изгледа дека сè е во ред, сè додека нè провериме дали сме направиле техничка грешка, дали сме внеле погрешен податок, дали недостига цел ред или цела колона во табелата. Секако, ние нема да ја избришеме целата табела и одново да започнеме со креирање, туку ќе ги поправиме грешките кои се направени. За да направиме промени ќе ја отвориме **„Vezba2.xlsx“**.

1.3.1 Менување и уредување на внесените податоци

Доколку сакаме да направиме промена на податокот во ќелијата B4, односно да внесеме нов податок наместо постоечкиот, едноставно кликнуваме врз неа и почнуваме да пишуваме. Во тој момент на статусната лента ќе пишува **Ready**, а тоа значи дека сме во мод за внесување нов податок. На пример: наместо Села, ќе напишеме Селими.

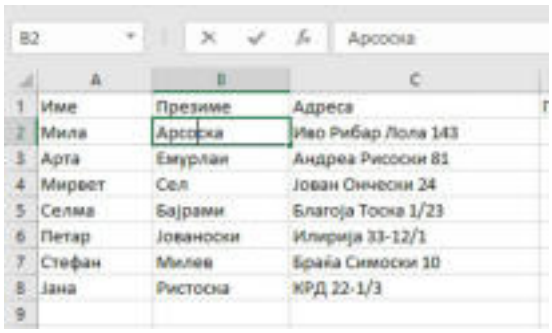


	A	B	C
1	Име	Презиме	Адреса
2	Мила	Арсоска	Иво Рибар Лола 143
3	Арта	Емурлаи	Андреа Рисоски 81
4	Мирвет	Сел	Јован Ончески 24
5	Селма	Бајрами	Благоја Тоска 1/23
6	Петар	Јованоски	Илирија 33-12/1
7	Стефан	Милев	Браќа Симоски 10
8	Јана	Ристоска	КРД 22-1/3
9			

Слика 1: Менување податок во ќелија

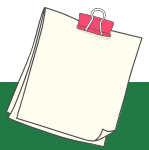
Исто така, во ќелијата B2 треба да додадеме еден знак, односно буква во презимето.

Тогаш, со глувчето двапати кликнуваме во ќелијата B2, го позиционираме курсорот на местото каде што треба да внесеме буква и од тастатура ја пишуваме буквата, во нашиов случај буквата „в“. Тогаш на статусната лента ќе пишува **Edit,** а тоа значи дека сме во мод за менување на податок.



	A	B	C
1	Име	Презиме	Адреса
2	Мила	Арсоска	Иво Рибар Лола 143
3	Арта	Емурлаи	Андреа Рисоски 81
4	Мирвет	Сел	Јован Ончески 24
5	Селма	Бајрами	Благоја Тоска 1/23
6	Петар	Јованоски	Илирија 33-12/1
7	Стефан	Милев	Браќа Симоски 10
8	Јана	Ристоска	КРД 22-1/3
9			

Слика 2: Додавање знак/буква при внесен



Забелешка!

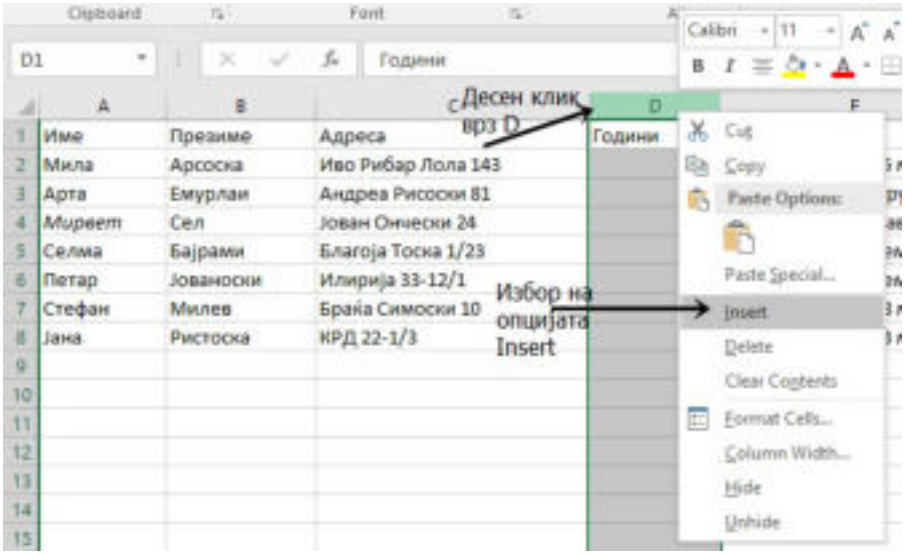
За брзо вметнување на повеќе од една колона или ред, селектирајте повеќе редови/колини, кликнете со десен клик врз нив и изберете ја опцијата Insert.

Податок или податоци од ќелиите можеме да избришеме така што најпрво ќе селектираме кои податоци сакаме да ги избришеме, а потоа кликнуваме на копчето **Delete** од тастатурата.

1.3.2 Активности со колони и редици

Во креираната табела можеме да додаваме и бришеме колони и редици, зависно од потребата на корисникот. Постапката за **вметнување нова колона** е следната:

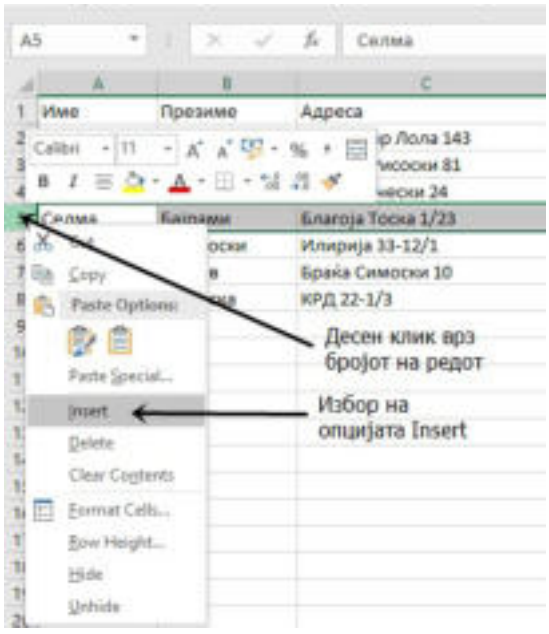
1. ја означуваме колоната пред која сакаме да додадеме нова колона;
2. правиме десен клик врз букавата на означената колона;
3. ја избираме опцијата **Insert**.



Слика 3: Вметнување колона во табела

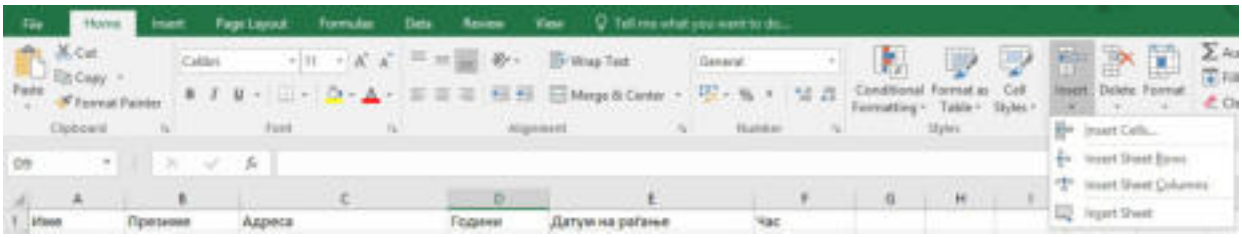
Иста е постапката и за **вметнување редица во табелата**:

1. ја означуваме колоната пред која сакаме да додадеме нов ред;
2. правиме десен клик врз бројот на означениот ред;
3. ја избираме опцијата **Insert**.



Слика 4: Вметнување редица во табела

Вметнувањето колона и редица во табела може да се изврши и преку менито **Hom**е и избор на соодветните алатки за вметнување колона и редица:



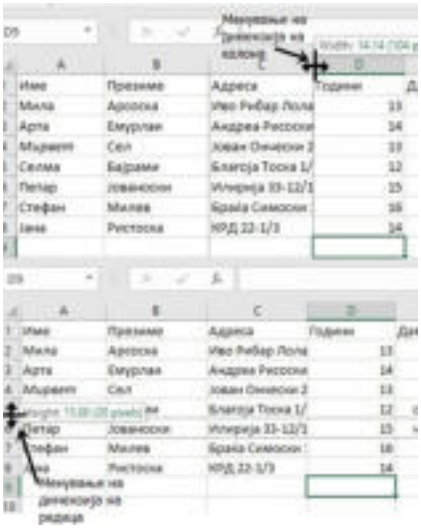
Слика 5: Вметнување колона и редица преку менито

Обидете се!

Бидејќи ја научивме постапката за вметнување колона и редица во табела, ајде да се обидеме да избришеме колона или ред, ако знаеме дека наместо опцијата Insert треба да избереме Delete.

1.3.3 Менување на димензиите на колоните и редовите

При внесување на податоци во ќелиите, потребно е честопати да ги **менуваме димензиите на колоните и редовите.** Постапката за произволно менување на димензиите на колоната, односно редицата, прикажана е на следната слика:



Слика 6: Менување димензија на колона и редица

Од сликата можеме да забележиме дека најпрво го позиционираме покажувачот помеѓу буквите на колоната, односно помеѓу броевите на редиците, добиваме курсор , притискаме и влечеме. При менување на димензијата на редицата, курсорот е променет за 90o, а постапката е иста.

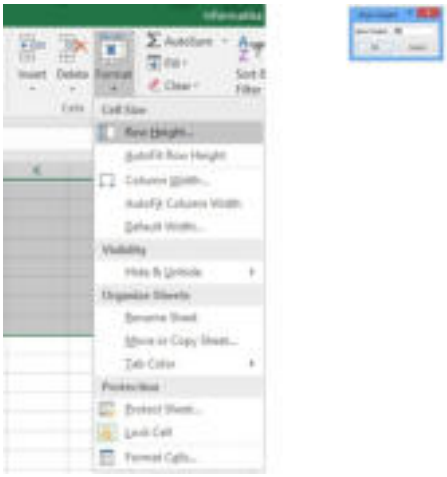
Забелешка!

Димензиите на колоните/редовите можете да ги менувате на побрз начин со двоклик на линијата по ознаката за колоната/редот.

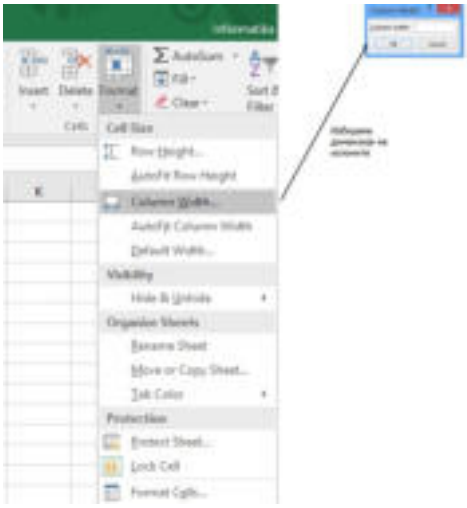
Задача

Отворете го документот „Vezba2.xlsx“ и изменете ги димензиите на колоната каде што се наоѓа датумот на раѓање и насловната редица. Потоа, извршете ја постапката за менување димензии на колона/ред преку опцијата Format од менито Home, со цел да добиете исти димензии за сите колони и редици.

Но, кога колоните и редиците корисникот сака да ги зголеми или намали за точно определена вредност, тогаш преку менито Home, ја избираме алатката Format:



Слика 7: Менување димензии на редици преку опцијата Format од менито Home



Слика 8: Менување димензија на редица преку опцијата Format од менито Home

Запомни!

Уредувањето или едитирањето на документ е постапка за корекција. За таа цел, можеме да ги менуваме податоците со кликнување врз ќелијата и веднаш да започнеме со пишување, или, пак, со двоклик и позиционирање на курсорот на местото каде сакаме да направиме корекција. Преку менито Home можеме да вметнуваме колона и редица, да бришеме колона и редица, како и да ги менуваме димензиите.

Прашања

1.Која е постапката за менување податок во ќелија?
2.Како се менуваат димензиите на колоните и редиците: Наведи ги постапките!

1.4 Форматирање табела

Да се потсетиме

Потсети се што е форматирање! Кои алатки најчесто се употребуваат за форматирање? Можете ли да објасните на што треба најмногу да се внимава при форматирање?

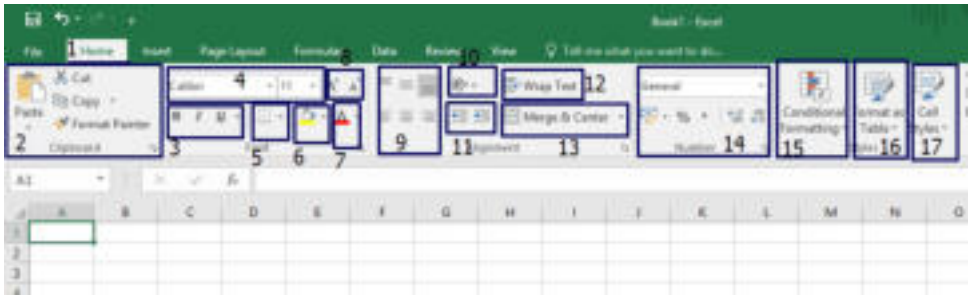
Изгледот на ќелиите, колоните и редиците се финализира со форматирање на изгледот на табелата.

Форматирањето на податоците, всушност претставува додавање на ефекти кои го разубавуваат изгледот на табелата, а со тоа овозможуваат поголема прегледност на податоците во истата.

Форматирањето на ќелиите претставува:

- промена на атрибути за фонт;
- порамнување на податоците во ќелиите;
- прикажување на текстот под агол;
- спојување и разделување ќелии;
- додавање рамки и линии на една или повеќе ќелии;
- додавање боја во ќелијата и додавање ефекти.

Но, пред да се преземе некаква активност, најпрво ќелиите треба да бидат селектирани. Најголем дел од алатките за форматирање се наоѓаат во менито **Home**:

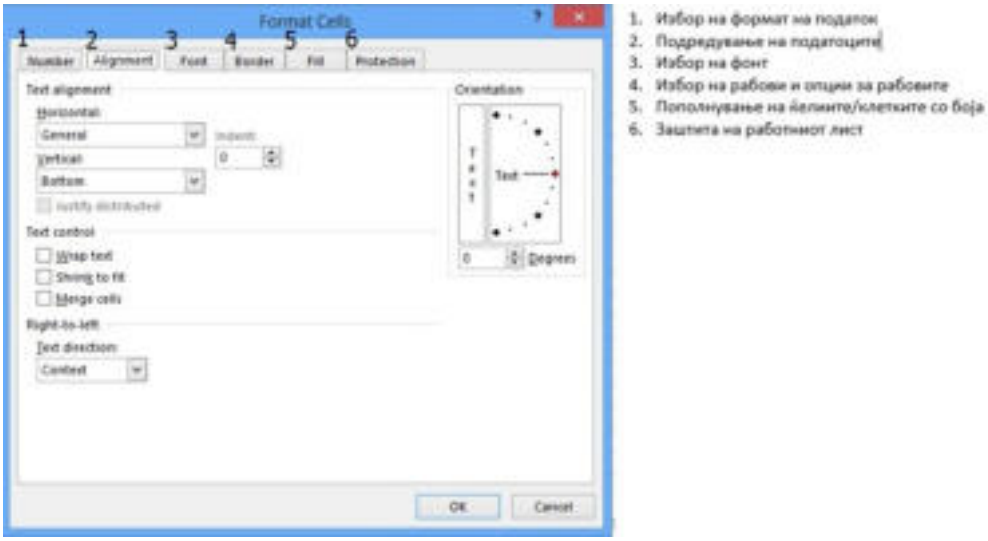


Слика 1: Алатки за форматирање од менито Home

1.Менито Home.
2.Наредбите за копирање, отсекување, вметнување и алатака за форматирање.
3.Стиловите: задебелен, накосен и подвлечен.
4.Избор на фонт и големина на фонт.
5.Избор на рабови.
6.Избор на боја на ќелиите.
7.Избор на боја на фонтот.
8.Алатки за зголемување/намалување на буквите.
9.Ориентација на податоците.
10.Вовлекување.
11.Разделување на текстот.

12.Спојување на ќелии и подредување на центар.
13.Избор на тип на податок.
14.Условно форматирање.
15.Форматирање на табела.
16.Стилови на ќелија.

Форматирањето на табелата, освен преку менито **Home** може да се постигне и со опцијата **Format Cells** по претходно селектирање на ќелиите од табелата:



Слика 2: Форматирање на податоците преку менито Format Cells

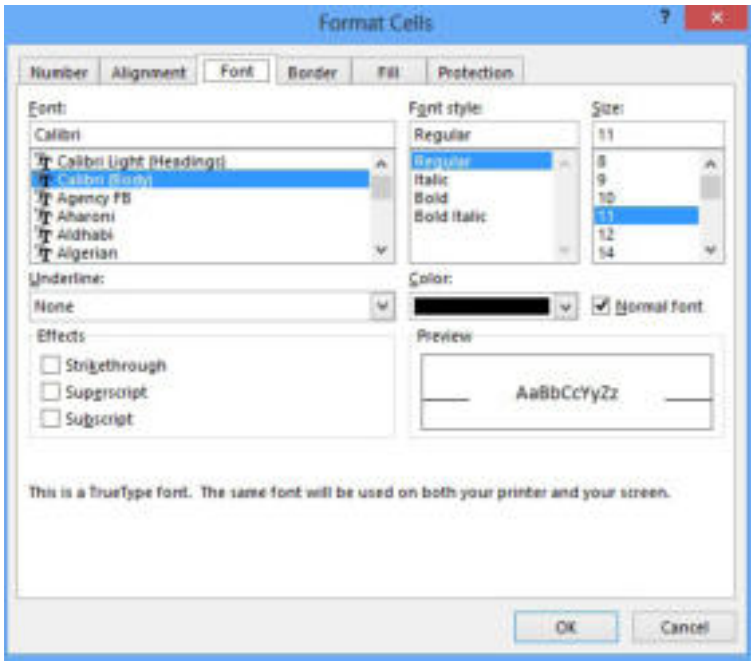
Преку табот Alignment корисникот може да прави подредувања на содржината на ќелиите, односно да ги подредува податоците во ќелиите, така што во Text Alignment се избира подредување на податоците хоризонтално и вертикално, како и ориентација или насока на текстот во ќелијата.
Дополнителни уредувања се вршат и со избор на следните опции:

Wrap Text – прикажува текст кој ќе се распореди во повеќе реда.

Shrink To Fit – го намалува текстот со цел да го собере во ќелијата без да се променат димензиите на ќелијата.

Merge Cells – овозможува спојување на селектирани ќелии.

Преку избор на табот **Font**, се избира облик на пишување на букви, стилови, големина и останатите ефекти кои се однесуваат на фонтот:

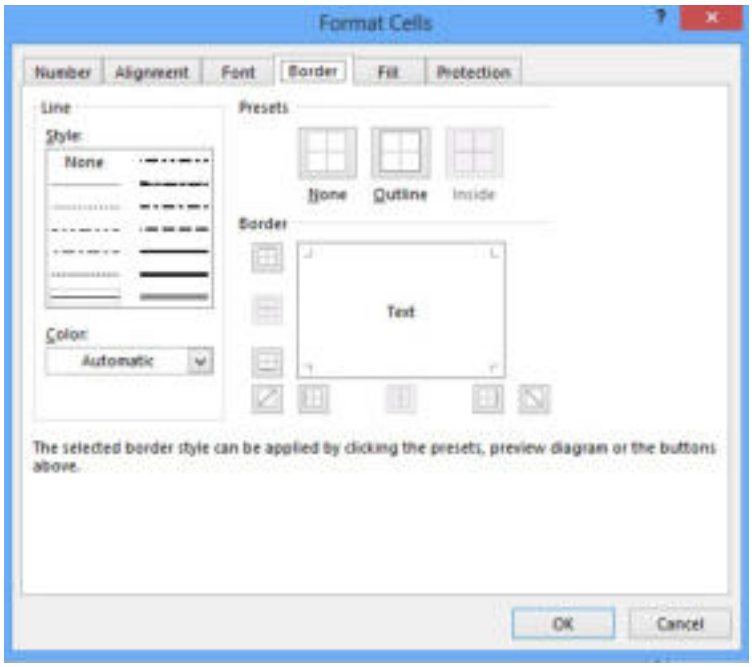


Слика 3: Избор на алатките за фонт

Забелешка!

Опциите за фонт, стил на фонтот и големина на фонтот можете да ги изберете и преку менито Home.

Во опцијата **Border,** се избира стил, дебелина и боја на рабовите на ќелиите на табелата:

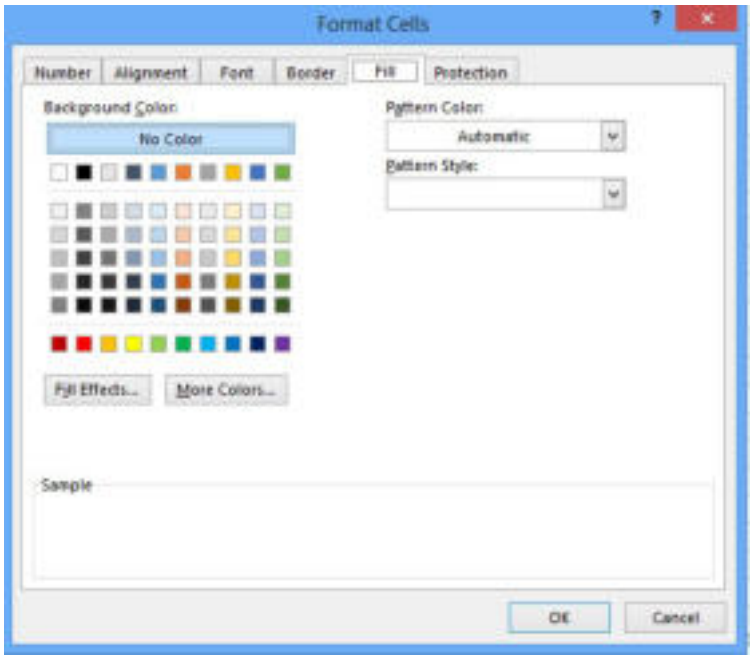


Слика 3: Избор на алатките за фонт

Забелешка!

Менито Home ги содржи алатките за уредување на рабовите на табела, како можност за побрз и полесен пристап.

Преку опцијата Fill се избира боја за пополнување на ќелиите:



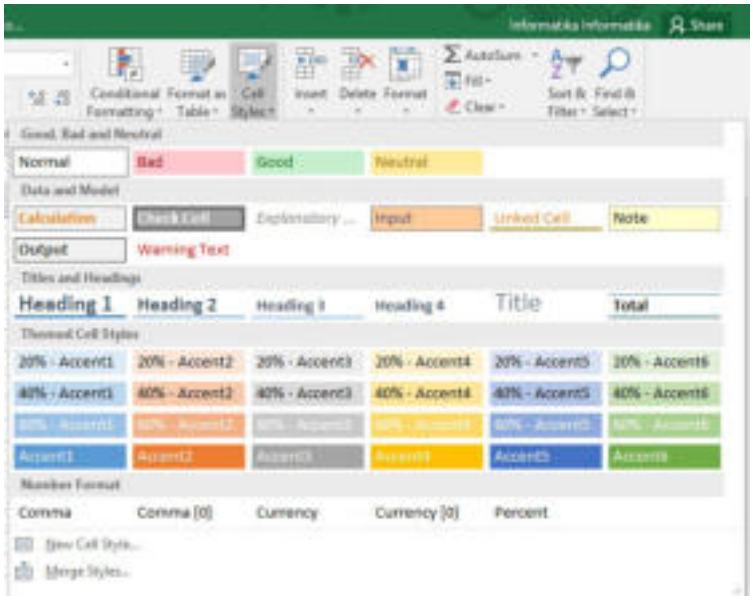
Слика 5: Избор на боја на подлогата на ќелијата

Задача

Отворете нов работен документ и во првиот работен лист (Sheet1) креирајте распоред на часови:

- во ќелиите A1, B1, C1, D1, E1 внесете ги имињата на деновите во неделата во кои посетувате настава;
- во ќелиите кои се наоѓаат под насловните ќелии внесете ги предметите согласно со распоредот на часови;
- додадете нова колона пред колоната понеделник и внесете реден број на часовите;
- додадете нов ред над насловните ќелии;
- спојте ги првите шест ќелии и напишете наслов „Распоред на часови“;
- насловните ќелии во кои се напишани имињата на деновите дајте им фонт Calibri, големина четиринаесет (14), стилови задебелени и накосени, боја на буквите темносина, а боја на позадината на ќелиите светлосина;
- на целата табела додадете рабови;
- димензиите на колоните/редиците поставете ги по потреба;
- зачувајте го документот во фолдерот на работната површина.

Форматирањето на ќелиите со готов формат може да се избере и преку избор на алатката **Cells Styles** од менито **Home**, како што е прикажано на сликата. Но, потребно е претходно ќелиите да бидат селектирани за да биде имплементиран стил кој корисникот го избрал од понудените опции:



Слика 6: Избор на стилови на ќелии

Покрај готовите формати на ќелии, корисникот може да избере готов формат на табели преку избор на алатката Format As Table, од менито Home, кој е прикажан на следната слика:

Слика 6: Избор на стилови на ќелии

Забелешка!

Покрај стандардното форматирање на табелата постои и специјално форматирање односно условно форматирање. До него пристапуваме преку менито Home и опцијата Conditional Formatting. Обидете се да го употребите ваквиот начин на форматирање во некоја од креираните табели.

Задача

Додадете нов работен лист и креирајте табела во која ќе внесете податоци за почеток и крај на часовите и времетраење на одморите помеѓу часовите. Форматирајте ја табелата со примена на стилови на ќелии (**Cell Styles**) од менито **Home** или избор на готов формат на табела преку опцијата **Format Table** повторно од менито **Home**.

Запомни!

Форматирањето табела значи додавање ефекти на табелата кои опфаќаат менување на начинот, форматот, ориентацијата и порамнувањето на податоците во ќелијата. При форматирање табела најчесто се употребуваат алатките од менито Home или, пак, преку наредбата **Format Cells**. Постојат и готови форми на ќелии и табели и истите ги избираме преку менито **Home** и наредбите **Cells Styles** и **Format As Table.**

Прашања

1.Што е форматирање?
2.Која е разликата меѓу форматирање и уредување табела?
3.Кои алатаки најчесто се употребуваат за форматирање табела?
4.Како се уредуваат порамнувањето и ориентацијата на текстот во ќелиите?
5.Кои опции можеме да ги избереме преку наредбата Format Cells? Дали можеме да пристапиме до нив на друг начин? Објасни!
6.Која е постапката за разделување текст?
7.Зошто служи наредбата Merge Cells?
8.Зошто служи опцијата Cells Styles од менито Home?
9.Дали употребувате готов формат на табели? Која е постапката за избор?

1.5 Автоматско пополнување податоци во табелата

Да се потсетиме

Се сеќавате што е компјутерска обработка на податоците? Кои се предностите од компјутерската обработка на податоците? Можете ли да го објасните терминот „автоматско“ обработување?

Автоматското пополнување на податоци во табелата претставува многу корисна алатка на **Excel**. Оваа алатка овозможува наместо рачно да ги внесуваме податоците можеме да ја користиме функцијата Autofill за пополнување на ќелиите. Автоматското полнење е една од главните функции во табеларниот процесор, поради што потребните податоци се внесуваат во табелата со многу помалку време отколку рачно.

Всушност, **Autofill** на **Microsoft Excel** ни овозможува да креираме табели поефикасно, овозможувајќи ни брзо да ги пополнуваме ќелиите со серија податоци и тоа не само со едноставни нумерички вредности, туку и со други типови податоци или формули. Освен тоа, можно е да се креираат поединечни листи што ќе му овозможат на корисникот да не губи време секојпат на нов влез со исти вредности.

Но, пред да ја запознаеме функцијата и применливоста на автоматското пополнување на ќелиите да ги осознаеме значењата на различните форми на покажувачи.

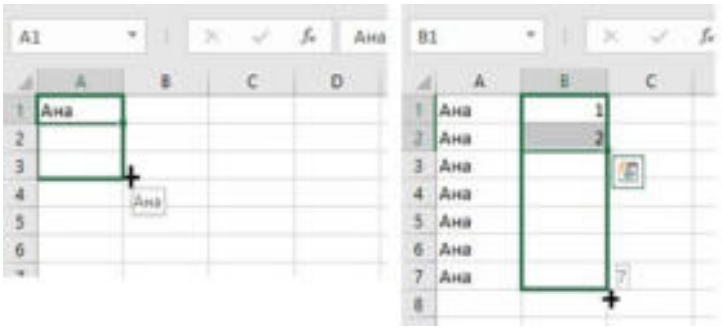
Покажувач	Значење
+	Автоматско пополнување ќелии
	Копирање содржина на ќелија
	Преместување содржина на ќелија
	Менување димнезија на колона
	Менување димензија на редица
	Означување колона
	Означување редица
	Означување ќелија

Табела 1: Форми на покажувачот

За да можеме автоматски да пополнуваме ќелии ќе го употребуваме следниот покажувач: + . Притоа, постапката за автоматско пополнување на ќелии е следната:

1.кликнуваме врз иконата која ќе ја означиме како активна ќелија и внесуваме податок;

2.по внесувањето на податокот кликнуваме Enter од тастатурата;
3.со глувчето доаѓаме до десниот долен агол на ќелијата во која се наоѓа податокот;
4.добиваме покажувач за автоматско пополнување ;
5.кликуваме и влечеме до ќелијата до која сакаме да го извршиме автоматското пополнување.



Слика 1: Автоматско пополнување на ќелии

Забелешка!

Во програмата за табеларни пресметки можат да се крираат листи, како на пример: деновите во неделата, месеците во годината и сл. Постапката за креирање на листите е преку менито File, Options, менито Advance и избираме Edit Custom List. Обидете се да креирате една готова листа!

Задача

1.отворете нов работен документ во Excel 2016;
2.во работниот лист 1 (Sheet 1), во колона А, да се креира низа од парни броеви упоребувајќи го автоматското пополнување на ќелии, почнувајќи од бројот два (2) до бројот дваесет и два (22);
3.во колона В, на истиот работен лист, да се креира низа од непарни броеви почнувајќи од бројот три (3), па сè до дваесет и три (23);
4.во колона С да се креира низа од името Ана, во колона D низа од името Ана Марија;
5.во колона Е да се креира низа која ќе се менува за вредност пет (5) почнувајќи од нула (0);
6.во колона F пополнете ја низата 100, 99, 98.....89.

Запомни!

Автоматското пополнување ќелии е една од многуте предности на програмата за табеларни пресметки, бидејќи ни овозможува на полесен и побрз начин внесување податоци, креирање низи, пополнување ќелии. За да извршиме автоматско пополнување на ќелии го употребуваме следниот покажувач: + . Автоматско пополнување може да се изврши нена само на податоци кои содржат текст, туку и броеви, различни формати на бројни податоци, формули и функции. Во програмата за табеларни пресметки може да се креираат и посебни листи кои ќе се употребуваат при автоматско пополнување на ќелиите.

1.6 Формули и функции во програмата за табеларни

Како што наведовме во самата дефиниција, програмата Excel, освен што ни овозможува табеларен и прегледен приказ на податоците, ни овозможува да вршиме и пресметки. Според тоа, главната цел за примената на формулите и функциите во програмата за табеларни пресметки е вршење на брзи и прецизни пресметки.

1.6.1 Формули

Формулите во програмите за табеларни пресметки се креираат од податоци и различни математички оператори меѓу нив. Excel 2016 ги користи стандардните знаци за формули прикажани во следната табела:

Знак	Математичка операција
+	Собирање
-	Одземање
*	Множење
/	Делење
^	Степен/експонент

Табела 1: Математички оператори за вршење пресметки

Забелешка!

Може да се користат и логички оператори како: <, >, =, <=, >=, <>, кои враќаат вредност вистина (True) или невистина (False). При креирање на формулите може да се користат и загради.

Формулите и функциите се внесуваат во ќелијата од табелата и секогаш започнуваат со знакот „=“.

На пример:

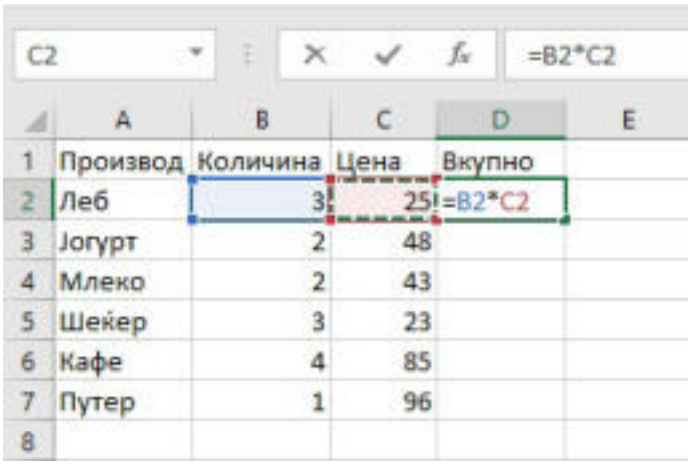
За да пресметаме колку треба да платиме на касата во супермаркетот со примена на формули, по внесувањето на податоците ги извршуваме пресметките:

1.во ќелијата D2 кликнуваме и го впишуваме знакот „=“;

2.ја впишуваме адресата на ќелијата во која се наоѓа количината;

3.го додаваме знакот за множење и адресата на ќелијата во која е впишана цената на прозиводот;

4.кликнуваме Enter од тастатура



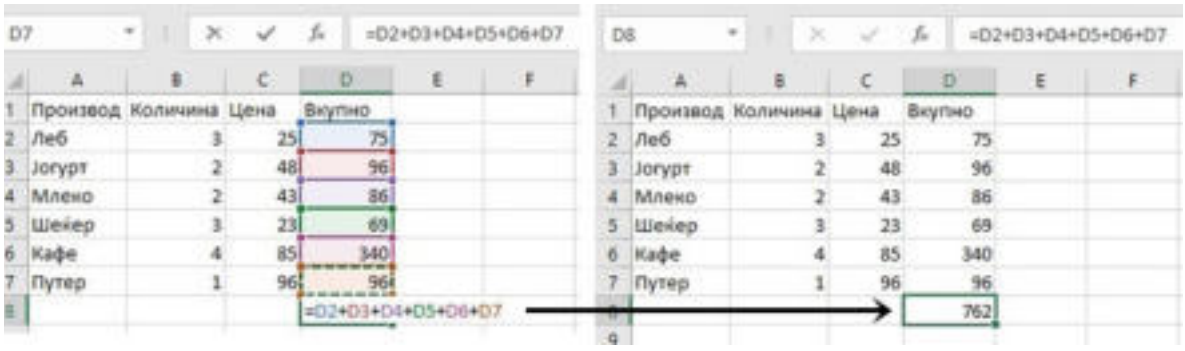
Слика 1: Пресметка со примена на формула

Забелешка!

Освен со впишување на адресите на ќелиите, адресата на ќелијата во формулата можеме да ја додадеме со кликнување врз неа.

Од задачата, забележуваме дека при внесување на формула во активната ќелија, со кликнување на копчето **Enter** од тастатура во активната ќелија се прикажува резултатот, а во **лентата за формули (Formula Bar)** се прикажува формулата која сме ја употребиле за да дојдеме до резултатот. Пресметката за сите останати артикли можеме да ја извршиме со автоматско пополнување на ќелиите.
Ајде да ја пресметаме и вкупната сума која треба да се плати!

1.Кликнуваме во D8 и го впишуваме знакот „=“.
2.Ги впишуваме адресите на ќелиите од D2:D7 со знакот за собирање помеѓу нив.
3.Кликнуваме Enter од тастатура и добиваме вкупен резултат.



Слика 2: Пресметка на вкупна сума со примена на формула

Забелешка!

При креирање формули нема да работиме со вредностите во ќелиите туку со нивните адреси, со цел при промена на вредноста на податокот автоматски да се промени и резултатот.

Задача

Да се пополни следната табела:

D2

	A	B	C	D
1	X	Y	z=x*2-2*y	s=(x+y)/4
2	0	3		
3	1	6		
4	2	9		
5	3	12		
6	4	15		
7	5	18		
8	6	21		
9	7	24		
10	8	27		
11	9	30		
12	10	33		

1.6.2 Функции

Функции се готови формули кои може да се користат за изведување одредени пресметки со податоците во табелата. Функциите имаат свое име и се групирани во категории: математички, статистички, логички и сл. Функциите работат со адреси на ќелии, ранг на ќелии или несоседни ќелии.

Пред функцијата се пишува знакот за еднакво (=), потоа следи името на функцијата и на крајот аргументите на функцијата. Аргументи на функцијата всушност се ознаките на ќелиите кои ќе учествуваат во пресметката.

Слика 3: Елементи на функцијата

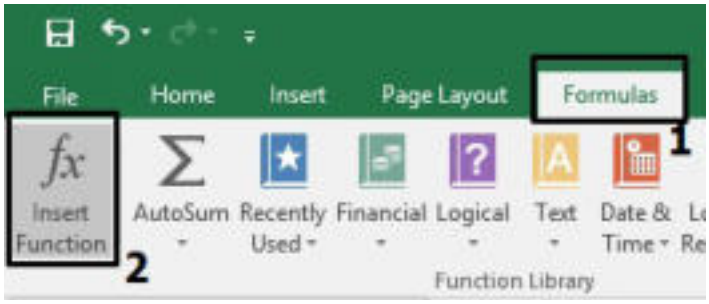
Најчесто користени функции се:

SUM - пресметува збир на бројни вредности во група ќелии.
MIN - дава најмала бројна вредност од група на ќелии.
MAX - дава најголема бројна вредност од група на ќелии.
AVERAGE - пресметува средна вредност од бројните вредности во група ќелии.

COUNT – го дава бројот на нумерички внесови во селектираните ќелии.

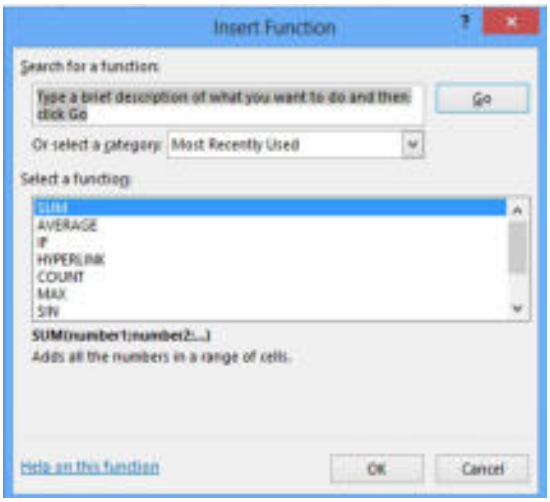
COUNTIF- го дава бројот на внесови во селектираните ќелии доколку е задоволен определен услов.

Постапката за внесување на функција започнува со селектирање на ќелијата во која ќе се изврши пресметката, а потоа преку менито **Formulas** и наредбата **Insert Function** пристапуваме кон внесувањето на функцијата.



Слика 4: Постапка за внесување функции

Притоа, се појавува следната слика од која избираме категорија и ја избираме функцијата:



Слика 5: Избор на функција

За да се изврши пресметката со помош на избраната функција додаваме и аргументи од следниот прозорец:

Слика 6: Избор на аргументи на функција

Со кликнување на копчето **OK** од прозорецот ќе го добиеме резултатот од пресметката.
Но, понекогаш во ќелијата во која треба да се прикаже резултатот од формулата или функцијата се прикажуваат симболите за грешка :#####, #DIV/0!, #NAME?, #REF!, #VALUE!. Тие уште се викаат и пораки за грешка во формулата или функцијата (**Error message**).

Пораки за грешки	Значење
#####	Колоната е тесна да го прикаже резултатот
#DIV/0!	Делење со нула не е можно
#NAME?	во формулата има погрешно запишана адреса на ќелија
#REF!	Во формулата има адреса на ќелија која е избришана
#VALUE!	во формулта една од ќелиите содржи текст

Табела 2: Пораки за грешки во формула или функција

Задача

Да се креира табела во програмата за табеларни пресметки како на следната слика:

	A	B	C	D	E	F	G
1		Број на прочитани книги по месеци					
2	Име и презиме на ученик	Јануари	Февруари	Март	Април	Мај	Вкупно прочитани книги
3	Ана Михајлоска	4	3	2	1	0	
4	Сара Етеми	5	2	2	1	1	
5	Арта Тахири	4	3	1	2	1	
6	Јован Спироски	3	1	2	1	1	
7	Дритон Емини	6	1	3	2	0	
8							
9	Вкупно прочитани книги по месеци						
10	Најмногу прочитани книги						
11	Најмалку прочитани книги						
12	Просечна вредност на прочитани книги						
13	Вкупно податоци						
14	Колку ученици НЕ прочитале ниту една книга?						

Да се извршат бараните пресметки употребувајќи ги функциите: **SUM**, **MIN**, **MAX, AVERAGE, COUNT** и **COUNTIF**.

Запомни!

Формулите во програмите за табеларни пресметки се математички изрази креирани од податоци и различни математички оператори помеѓу нив. Во формулите не се работи со податоците во ќелиите, туку со нивните адреси. Функциите се готови формули кои може да се користат за изведување на одредени пресметки со податоците во табелата. Функциите имаат свое име и се групирани по категории: математички, логички, статистички и сл.

Прашања

1.Што се формули, а што се функции?
2.Опишете ја разликата меѓу формулите и функциите!
3.Кои се најчесто употребувани функции?
4.Истражете, набројте и наведете други функции на Excel, освен изучуваните во наставната единица!
5.Напиши ја синтаксата на функцијата која пресметува збир на неколку вредности од последователни ќелии!

1.7 Сортирање податоци

Ако работниот лист содржи многу содржини, може да биде тешко да се најдат брзо информации. Филтрите може да се користат за да се стеснат податоците во работниот лист, овозможувајќи да се видат само информациите што му се потребни на корисникот.
Податоците во табелата може да се организираат и притоа да се креираат извештаи според потребите на корисникот. Тоа се прави со следните операции:

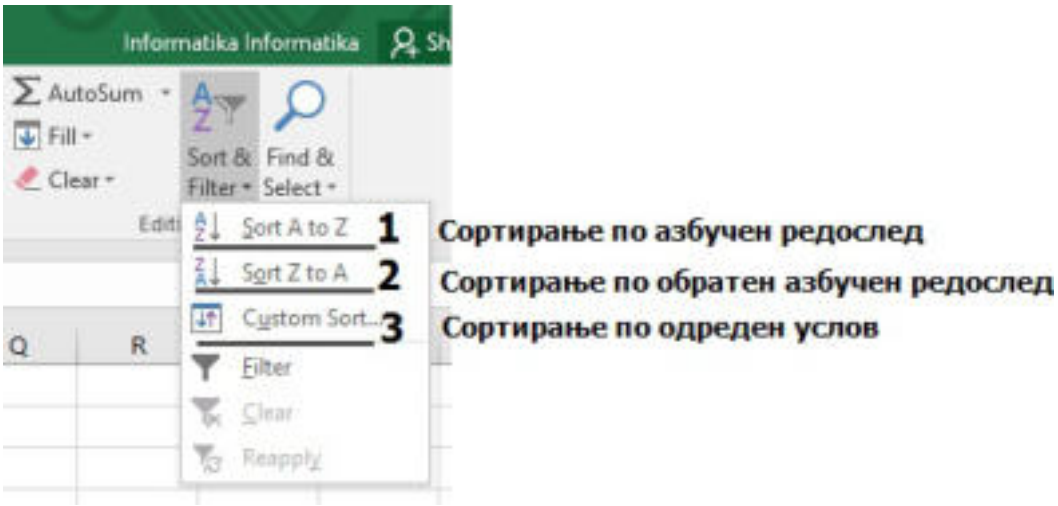
- сортирање на податоците по зададен редослед;
- филтрирање на податоците според определени критериуми;
- креирање збирни и подзбирни податоци.

Ние во оваа наставна единица ќе се задржиме на постапката за сортирање, како посебна алатка која го олеснува пристапувањето до потребните информации како издвоени од останатите.

Сортирање, значи промена на редоследот на редовите во табелата, за да бидат наредени податоците во нив според одреден редослед. Во Excel можни се следните начини на сортирање:

- сортирање по опаѓачки или растечки редослед;
- сортирање по азбучен, нумерички, хронолошки или редослед дефиниран од страна на корисникот;
- сортирање според дефинирано условно форматирање.

Ако сортирањето треба да се направи според еден критериум, односно по една колона, тогаш доволно е само да се кликне врз едно поле од колоната по кој корисникот сака да сортира и потоа од менито Home се

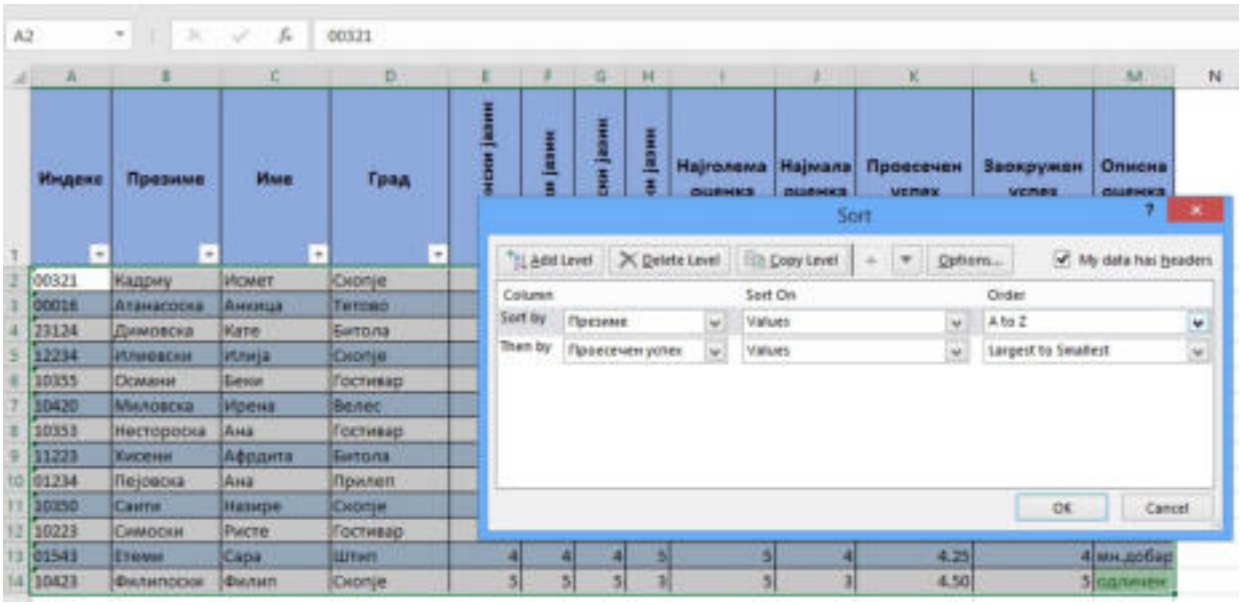


Слика 1: Сортирање

Доколку корисникот ја избере опцијата **Custom Sort**, тогаш во полето **Column (Sort by)** се избира критериумот по кој ќе се врши сортирањето, во полето Sort on се избира каков тип податоци ќе се сортираат, и во полето **з** се избира редоследот на сортирање. На пример, во следната табела направено е сортирање по следните критериуми:

1.по презиме на ученикот по азбучен редослед;

2.просечен успех како втор критериум по редослед од најголемиот просек кон најмалиот.



Слика 2: Сортирање податоци со Custom Sort

Задача

Во програмата за табеларни пресметки да се креира следната табела „Податоци од библиотека“:

	A	B	C	D	E	F
1	Податоци од библиотека					
2	Ред бр	Лектира	Име на автор	Презиме на автор	Оддление	Изнајмена
3	1	Мајка	Јован	Јовановиќ - Змај	II	84
4	2	Зоки Поки	Оливера	Николовска	II	68
5	3	Поезија	Рифат	Кукај	III	35
6	4	Шеќерна приказна	Славко	Јанески	III	44
7	5	Сказна за детето Вилен	Глигор	Прличев	IV	54
8	6	Училиште	Драган	Луќиќ	III	30
9	7	Орхан	Нејати	Зекирија	IV	57
10	8	Летни цвеќиња	Наим	Фрашери	VI	67
11	9	Силјан Штркот	Марко	Цепенков	VII	46
12	10	Принцезата Арѓиро	Исмаил	Кадаре	VII	44
13	11	Бегалка	Видое	Подгорец	VIII	70

Да се изврши сортирање по категорија **„Лектира“** по азбучен редослед, а потоа по колку пати е **„Изнајмена“** започнувајќи од најголемата вредност кон најмалата!

	A	B	C	D	E	F
1	Податоци од библиотека					
2	Ред бр	Лектира	Име на автор	Презиме на автор	Оддление	Изнајмена
3	1	Мајка	Јован	Јовановиќ - Змај	II	84
4	2	Зоки Поки	Оливера	Николовска	II	68
5	3	Поезија	Рифат	Кукај	III	35
6	4	Шеќерна приказна	Славко	Јанески	III	44
7	5	Сказна за детето Вилен	Глигор	Прличев	IV	54
8	6	Училиште	Драган	Лукиќ	III	30
9	7	Орхан	Неџати	Зекирија	IV	57
10	8	Летни цвеќиња	Наим	Фрашери	VI	67
11	9	Силјан Штркот	Марко	Цепенков	VII	46
12	10	Принцезата Арѓиро	Исмаил	Кадаре	VII	44
13	11	Бегалка	Видое	Подгорец	VIII	70

Запомни!

Сортирање е процес на подредување на податоците според правила. Подредувањето може да биде по азбучен редослед, по обратен азбучен редослед или според дефинирани услови кои се задаваат преку опцијата **Custom Sort.**

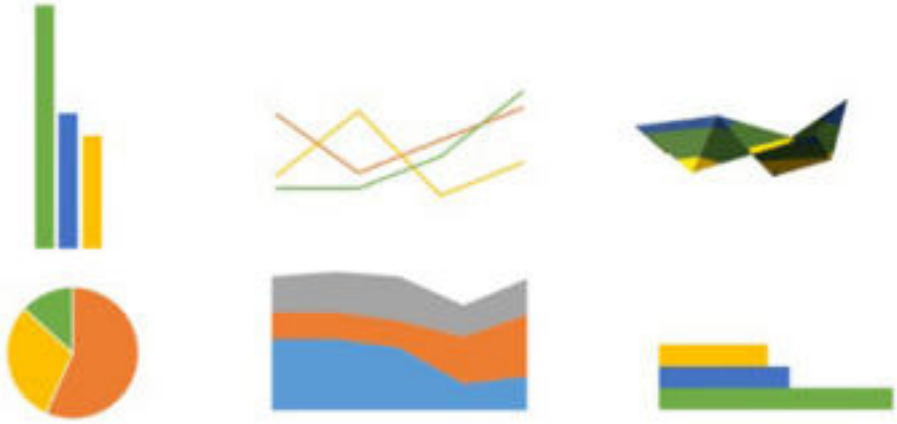
Прашања

1.Што е сортирање?
2.Која е постапката за сортирање на податоците по азбучен редослед?
3.Како можеме да извршиме сортирање на податоците по даден услов?

1.8 Креирање графикон

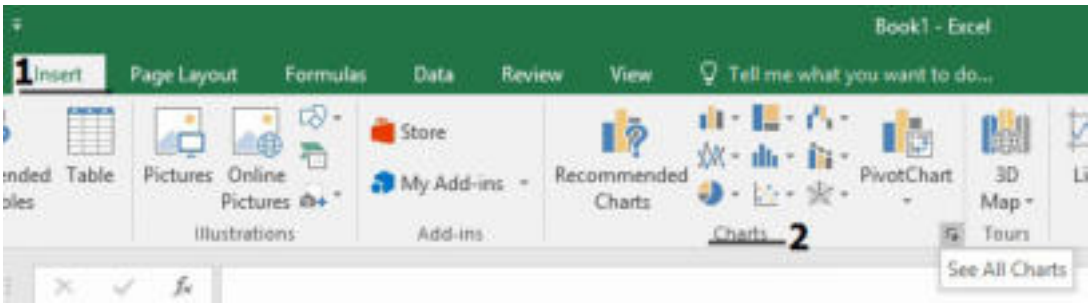
Откако веќе е креирана табела, направени се форматирања според постапките кои ги наведовме во претходните наставни единици, честопати потребно е податоците да се претстават на сликовит начин, односно со употреба на графикон. Со сликовитото прикажување на податоците му се овозможува на корисникот брз и јасен увид во вредностите на податоците, нивно споредување и анализа.

Според тоа, г**рафиконите се сликовито прикажување на податоците од табелата.** Всушност, графиконот ги претвора податоците во слика. Графиконите може да бидат: столбести, во вид на линии, пита, ѓеврек и сл., тие истите ни користат за различни цели, како на пример: за споредба по категории, за споредба на процентуални вредности и сл., како што е прикажано на следната слика:



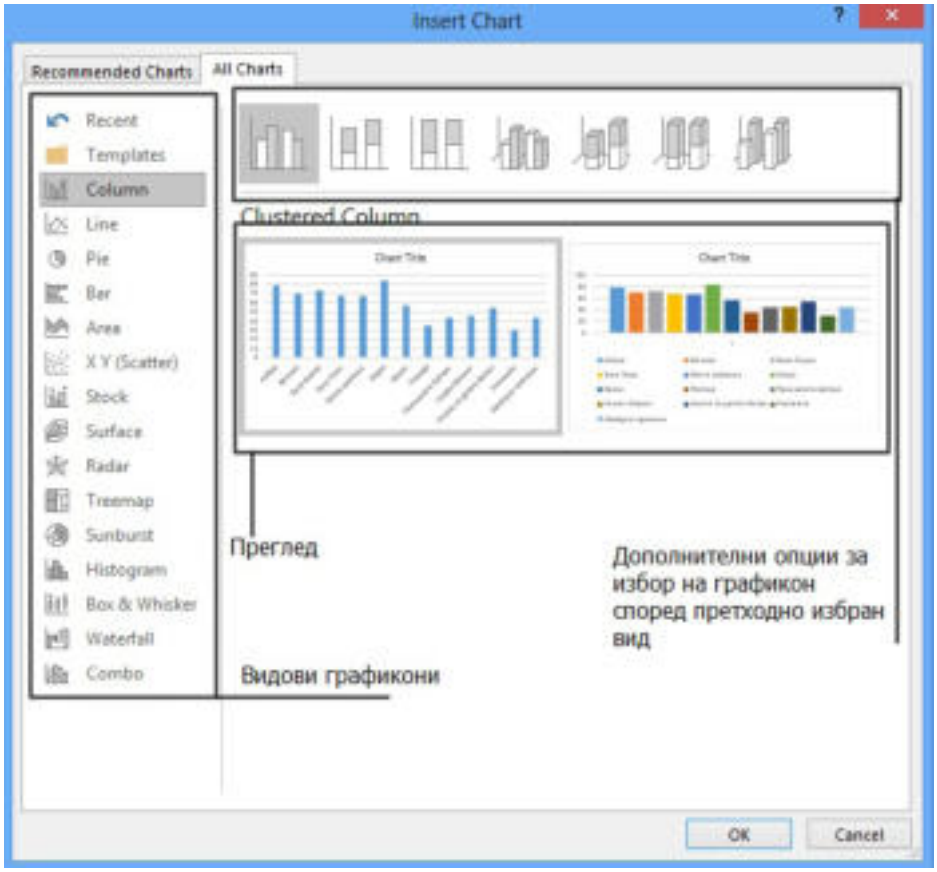
Слика 1: Видови графикони

Постапката за креирање графикон во програмата за табеларни пресметки е едноставна доколку податоците се правилно напишани и организирани во работниот лист. Потоа, се селектира рангот на ќелии со податоци кои треба да бидат прикажани со графикон. По изборот на податоци кои графички ќе бидат прикажани можеме да ја примениме наредбата за креирање графикони во **Excel 2016** преку менито **Insert** и делот **Charts**:



Слика 2: Постапка за вметнување графикон

Од сликата, забележуваме дека при вметнување на графикон веднаш можеме да ја избереме категоријата, односно видот на графиконот или, пак, преку **Quick Launcher**, кој се наоѓа на левата страна на рубриката **Chart**, при што се прикажува следниот прозорец:



Слика 3: Постапка за избор видови графикони

Со кликнување на копчето **OK**, графиконот се појавува во работниот лист во кој се наоѓа и изворот на податоци, односно изворната табела. Тогаш одлучуваме дали графиконот е прегледен да остане во истиот работен лист или, пак, да го пренесме во нов работен лист. Исто така, додека е селектиран графиконот се појавуваат менијата **Design** и **Format** кои овозможуваат форматирање и уредување на изгледот на графиконот. На пример, преку менито **Design** можеме да ги примениме следните можности:

- додавање елементи на графиконот;
- распоред на елементите на графиконот;
- избор на боја;
- стил на графиконот;
- избор на податоци;
- промена на видот на графиконот;
- префрлање на графиконот во нов работен лист/во истиот работен лист со табелата.

Преку менито Format можеме да додадеме облици на деловите од

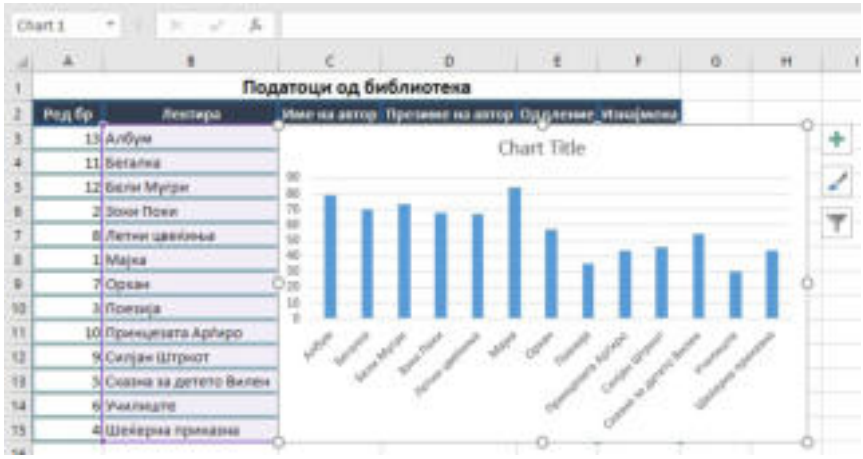
графиконот, боја на линијата, обојување на елементите, стилови на буквите, димензии на елементите и сл.

Забелешка!

При креирање формули нема да работиме со вредностите во ќелиите туку со нивните адреси, со цел при промена на вредноста на податокот автоматски да се промени и резултатот.

Задача

Ајде да ја отвориме табелата „Податоци од библиотека“ која е зачувана во фолдерот на работната површина. Да креираме графикон во кој сликовито ќе биде прикажано која книга најмногу е прочитана, употребувајќи графикон **Columns**, како што е прикажано на следната слика:



Графиконот да биде на нов работен лист и да се форматира применувајќи ги алатките од менито **Design** и **Format**.

Запомни!

Графиконите се сликовито прикажување на податоците од табелата. Тие овозможуваат брз и јасен увид во вредностите на податоците, нивно споредување и анализа. Графиконите можат да бидат: столбести, во вид на линии, пита, ѓеврек и сл.

Прашања

1.Што е графикон?
2.Опиши ја постапката за креирање графикон?
3.Кои видови графикони овозможува програмата Excel 2016?
4.Кои алатки може да се употребуваат од менито Design и Format за уредување и форматирање на графиконот?

ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

Задача 1

Отворете еден работен документ во MS Excel 2016. Првиот работен лист именувајте го „куќа“ и во него со примена на различните начини на селектирање обојте ги ќелиите така што би добиле куќа. Документот зачувајте го со вашето презиме и одделение!

Задача 2

Во програмата за табеларни пресметки во работен лист 1 (Sheet 1) креирајте телефонски именик на учениците од одделението. Табелата да ги содржи следните податоци: реден број, име, презиме и телефон. Покрај телефонот на другарчето, додадете го и датумот на неговиот роденден.

Задача 3

Направете список на книги од вашата библиотека. Притоа, внесете ги следните податоци: Автор, наслов на книгата и година на издавање.

Задача 4

Креирајте табела во која ќе внесете податоци: месец, приходи, расходи. Откако ќе ги внесете податоците додадете нова колона: разлика (разлика меѓу приходи и расходи). Над насловните ќелии додадете нов ред. Ќелиите на новиот ред обојте ги со Fill color. Дајте им димензии на редиците дваесет (20), додека на колоните димензии дваесет и пет (25). Зачувајте го документот!

Задача 5

Во работен документ на програмата за табеларни пресметки креирајте ја следната табела:

E5 | 0

	A	B	C	D	E
1	Успех по предмети				
2	Предмет	Оценети	Неоценети	Позитивни оценки	Негативни оценки
3	Математика	12	1	10	2
4	Информатика	11	2	11	0
5	Англиски јазик	10	3	10	0
6	Мајчин јазик				

Форматирајте ја табелата употребувајќи ги алатките за форматирање: фонт, големина, стилови, подредување, ориентација, како и примена на стилови на ќелии или готови формати на табели. Документот зачувајте го!

Задача 6

Во програмата за табеларни пресметки да се креира следната табела и со примена на формули за пресметка на периметар и плоштина да се изврши пресметка:

	A	B	C	D	E
1		Геометриски фигури			
2		Страна 1	Страна 2	Периметар	Плоштина
3	Правоаголник	6	4		
4	Коцка	3	3		
5	Квадрат	5	5		
6					

Задача 7

Да се креира табела во програмата за табеларни пресметки во која се прикажани просечните температури во државата, по месеци. Со помош на функции определете кој месец е најтопол, а кој најстуден во годината!

	A	B
1	Јануари	5.6
2	Февруари	6.2
3	Март	8.8
4	Април	12
5	Мај	17
6	Јуни	21
7	Јули	23
8	Август	23
9	Септрмври	18
10	Октомври	15
11	Ноември	10
12	Декември	6.9
13		

Задача 8

Креирајте табела со податоци на вашите другарчиња: име, презиме, возраст и сортирајте ги по обратен азбучен редослед.

Задача 9

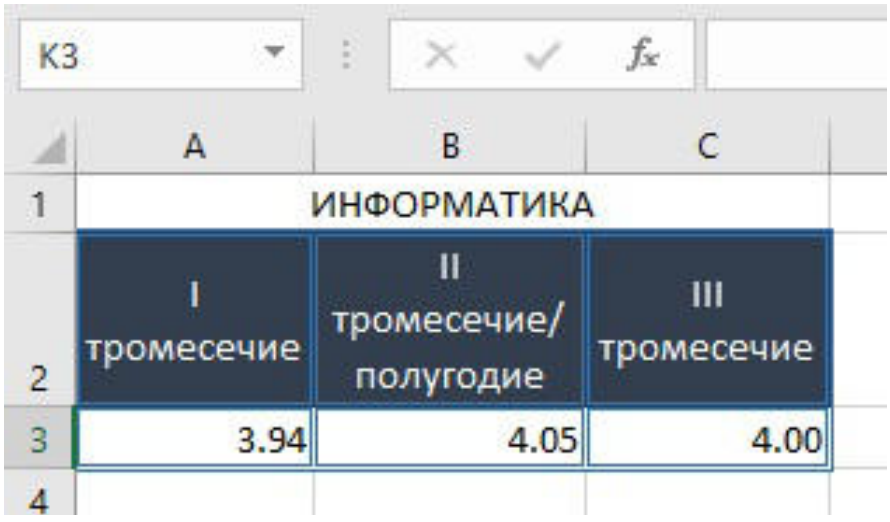
Во програмата за табеларни пресметки креирајте и форматирајте табела со податоци на вашите другарчиња: име, презиме, успех. Сортирајте ги по презиме, по азбучен редослед, а потоа, по успех почнувајќи од најголемиот кон најмалиот.

Задача 10

Да се креира графикон во програмата за табеларни пресметки во која ќе се спореди успехот на одделението од VI – IX –то одделение. Притоа, да се примени графикон пита, креиран во нов работен лист.

Задача 11

Креирајте табела во која ќе биде прикажан среден успех на крајот од секое тромесечие, како што е прикажан на сликата. Потоа, податоците графички прикажете ги со примена на графикон Columns.



Задача 12

Истражете која е цената на водата по метар квадратен во вашиот град и пресметајте колкава сметка треба да платите!

Проектна задача

Во програмата за табеларни пресметки, во работниот лист 1 да се креира следната табела:

Реден број	Презиме	Име	Град	Македонски јазик	Англиски јазик	Германски јазик	Албански јазик
1	Кадриу	Исмет	Тетово	4	5	5	3
2	Атанасоска	Анкица	Скопје	5	5	5	5
3	Димовска	Кате	Прилеп	4	4	2	3
4	Илиевски	Илија	Битола	3	3	2	2
5	Османи	Бакија	Скопје	3	3	5	4
6	Миловска	Ирена	Штип	4	4	4	5
7	Нестороска	Ана	Гостивар	5	4	4	2
8	Хисени	Афродита	Велес	2	2	3	3
9	Пејовска	Ана	Битола	2	3	4	5
10	Санти	Назире	Скопје	5	3	4	2
11	Симоски	Ристе	Гостивар	5	2	3	4
12	Етеми	Сара	Скопје	5	5	5	3
13	Филипоски	Филип	Гостивар	3	3	5	5
14	Спироски	Ристе	Валандово	3	4	4	4

- Насловите на ќелиите со јазиците да имаат вертикална насока на текстот и да се примени опција Wrap Text.
- Веднаш до колоната „Албански јазик“ да се додаде колона „Најголема оценка“ и со помош на функција да се пресмета најголемата оценка.
- До колоната „Најголема оценка“ да се додаде колона „Најмала оцека“ и со помош на функција да се пресмета најмалата оценка.
- На крајот од табелата да се пресмета просечен успех на секој поединечен ученик/студент.
- Ќелиите од A15 до J15 да се спојат и да се напише просек на класот.
- Учениците со просечна оценка „добар“ да се обојат со црвена боја, а „одличните“ ученици со зелена боја, користејќи условно форматирање.
- Во ќелијата K15 да се пресмета просекот на одделението.
- Под колоните на предметите да се избројат колку петки, четворки, тројки и колку двојки има по предметите соодветно.
- Табелата да се форматира употребувајќи: фонт, боја на букви, стилови на буквите, боја на позадина на ќелиите, вид рамки и боја на рамки со услов да се прикаже прегледност. Може да се употребуваат и готовите формати/дизајн на табела.
- Сортирајте ги податоци по презиме почнувајќи од А, а вториот услов по успех почнувајќи од најдобриот.
- Креирајте графикон на посебен работен лист во кој ќе се прикаже успехот на учениците. Графиконот форматирајте го по желба.
- Зачувајте го документот!

Запознавање со информатички концепти преку решавање логички натпреварувачки задачи

Запознавање со информатички концепти преку решавањена логички натпреварувачки задачи
Анализа и решавање на логички натпреварувачки задачи
Поврзаност на задачата со концепти од компјутерската наука – информатички концепти
- Структури на податоци
- Бинарни броеви
- Криптографија

ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

2 Запознавање со информатички концепти преку решавање логички натпреварувачки задачи

Што ќе научиме?

Да креираме табели со различен тип на податоци и низи од пресметки со примена на формули и функции, уредени со алатките за форматирање и графички претставени со графикони.

При изучување на компјутерските науки, покрај другите, една од највозбудливите и најинтересните работи е осознавањето на нов начин на размислување и решавање на проблемски задачи и ситуации, кој се вика **компјутерско размислување**. Овој начин на размислување е клучен за 21 век, како основа за постигнување на успех во многу ситуации во секојдневното живеење, во текот на образованието, како и во професионалното издигнување и усовршување.

Пред да се употребат компјутерите за решавање на проблем или проблемска ситуација, потребно е да се разбере суштината на проблемот. Затоа, може да се каже дека компјутерското размислување претставува збир на повеќе вештини, како што се: креативност, можност за објаснување и докажување, тимска работа и соработка.

Главни елементи на компјутерското размислување се:

- логичко размислување;
- алгоритамско размилсување;
- ефикасно решавање;
- научно размислување;
- иновативно размислување.

Во оваа тема ќе ставиме посебен акцент на **логичкото размислување**, како посебна и основна вештина на компјутерското размислување. Логичкото размислување најчесто се поврзува со компјутерските концепти, но во поново време претставува синоним за интегрирање на компјутерската идеја и во други дисциплини.

Кога ќе кажеме дека нешто е логичко, всушност сметаме дека тоа нешто има смисла. Способноста на поединецот да размислува на начин базиран на факти и докази е познат како логичка способност за размислување. Според тоа, **логичкото размислување може да се дефинира како процес во кој се употребува објективно расудување со цел да се дојде до одреден заклучок.**

Обидете се!

Истражете го поимот логика! Дефинирајте го и обидете се да истражите за разликите меѓу логичкото размислување и критичкото рамислување? Што е слично, а што е различно меѓу нив?

Проблемските задачи или ситуации кои се решаваат со помош на логичкото размислување имаат структура, врски помеѓу фактите и имаат смисла, односно логичка причинско-последична врска. Ваквиот начин на размислување вклучува длабока **анализа**, како на пример: мерење на сите расположливи опции, користење на факти и бројки, како и донесување на важни одлуки кои се засноваат врз добрите и лошите страни, а тоа значи дека при ваков процес предвид не се земаат елементите на чувства и емоции. Најчестиот пример за логичко размислување е **СУДОКУ**. Дадени се информации за вредностите кои се содржани во неколку ќелии. За да се реши сложувалката треба да се заклучат вредностите во сите други ќелии, почитувајќи го правилото при решавање на оваа проблемска задача. Ако се направи најмала грешка, тогаш ќе бидеме далеку од решението.

Иако овој процес е незабележителен, сите ние секојдневно се соочуваме со разни ситуации и состојби кои ги надминуваме благодарение на нашите вештини за расудување. **На пример**: додека ги пресметаме цените во супермаркетот за да провериме дали можеме да добиеме сè што ни треба за пониска цена или, пак, додека се обидуваме да ги вклопиме сите наши обврски во еден ден, нашата машина за размислување постојано врти за да најде соодветно решение. Согласно со тоа, при логичко размислување, најважни работи кои мора да ги почитуваме се следните:

- **Не ги гледај работите единствено од своја перспектива.** Тоа е субјективно и не нè води кон целта, односно кон конечното решение. На пример: Стефан и Алмир ручаат во ресторан. Чинијата на Стефан има непријатен и лош мирис, додека, пак, Алмир ужива во ручекот. Бидејќи на Стефан не му се допаднал мирисот на храната, тој брзо заклучил дека оброкот не е хранлив, не е здрав и не е убаво подготвен. Ова не е логичен начин за доаѓање до заклучок, бидејќи Стефан нема докази дека храната не е здрава и е лошо подготвена. За да се дојде до логичен заклучок, мора да се исклучат сопствените субјективни мислења и да се стави фокус на докажани информации, како на пример, кои состојки се употребени за да се подготви оброкот, да се осознае постапката за приготвување и да не се базира на претпоставки.

- **Размислувај пред да стартуваш – Креирај стратегија.** Стратегијата игра голема улога во процесот на размислување. Започнете со истражување поставувајќи прашања кои ќе помогнат во толкувањето на фактите. Активно барајте детали и осознавајте како функционираат тие како издвоени делови, а како во група, пред да дојдете до големата слика.

Внимавајте на значењето на зборовите. Малите јазични варијации прават голема разлика во значењето на поставената задача или проблемска ситуација. Утврдувајќи ја разликата помеѓу изјавите, дефинитивно ќе се исфрлат недоследностите и двосмисленостите во логичкото размилување. На пример: зборот **„неопходно“** е различно од **„доволно“.** Неопходното, значи дека мора да се исполни условот или потапката, за разлика од доволно, коешто означува да се направи минимален напор за доаѓање до решение.

На крајот, да заклучиме, дека логичкото размислување како збир на вештини и техники за решавање на проблемски задачи и ситуации, претставува чекор пред програмирањето како процес за креирање софтвер/програми за различни цели.

Клучни поими!

компјутерско размислување, логичко размислување, анализа, програмирање, структура на податоци, бинарни броеви, кодирање, криптографија, слободен софтвер.

Запомни!

Компјутерското размислување е нов начин на решавање на проблемски задачи и ситуации, кои со помош на логичко размислување се доаѓа до објективно решение. Логичкото размислување се заснова на факти и докази до кои се доаѓа на различен начин применувајќи различни техники и методи. Ваквиот начин на размислување никогаш не се заснова на претпоставки, желби, чувства и емотивни состојби.

2.1 Анализа и решавање логички натпреварувачки задачи

Анализата на проблемската или логичката задача е првиот и основен чекор кој мора објективно да го направиме за да го започнеме патот до конечното решение. Со помош на **анализата** ја разложуваме задачата или проблемската ситуација, откриваме факти или елементи на целината, откриваме поврзаности со цел да дојдеме до резултатот.
Анализата на задачите вклучува три чекори кои можат да се сумираат на следниов начин:

Слика 1: Чекори при анализа на проблемска задача

Ако анализата на задачите е внимателно организирана и ги следи соодветните чекори, може да се примени при решавање на многу задачи и да се дојде до конечно решение, токму поради вклучената детална, внимателна и интегрирана анализа.

Обидете се!

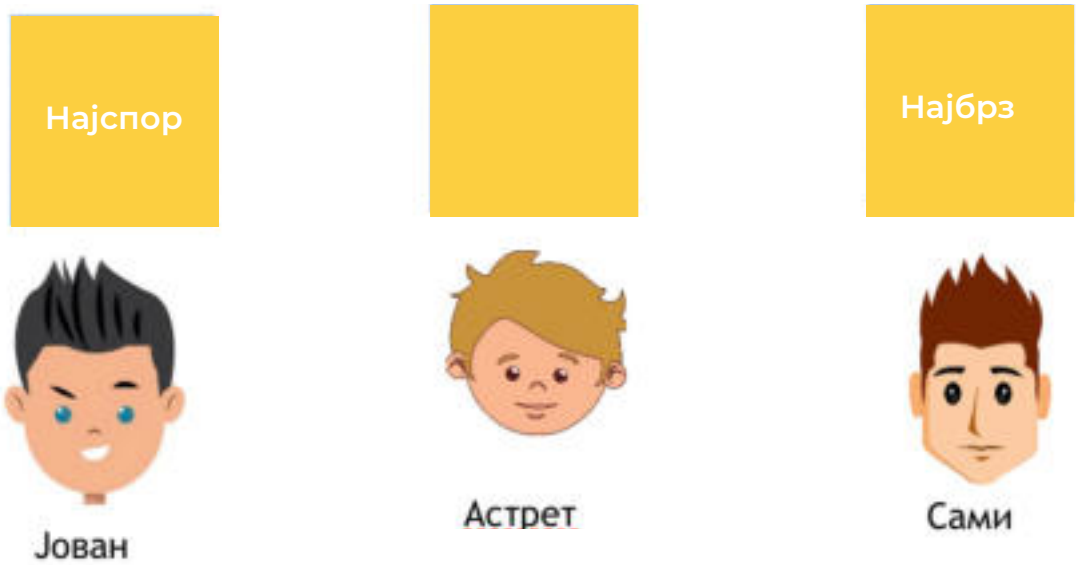
Обидете се да истражите кој е обратен или спротивен процес на анализата, ако знаеме дека анализата е процес на разложување на задачата на посебни елементи или делови!

Ајде да направиме анализа на неколку проблемски задачи преку примена на некои од техниките кои ги наведовме, притоа имплементирајќи го стекнатото знаење за логичко размислување.

Задача

Јован, Астрет и Сами се врсници и учат во исто училиште. Сите тројца се добри атлетичари кои освоиле повеќе награди на државни и меѓународни натпревари. Кој од нив е најспор, ако сите наведени искази се точни?

1.Јован не е најспор
2.Астрет е најбрз
3.Сами не е најбрз



Во еден од исказите наведено е дека **Астрет** е најбрз и затоа, ликот на **Астрет** го поврзуваме со квадратчето означено како „најбрз“. Сега останууваат две полиња непополнети. Исто така, наведено е дека **Јован** „не е најспор“ и затоа го поврзуваме со второто квадратче. Останува празно само првото поле означено како „најспор“ и таму го поврзуваме **Сами**. Според тоа, на прашањето кој е најспор, точен одговор е **Сами**.

При решавање на поставената задача ја употребивме техниката **апстракција**, бидејќи ги извлековме најважните карактеритики на задачата. Идентификуваните карактеристики нè упатија од каде треба да почнеме да ја испитуваме проблемската задача, по кој пат да одиме и како да дојдеме до можното решение. Исто се случува и при решавање на проблемска ситуација во **информатиката**. Секогаш се работи со информациите кои се важни за решавање на проблемската ситуација, а непотребните информации се исфрлаат или занемаруваат за да се реши проблемот.

Задача

Слично како и претходната задача, обидете се самостојно да дојдете до точното решение на следната задача!

Ако сите искази се точни, каде се наоѓа богатството?

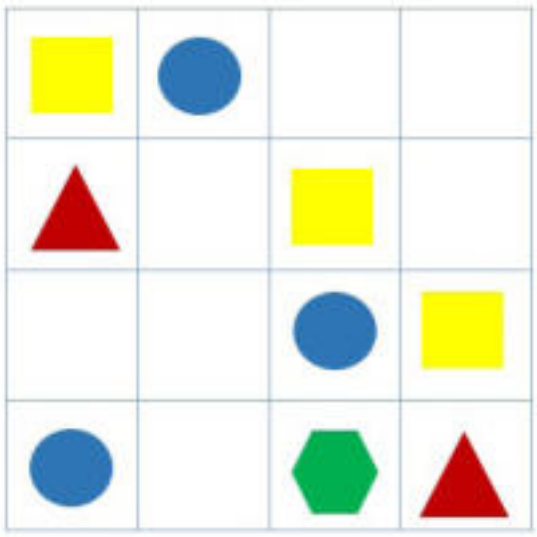
1 
Богатството не е во 2

2 
Богатството е во 1 или 3

3 
Богатството не е тука

Прашања и задачи

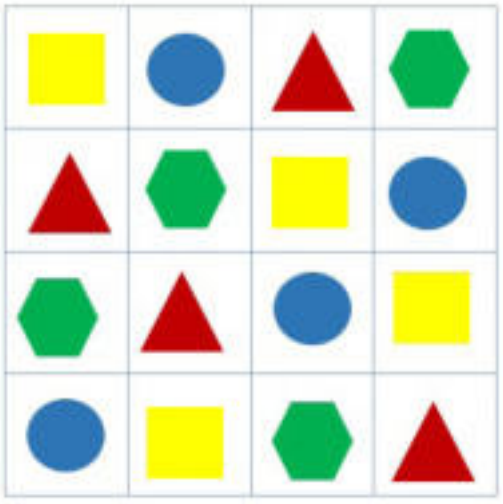
Судоку е една од најпопуларните игри кои во себе вклучуваат логичко размислување и процес на елиминација на фактите кои не се од интерес за доаѓање до резултатот.
Ајде да решиме една едноставна Судоку задача!



Во табелата, во некои полиња се распоредени геометриски фигури, а некои полиња се празни. Празните полиња треба да се пополнат со следните фигури:

Секоја геометриска фигура треба да има ист број фигури, распоредени така што хоризонтално и вертикално секогаш ќе има само една фигура од ист вид. Забележуваме дека кругот може да се додаде во табелата само еднаш и затоа треба да се определи соодветното место, односно во хоризонатална и во вертикална насока треба да постои само една таква фигура. Ајде да го најдеме соодветното место. Во првата, третата и четвртата редица (хоризонтално), веќе постои фигура круг, така што како избор останува втората редица.

Но, во кое поле во втората редица? Во која колона?
Колона еден, два и три веќе содржат фигура круг и тие испаѓаат од опција. Според тоа, првата, третата и четвртата редица, како и првата втората и трета колона ги исфрламе од како можност. Единствена можност останува фигурата круг да ја додадеме во втората редица и во четвртата колона, и тоа е точен одговор. Ако ги пополниме сите полиња, согласно со примерот со кругот, тогаш доаѓаме до следниот резултат:



Обидете се!

Обидете се да решите Судоку применувајќи ги правилата за пополнување на полињата со бројки! Постојат многу електронски примери, но во печатените медиуми можат да се најдат во печатена верзија.

Процесот на **елиминација**, при решавање на логички натпреварувачки задачи, овозможува исфрлање на фактите кои не се значајни и кои не нè водат кон конечното решение на проблемската ситуација, како што е во случајот Судоку. Во секојдневието, елиминацијата најчесто се употребува при решавање на тестови кои имаат понудени повеќе можни одговори, така што се елиминираат неточните одговори за да се намали бројот на можните опции за точен одговор.

Според тоа, при анализирањето на логичките задачи употребивме техники на логичко размислување за да дојдеме до конечно решение. Тука ги употребивме техниките на апстракција, елиминација и почитување на зададените правила. Но, при решавање на проблемски ситуации и задачи, како дел од информатички или компјутерски процеси, се користат и други техники кои произлегуваат од потребата и природата на задачата, со единствена цел, на најоптимален начин, да се стигне до конечно решение.

Запомни!

Анализата на проблемските ситуации и задачи треба да биде објективна и не пристрасна, односно да не се вклучуваат елементи на субјективност. Постапката за анализа вклучува три чекори: разложување на задачата на составни елементи, одредување на врските помеѓу елементите и реструктурирање согласно со добиените резултати. Најчести техники при анализа на задача или проблемска ситуација се: апстракција, елиминација, почитување на правила, изнаоѓање на сличности и др.

2.2 Поврзаност на задачата со концепти од компјутерската наука –информатички концепти

Логичкото размислување и решавањето на логички натпреварувачки задачи се вештини кои можат да се употребуваат во повеќе научни дисциплини, но најчесто се поврзуваат со компјутерите и **информатичките концепти**. Затоа, неслучајно се вели дека логичкото размислување е чекор пред **програмирањето**, како процес на создавање на апликативни решенија.

Според тоа, **програмирањето се дефинира како процес на пишување програма со помош на програмски јазици**. Луѓето кои ги креираат програмите се викаат **програмери**.

Од почетокот на создавање на програмата програмерите извршуваат бројни чекори, кои секој на посебен начин ни отвораат пат до решението, односно до создавањето на програмата. При овој процес програмерите се среќаваат со информатичките концепти. Поради тоа, во оваа наставна содржина ќе се запознаеме со дел од нив.

2.2.1 Структури на податоци

Структурите на податоци се посебна форма или начин на прибирање, организирање, обработка и чување на податоците. Најчесто употребувани структури се: низи, поврзани листи, стекови/купови, редови, приоритетни редови, бинарно дрво и сл.

Според тоа, структурите на податоци можат да се употребуваат при креирање на:

- Листа со чекори за доаѓање до одредено решение.
- Низа од инструкции за да се стигне до некоја дестинација.
- Упатство за пополнување на формулар или шема, и сл.

Задача

Да се креира низа со чекори, употребувајќи ги наведените ознаки со цел да се добие следната слика:

Чекор 1	2	3	4	5	6	7	8	9	10
Чекор 11	12	13	14	15	16	17	18	19	20
Чекор 21	22	23	24	25	26	27	28	29	30

Преку решената задача, успеавме да креираме низа со инструкции за конечното решение, односно обојување на мрежата од квадратчиња според прикажаната слика.

Задача

Дали сте слушнале за Кулите на Ханои? Еве еден пример!
Во овој пример ќе бидат прикажани чекорите за извршување на задача која содржи само три дискови.

Целта е да се преземат чекори за префрлање на дисковите од едно на друго место, притоа да се внимава на редоследот.

Чекор 1: дискот со број 1 го префрламе во третиот ред.
Чекор 2: дискот со број 2 го префрламе во вториот ред.
Чекор 3: дискот со број 1 го префрламе во вториот ред, веднаш над дискот со број 2.
Чекор 4: дискот со број 3 го префрламе во третиот ред.
Чекор 5: дискот со број 1 го префрламе во првиот ред.
Чекор 6: дискот со број 2 го префрламе во третиот ред, над дискот со број 3.
Чекор 7: дискот со број 1 го префрламе во третиот ред, над дискот со број 2.

Според наведените чекори успеавме да ги префрлиме сите дискови од едно на друго место, притоа внимавајќи на правилата за преместување која ги дава играта. Со седум чекори, дисковите од првиот ред ги префрливме во последниот ред, при што редоследот на дисковите остана ист.

При извршувањето на задачата **„Кулите на Ханои“**, се запознавме со **стек/куп** структура на податоци, чијашто главна карактеристика е редење, така што прво се земаат оние елементи кои се на врвот од купот. Имено, за стек велиме дека се почитува принципот „**прв влегува, последен излегува“.** Секако, при префрлањето се почитуваат правила кои произлегуваат од природата на самата задача.

Обидете се!

Со помош на наставникот по биологија обидете се да креирате низа во која ќе го прикажете животниот циклус на пеперутката. Колку периоди има животниот циклус на пепертутката? Може ли да се прескокне некој од наведените циклуси?

2.2.2 Бинарни броеви

Да се потсетиме!

Кој е јазикот на компјутерите? Како се претставуваат податоците со помош на овој јазик?

Бинарниот броен систем се состои од две нумерички вредности, цифрите 0 и 1. Само една цифра 0 или 1 се вика **бит**. Со еден бит, можеме да претставиме „точно“ или „неточно“, „да“ или „не“, „зафатено“ или „слободно“ и многу други ситуации.

Со помош на бинарните броеви, податоците во компјутерите се претставуваат во **низи од нули и единици**. Претворањето на буквите во еден збор во бинарни броеви претставува **бинарен код**. Секој природен број може да биде претставен во низа на бинарни броеви, на пример:

Природни броеви	0	1	2	3	4	5	6	7	8	9	10
Бинарен приказ	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

За да конвертираме еден природен број во бинарен приказ, тогаш употребуваме основа 2 со степен: 0, 1, 2, 3 итн. На пример, бројот 5 во бинарен приказ би изгледал вака: 0101, а овој резултат доаѓа од:

5	Степен со основа 2	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
		$4 + 1 = 5$			
	Приказ со битови	0	1	0	1

Ајде да ја пополниме следната сложувалка, така што во секое поле ќе го запишеме бинарното претставување на природниот број:

Хоризонтално:

1) 2
3) 5
4) 34
5) 20

Вертикално:

1) 17
2) 10
3) 10

Со помош на логичко размислување да ја решиме следната натпреварувачка задача:

Задача

Учениците од VII одделение решавале тест по географија кој содржел прашања со повеќе понудени одговори. Наставничката ги бележала точните одговори со 1, а неточните со 0 во табела, како што е прикажано подолу:

	Прашање 1	Прашање 2	Прашање 3	Прашање 4	Прашање 5	Прашање 6
Ана	0	0	1	1	1	0
Сара	1	1	1	1	1	1
Алмир	1	0	0	0	1	1
Сафет	0	0	0	0	1	0
Елона	0	0	1	1	0	1
Амира	1	0	1	0	1	1
Јелена	0	0	1	1	1	1

Од табелата, наставничката сакала да креира извештај за тоа кој материјал е најслаб, а кој најмногу усвоен, со цел да креира дополнителни начини за целосно усвојување на материјалот.

Одговорете на прашањата согласно со податоците кои се наведени во табелата!

- Кое прашање го одговориле најмногу ученици?
- Кое прашање го одговориле најмалку ученици?
- Кој ученик има највисок резултат?
- Кој е најслаб ученик?
- Кои ученици треба да посетат дополнителна настава според резултатите?

2.2.3 Криптографија

Криптографијата е научна дисциплина која се занимава со проучување на методите за испраќање и примање пораки на начин кој е разбирлив само на оние за кои е наменета. Всушност, криптографијата овозможува користење на тајни кодови кои на корисниците им гарантира:

- заштита на податоците од неовластен пристап;
- сигурна и заштитена комуникација;
- веродостојност и тајност на пораките;
- проверка на потеклото на податоците, односно информациите и сл.

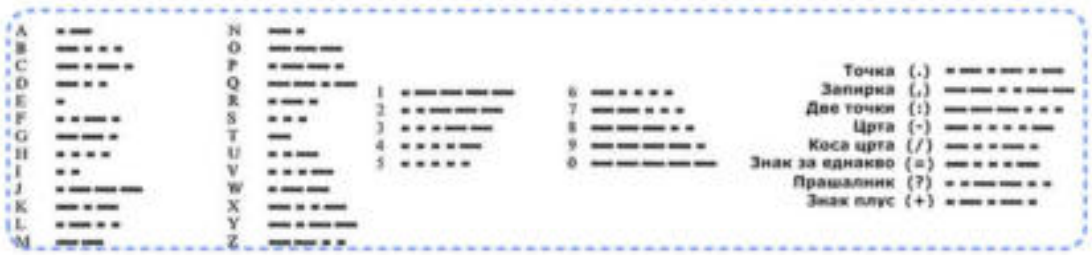
Криптографијата се употребува при работа со кредитни картички, компјутерски лозинки, електронска трговија и при вршење на други активности, каде што од особено значење е заштита на корисникот.
За криптографијата можеме да кажеме дека е процес кој вклучува трансформации на податоците или пораките:

- шифрирање (енкрипција);
- дешифрирање (декрипција).

Шифрирањето или **енкрипцијата** е процес на трансформација на пораката во шифра, а **дешифрирање** или **декрипција** е обратен процес кој ја трансформира шифрираната порака во оригинална форма. Овој процес се вика криптографски процес, истиот е прикажан на следната слика:

Слика 2: Криптографски процес

Најпознат пример за криптографија е Морзеовата азбука, која се користела при телеграфска комуникација. На следната слика прикажана е секоја буква, број и интерпункциски знак претставен со Морзеовата азбука:



Но, најчесто употребувано шифрирање е Цезаровата шифра. Ајде да дешифрираме, т.е. да декриптираме порака употребувајќи го Цезаровиот модел!

Задача

Јана и Азра се другарки од VII одделение. Заедно треба да креираат проектна задача „Бебедност при користење на интернет“ и за таа цел потребен им е интернет. При форматирање на компјутерот се избришала лозинката за пристап до интернет. Братот на Јана ја помни лозинката, но сака да си поигра со нив давајќи им шифрирана лозинка и неколку инструкции:

1. дешифрирањето може да се изврши со Цезаровиот модел;
2. цезаровиот модел користи таен број (клуч);
3.при шифрирање употребени се неколку цели броеви, како клучеви за поместување.

Лозинка	?	?	?	?	?	?	?	?	?	?	?
Клуч	1	1	3	3	2	1	2	2	3	1	2
Шифра	S	M	T	M	O	Љ	Џ	P	Ж	J	Џ

Ајде да започнеме! Во табела ќе ја напишеме азбуката, а потоа ќе ги употребиме клучевите. Во нашиов случај дадени се клучеви за дешифрирање, односно поместување за 1 , 2 и 3 места:

Табела 1: Цезаров модел со клуч 1

Буква	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	Ш
Клуч 1	1	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ
Шифра	Ш	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	

Табела 2: Цезаров модел со клуч 2

Буква	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	Ш
Клуч 2	1	2	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч
Шифра	Џ	Ш	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	

Табела 3: Цезаров модел со клуч 3

Буква	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	Ш
Клуч 3	1	2	3	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц
Шифра	Ч	Џ	Ш	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	

Во задачата ни е дадена шифрата и клучот, ајде да ја најдеме лозинката!

Клуч	1	1	3	3	2	1	2	2	3	1	2
Шифра	S	M	T	M	O	Љ	Џ	P	Ж	J	Џ
Лозинка	И	Н	Ф	О	Р	М	А	Т	И	К	А

Запомни!

Логичко размислување и решавање логички натпреварувачки задачи се еден чекор пред програмирањето. Програмирањето е процес на пишување на програма. Луѓето кои креираат програми се викаат програмери. Логичките задачи и нивното решавање се поврзуваат со информатички концепти, како што се: низи, листи, стекови/купови, редови, приоритетни редови, бинарни броеви, криптографија и сл.

ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

Задача 1

Креирајте низа од инструкции кои ќе го прикажат патот од влезот на училиштето до вашата училница.

Задача 2

Природните броеви 12, 18 и 36 претворете ги во бинарни броеви!

Задача 3

Бинарниот број 11100111 претворете го во природен број!

Задача 4

Бројот 236 да се претвори во бинарен број!

Задача 5

Креирајте шифрирана порака употребувајќи го Цезаровиот модел!

Задача 6

Истражете ги карактеристиките на игрите Packman и Tetris. Со кои информатички концепти можеме да ги поврземе? Објаснете!

Задача 7

Алма сака да и испрати порака на Ана: „ИЗЛЕГУВАМЕ“ со Цезаровиот код. Како гласи пораката која Алма ќе ја испрати?

Задача 8

Стефан добил порака од Амир: „ПГОП“ кодирана во Цезаровиот код. Што сакал да му каже Амир на Стефан, ако Стефан знае дека клучот е број три (3)?

Задача 9

Јасмин испратил порака на Томи: „УНК ОТДСА“. Но, пораката ја пресретнал Елвин. Елвин знае дека Јасмин и Томи се допишуваат на начин што секоја буква од забуката се поместува за шест (6) места. Така ја дешифрирал пораката. Која порака ја добил Елвин при дешифрирањето?

Дополнителни задачи:

За увежбување на логичкото размислување, решавањето логички натпреварувачки задачи може да се употребуваат прирачниците со задачи и објаснување на задачите од Дабар кои се објавени на официјалната страница **www.talent.mk**.

Интернет адреси за решавање логички натпреварувачки задачи и игри кои можете да ги употребувате се:

www.bebras.org
https://www.bbc.com/bitesize/articles/zqnc4wx
http://digit.mile.mk/

Напредно програмирање во визуелна околина

Вовед во програмирање во визуелна околина
Графичко програмирање. Запознавање со програмата Scratch
Интерактивни програми со настани
Изработка на програми со посложени проблемски ситуации
ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

3.Вовед во програмирање во визуелна околина

Да се потсетиме!

Што се програми? Наведи дефиниции за програмирање! Како се изработува програма? Кои програмски јазици ги познаваш?

Логичкото размислување и решавање на логички натпреварувачки задачи претставуваат вовед во **програмирањето**. Не случајно се вели дека логичкото размислување е чекор пред **програмирањето**.

Слика 1: лого на Scratch

Според тоа, **програмирањето може да се дефинира и како начин или вештина на размислување со цел за доаѓање до конечно решение на проблемска задача или проблемска ситуација**.

Следствено на тоа, целта на програмирањето е создавање сет на инструкции кои компјутерот мора да ги следи за извршување специфични операции или реализирање на посакувано дејствие.

При пишување и развивање програми, програмерот употребува група програми, кои ја сочинуваат **околината за програмирање**. Тие програми се:

- **едитор** – во кој се пишува изворниот код на програмата;
- **компајлер** – го претвора изворниот код во машинска инструкција;
- **библиотека на готови програми** – колекција или збир од веќе напишани мали и често користени програми во програмската околина;
- **дебагер** – се употребува за пронаоѓање грешки.

Најинтересно и најпривлечно е **програмирањето во визуелна околина**, бидејќи при решавање на проблемските ситуации, креирање на чекори и инструкции, се употребуваат визуелни објекти, како што се: ликови, позадини, звуци, движења, ефекти и сл., со кои програмирањето станува лесно и забавно.

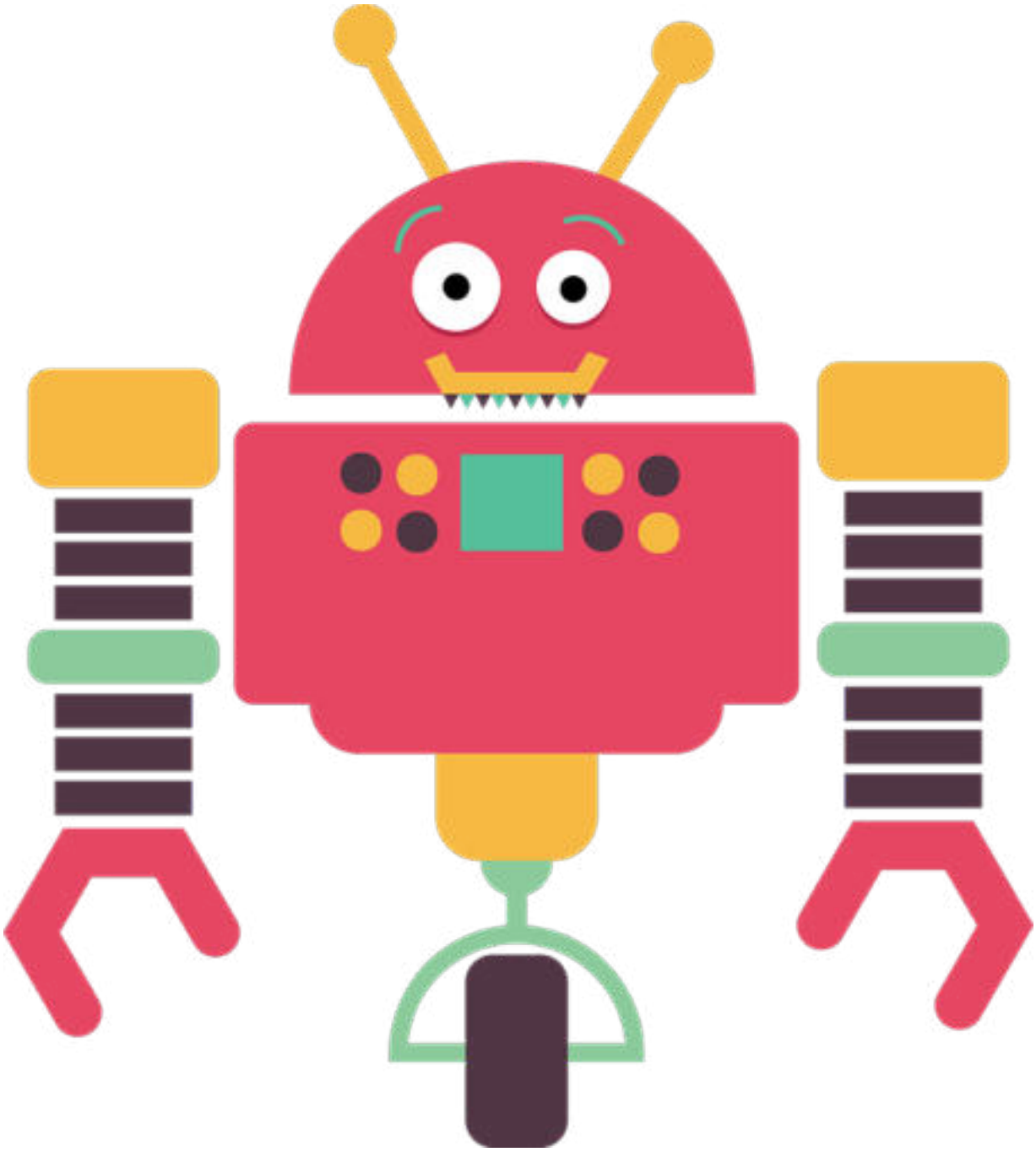
Во процесот на изучување на програмирањето постојат повеќе апликации кои се примери за визуелна околина за создавање програми, од кои најчесто употребувана е Scratch.

Scratch е визуелен програмски јазик со кој се креираат интересни приказни, анимации, игри и сл. Програмирањето во Scratch се базира врз правење проекти со помош на објекти, т.н. фигури. Неговиот сет од наредби потсетува на сложувалка, тие овозможуваат широк спектар на можности за креирање сложени програмски решенија и совладување на проблемскиот и алгоритамски начин на размислување.

Во оваа тема ќе се запознаеме со **програмскиот јазик Scratch**, неговата работна околина и начинот на креирање проектни активности како решенија на задачи.

Клучни поими!

графички приказ, координати, константи, променливи, искази (наредби), интерактивни програми, објекти, настани (случки).



3.1 Графичко програмирање. Запознавање со програмата Scratch

Програмирањето во визуелна околина претставува занимлив начин на изучување на концептот и чекорите при креирање програма. Главна карактеристика на ваквиот начин на програмирање е тоа што сите алатки, фигури, наредби, па дури и кодовите се **графички прикажани**. Ваквиот интерфејс им овозможува на учениците на најлесен и наједноставен начин да креираат проекти. При креирањето на проектите учениците ја развиваат својата креативност и креативно размислуваат, почнуваат систематски да резонираат и заеднички да работат. Најчесто употребуван програмски јазик кој се имплементира во визуелна околина за програмирање е **Scratch**.

Scratch е програмски јазик кој овозможува креирање на интерактивни приказни, анимации, игри, музички инструменти и сл. Бидејќи сите алатки се графички прикажани со соодветни икони, учениците можат да влечат и да комбинираат блокови од кодови, да вклучат ефекти на анимација, аудиоефекти и сл., без притоа да ја учат синтаксата или текстуалните наредби на програмскиот јазик.

Визуелната околина за програмирање **Scratch** може да се користи online на следниот линк: **https://scratch.mit.edu**. Оваа страница претставува официјална страница на **Scratch**. Најпрво се креира корисничка сметка со која корисникот се најавува со свое корисничко име и лозинка, потоа се започнува со креирањето на проекти. Постои и offline верзија која бесплатно се симнува од официјалната страницата.

3.1.1 Стартување на Scratch

Offline верзијата на Scratch се стартува со двоклик врз нејзината икона која се наоѓа на работната површина (**Desktop**) на компјутерот, при што се отвора следниот прозорец преку кој пристапуваме до алатките за работа:

Слика 1: Почетен екран на Scratch

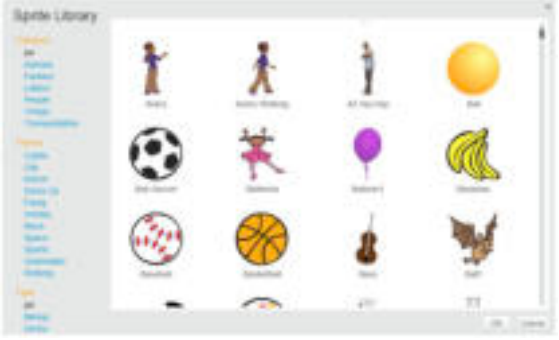
Ајде детално да го опишеме работниот прозорец:

- насловната лента го прикажува името на програмата;
- копчињата за работа со прозорецот овозможуваат минимизирање, максимизирање и затворање на проектот;
- алатката за избор на јазик овозможува корисникот да избере на кој јазик да му бидат прикажани менијата и наредбите. Scratch е развиена во повеќе јазици;
- преку менито file корисникот изведува активности со проектот: зачувување, отворање нов проект, отворање постоечки проект и слични активности;
- преку мени лентата пристапуваме до наредбите и уредувањето на објектите кои ќе се употребуваат во проектот;
- на работната површина се поставуваат фигурите кои се ликови во проектот;
- преку делот за додавање фигури, корисникот може да направи избор на ликови кои ќе бидат дел од проектната задача;
- изборот на позадина овозможува примена на соодветна тема на проектната задача;
- палетата на наредби содржи блокови со наредби или искази кои корисникот може да ги избере. Бидејќи тие имаат различна функција, односно се применуваат за различни цели, различно се обоени. Преку обојувањата корисникот знае во која категорија припаѓа наредбата;
- местото за програма претставува збир на сетови од инструкции кои корисникот ги избрал и со влечење ги комбинирал за да се прикаже и изведе активност.

3.1.2 Запознавање со основните алатки на Scratch

Објекти (Sprite)

Кога корисникот започнува да креира програма, по стартувањето на визуелната околина на екранот прво ја забележува фигурата или ликот кој има доминантна улога во пректот, односно ќе биде програмиран за изведување на активности. На почетокот ликот е логото на Scratch, но може да се менува со кликнување на иконата Choose Sprite From Library и избор од галеријата, како што е прикажано на сликата:



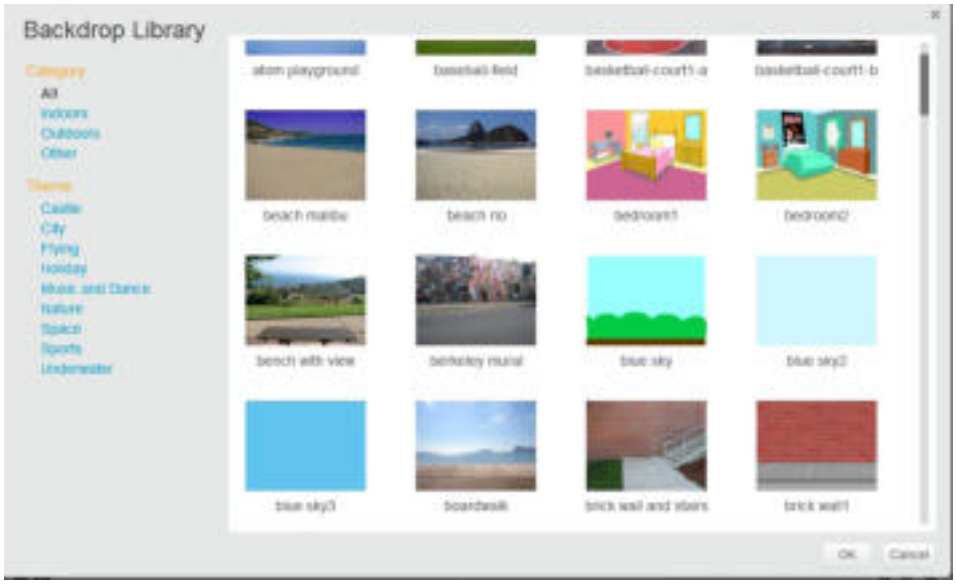
Слика 2: Галерија на ликови во Scratch

Од сликата може да се забележи дека ликовите се поделени по различни категории. Објект (**sprite**) се додава со кликнување врз категоријата, потоа врз објектот и на крајот на копчето **OK**.

Објектите освен од галеријата може да се додаваат од компјутерот, камерата или од креиран лик од страна на корисникот.

Позадина (Background)

И позадината се поставува на ист начин како и ликовите. Се кликнува врз иконата за менување на позадина и се појавува прозорецот за избор на теми на позадини, како што е прикажано на сликата:



Слика 3: Галерија на теми за позадини

Исто така, темите на позадините можат да се додадат и од компјутерот или самостојно да се креираат.

Наредби (Искази)

Во програмирањето исказите претставуваат наредби кои му кажуваат на компјутерот што да направи. Во Scratch, исказите се наоѓаат во делот scripts:

Слика 4: Видови искази (наредби) во Scratch

Од сликата забележуваме дека исказите, односно наредбите се поделени во категории: движење, изглед, звук, молив, настан, контрола и сл. Секоја категорија на настан или наредба е означена со посебна боја. Кодовите на овие наредби се во вид на блокови во боја, како и категоријата.

Креирањето на програмски чекор се врши со кликнување и влечење на блокот и поврзување на блоковите, односно чекорите во една целина. Кога ќе се постават овие наредби, се започнува со стартување на креираниот програм.

3.1.3 Креирање едноставни програми во Scratch

За да ги осознаеме можностите и функционалностите на алатките на Scratch, ќе креираме едноставна програма. Темата на проектната задача е танцување. Ајде да започнеме!

Најпрво ја стартуваме програмата Scratch и избираме лик. За оваа проектна активност го избираме ликот Cassy Dance и го сместуваме во средината на работната површина. Како ја наоѓаме средината на работната површина?

Обидете се!

Со помош на наставникот по математика обидете се да дефинирате што е тоа координативен систем? Како се претставува?

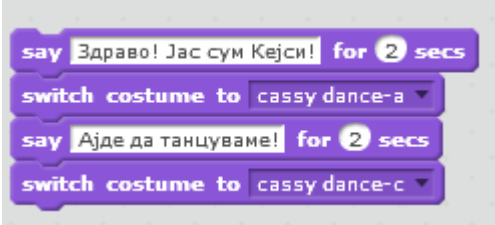
Работната површина во **Scratch** е претставена во вид на **координативен систем** составен од x и y оските. Координативниот систем е во вид на математичка мрежа или шема составена од многу точки кои се викаат **координати**. Местото на точките всушност ги определува вредностите на x и y. Центарот или средината на работната површина има вредност: x=0 и y=0. Секоја точка надвор од средината има посебна вредност на x и y. Всушност, преку вредностите на координатите се определува местото, односно позицијата на објектот.

Ликот **Cassy Dance** претставува објект во проектната задача. За да го сместиме во средината на работната површина, преку менито **motion** ги избираме блоковите: **set x to 0 и set y to 0**, преку кои вредностите на x и y ќе ги означиме со 0:

Слика 5: Доделување вредности на x и y

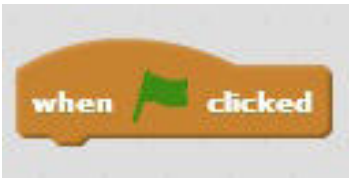
При изборот на блокови со наредба за поставување на вредности на оските, забележуваме дека местото во кое треба да се определи вредноста е во вид на круг, односно поле за внесување. Тоа ни укажува, дека согласно со потребите, вредноста можеме да ја менуваме мануелно. вредноста можеме да ја менуваме мануелно, согласно со потребите. Според тоа, во програмирањето овие резервирани места за вредности се нарекуваат **варијабли** или **променливи**.

На почеток, објектот, во нашиов случај Cassy Dance, може да даде воведни пораки. Тоа го правиме преку менито Scripts и опцијата Looks, при што ги избираме следните блокови со наредби:



Слика 6: Сет наредби за пишување порака

Од кодот може да се забележи дека освен што се дава наредба за порака, се прават и промени во положбата на објектот, со што се привидува дека ликот се движи. За да ја започнеме реализацијата на сетот наредби, пред сетот виолетови кодови, преку менито **Events**, го додаваме следниот блок со наредба:

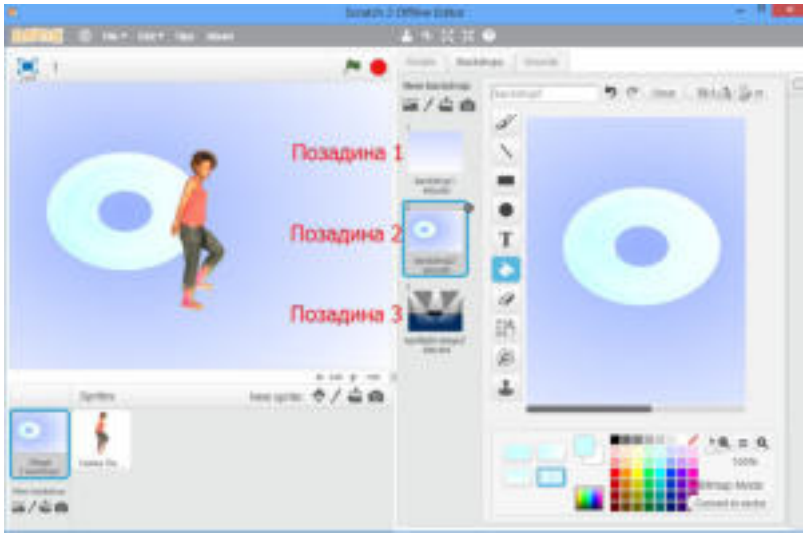


Слика 7: Додавање почеток за извршување на сетот инструкции

Со постапката за додавање на **позадина (Backdrop)** ја избираме **Spotlight Stage 2**. Додека е селектирана позадината, во мени лентата избираме **Backdrop** и добиваме ваков изглед:

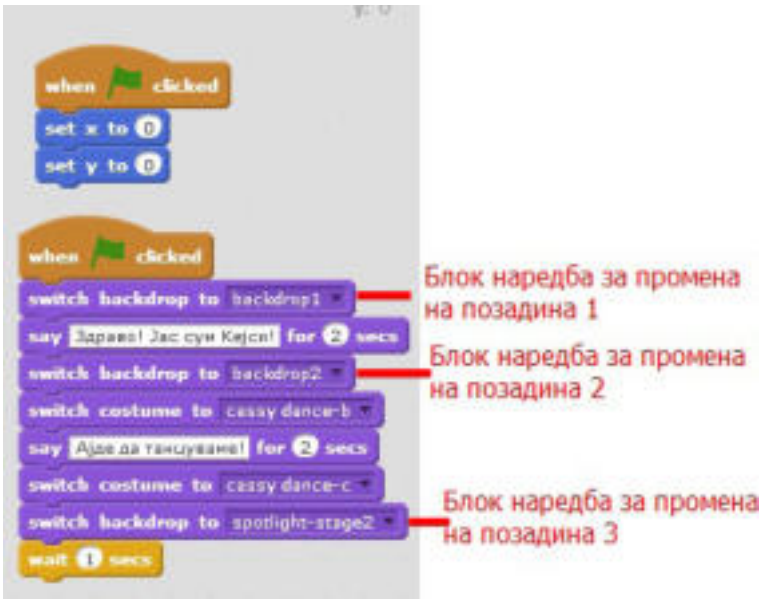
Слика 8: Уредување позадина

Правиме копија од **backdrop 1** со десен клик и опцијата **duplicate**. Се појавува уште една бела позадина. Овие две бели позадини можеме да ги уредиме избирајќи ја алатката **Fill with color**, избираме боја од палетата со бои која се наоѓа во долниот дел на прозорецот и ја боиме позадината. Истото го повторуваме и со **backdrop 2**.



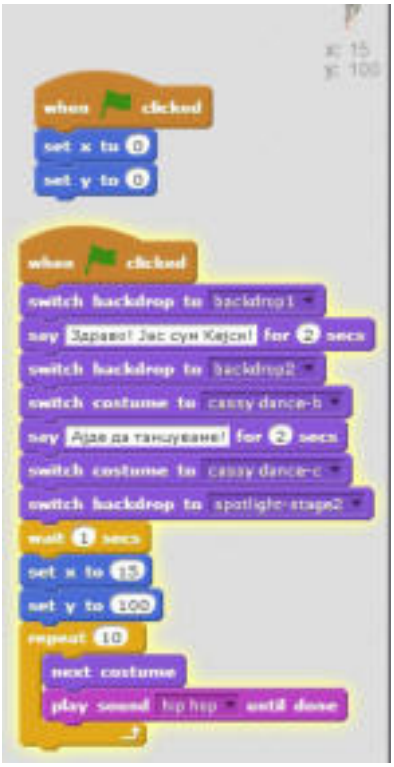
Слика 9: Креирање и уредување позадини

Следно, овие додадени, креирани и уредени позадини да ги додадеме во кодот на проектната активност. Преку менито **Look**, ги додаваме блоковите со кодови за менување на позадината. Во прилог редоследот на наредбите:



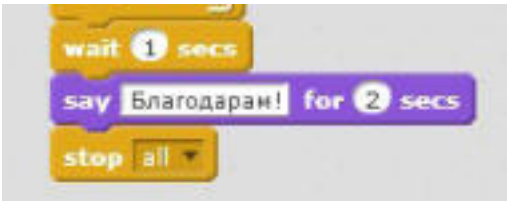
Слика 10: Сет од наредби за промена на позадина во проектна активност

За да го поставиме објектот на бината, додаваме вредности на координатите, преку менито motion, наредбите set x to 15 и set y to 100. Вака поставениот објект ќе биде програмиран да изведува танц и притоа да биде придружуван од музика. Блок-наредбите, односно исказите кои треба да се додадат се со следниот редослед:



Слика 11: Додавање циклуси за повторување

За да го завршиме кодот додаваме сет од инструкции. На пример, нашава проектна задача ќе ја завршиме со порака и запирање на сите процеси:



Слика 12: Завршување на програмата

Со тоа го завршивме нашиот прв проект. Ајде да го прегледаме!

На левата страна на работниот прозорец, во делот каде што се прикажуваат објектите, во десниот горен агол, се наоѓаат иконите за започнување и завршување на проектот:

Слика 12: Завршување на програмата

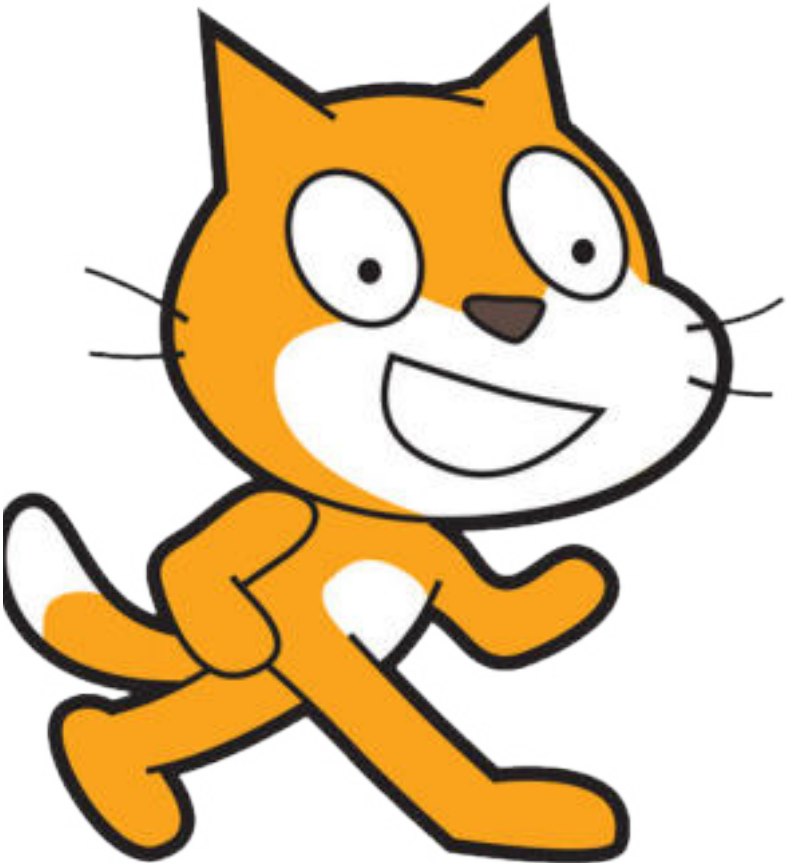
Со кликнување на зеленото знаменце започнуваме со извршување на сетот од инструкции, при што визуелно гледаме што сме изработиле, доколку пак сакаме да го стопираме приказот, кликнуваме на црвеното копче за стопирање.

Запомни!

Програмите за графичко програмирање овозможуваат креирање низи од наредби за извршување на програма преку примена на алатки, фигури, наедби и сл. кои имаат графички приказ. Scratch е програмски јазик кој овозможува креирање интерактивни приказни, анимации, игри и сл. Наредбите во Scratch се прикажани во вид блокови кои наликуваат на сложувалки. Тие се комбинираат со влечење и спојување. Работната површина на Scratch може да биде претставена во вид математичка мрежа или шема составена од точки – кооридинати. На објектите додадени во проектната задача во Scartch им се доделуваат блокови со искази (наредби). Исказите (наредбите) му кажуваат на компјутерот што да направи. Некои искази содржат резервирани места за вредности и тие се викаат варијабли или променливи.

Прашања

1.Што означуваат блоковите во Scartch? На што наликуваат?
2.Како се поделени блок-наредбите? Наведи неколку категории!
3.Што означува зеленото знаменце кое се наоѓа во горниот дел на приказот на проектот?
4.Што ни покажуваат координатите на објектот во проектната задача?
5.Како се поставува позадина на проектна задача?



3.2 Интерактивни програми со настани

Да се потсетиме!

Дали се сеќавате кога прв пат го споменавме терминот „интерактивни програми“? Во кои теми на обработка? Можете ли да се сетите како го дефинираме?

Терминот **„интерактивни“** досега повеќепати сме го споменувале во различни теми и во различни области. На пример, часот по информатика можеме да го карактеризираме како **интерактивен**, бидејќи постојат активности, комуникација за реализација на активностите. Честопати се води дискусија на часовите за решавање задачи и проблемски ситуации, што значи дека сите гласно размислуваме, даваме констатации, ставови, но и одговори на многу пршања.

Исто се случува и кога употребуваме образовен софтвер. Програмот ни дава инструкции, а ние како корисници ги почитуваме инструкциите и согласно со нив преземаме соодветни активности.

Кога станува збор за компјутерска игра, следењето на правилата на играта всушност ја претставува таа интеракција помеѓу корисникот и самата игра. Секој корисник сака да ја постигне целта, односно да освои победа и затоа мора да постапува согласно со правилата на играта.
Во оваа наставна единица ќе креираме проект кој во себе вклучува настани од **интерактивен карактер**.

Главниот лик во нашава интерактивна игра е **Dinoaur1**. Тој ќе ги извршува математичките пресметки на задачите кои ќе ги зададеме. За почеток, ќе креираме низи од блок наредби кои ќе пресметуваат **количник на два внесени броеви**.

Ајде да започнеме со програмирање!

Најпрво ја стартуваме програмата **Scratch**. Со помош на алатката ножици (**delete**) го бришеме објектот, односно ликот **Cat**. Преку алатката за додавање на објекти, во делот **spirit**, додаваме објект од постоечката галерија. За нашиот проект ќе го избереме објектот **Dinoaur1**. Следно, додаваме позадини (**backdrop**). Едната позадина ќе ја креираме и уредиме ние со помош на алатките и палетата со бои кои ни ги нуди програмата, а втората позадина е слика од нашиот компјутер. Почетниот екран би изгледал вака:

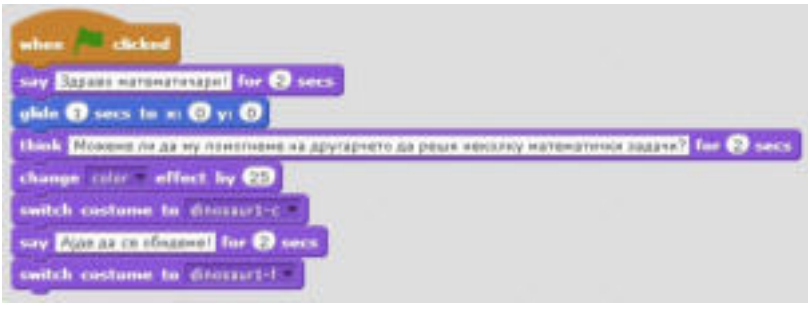
Слика 1: Додавање објекти и позадина на проектот

Кликнуваме врз Dinoaur 1 и започнуваме со креирање на низите од блок команди:

Слика 2: Определување на местото на објектот

Наведената низа од блок наредби, значи дека со кликнување на зеленото знаменце кое означува почеток на извршување на низите од инструкции, вредноста на **x** и **y** да биде **-122** и **-85**, што значи дека со тоа сме ја определиле локацијата на **Dinoaur1**.

Следните блок кодови се почетни настани во проектот:



Слика 3: Низа од блок-наредби кои објавуваат порака

А тоа значи, дека кога ќе го кликнеме зеленото знаменце ќе се изврши низата од блок-наредби. Најпрво, **Dinoaur 1** нè поздравува во „**здраво математичари**“, се движи, односно се „**излизгува**“ до средината на работната површина. Кога **Dinoaur 1** доаѓа до средината на работната површина се прикажува како да размислува, при што се менува неговата боја и неговиот приказ кој наликува на движење.

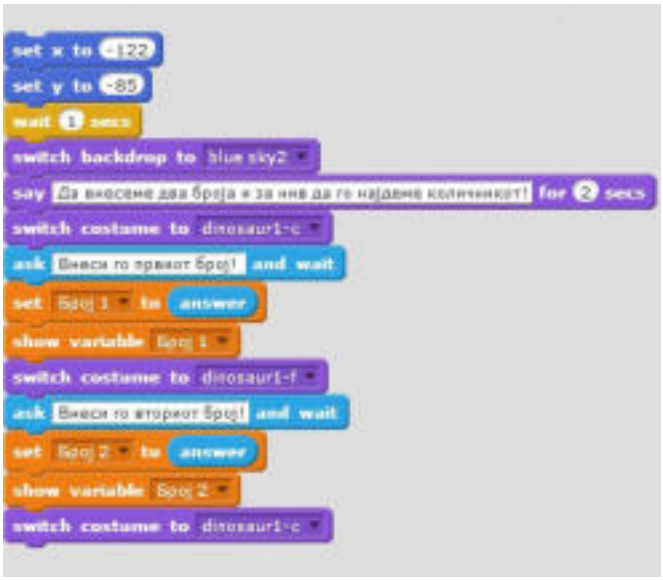
Понатаму, продолжуваме со низи од блокови кои ќе нè водат во нова ситуација. **Dinoaur1** ќе ги дава инструкциите, а ние од тастатура ќе ги внесуваме податоците кои се бараат. Всушност, **Dinoaur1** е нашиот математичар и тој ќе ни го соопшти резултатот од извршената пресметка на количник на два внесени броја. За таа цел, најпрво ќе дефинираме три варијабли, односно променливи кои ќе имаат вредности внесени од тастатура и резултат од пресметка. Постапката за креирање на варијабли е следната: преку менито **Scripts** пристапуваме до категоријата **Data** и таму кликнуваме на **Make variable**. Бидејќи оваа варијабла ќе ја употребуваме и во други ситуации во проектната активност, ја кликнуваме опцијата и даваме име на варијаблата.

Забелешка!

Името на варијаблата, односно променливата можете да го дадете по желба, но да потсетува на тоа каква вредност ќе има и која е улогата. На пример, во задачата дефиниравме варијабли со имиња Број1 и Број2, но можеше да се дефинираат и како деленик и делител согласно со математичките називи.

Слика 4: Креирање варијабли/променливи

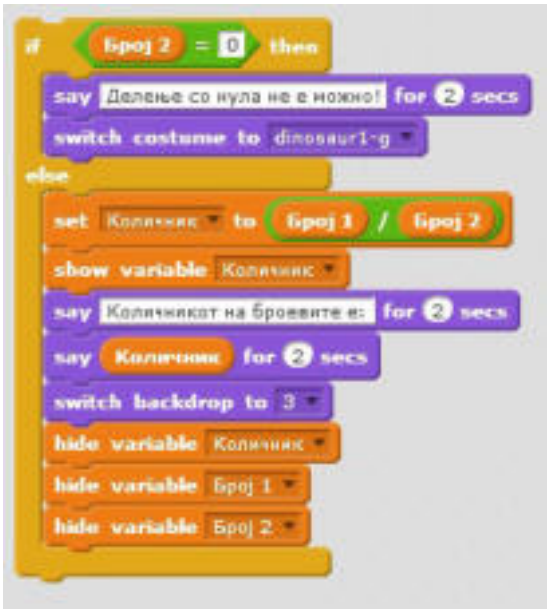
Во конкретната проектна задача креираме **три варијабли**. Првата варијабла ќе ја содржи вредноста што ќе ја внесеме од тастатура за првиот број (**број 1**), втората варијабла ќе ја содржи внесената вредност за вториот број (**број 2**) и третата варијабла (**количник**) ќе прикаже вредност која произлегува од решението. Ајде соодветно да ги примениме креираните варијабли!



Слика 5: Примена на блок-наредби од категоријата Data

Преку менито **Motion** им даваме вредности на **x** и **y**, со цел да го вратиме **Dionaur1** во првобитна положба, односно на почетното место. Преку менито **Looks**, даваме блок наредба за менување на позадината (**backdrop**) и блок наредба за приказ на порака. **Dinoaur1** го менува својот изглед и кажува да го внесеме првиот број. Тука се применува блок наредбата преку која ни се овозможува опција за внесување на број од тастатура и приказ на полето со внесената вредност. Истото го повторуваме и за вториот број, како што е прикажано на сликата.

Следно што треба да направиме е да провериме дали вредноста на вториот број е 0, бидејќи математичкото правило вели дека делење со нула не е можно. За таа цел, преку менито **Control** ја применуваме блок наредбата: „ако е задоволен условот, тогаш да се извршат следните инструкции“ и „ако не е задоволен условот, да се извршат следните инструкции“, односно:

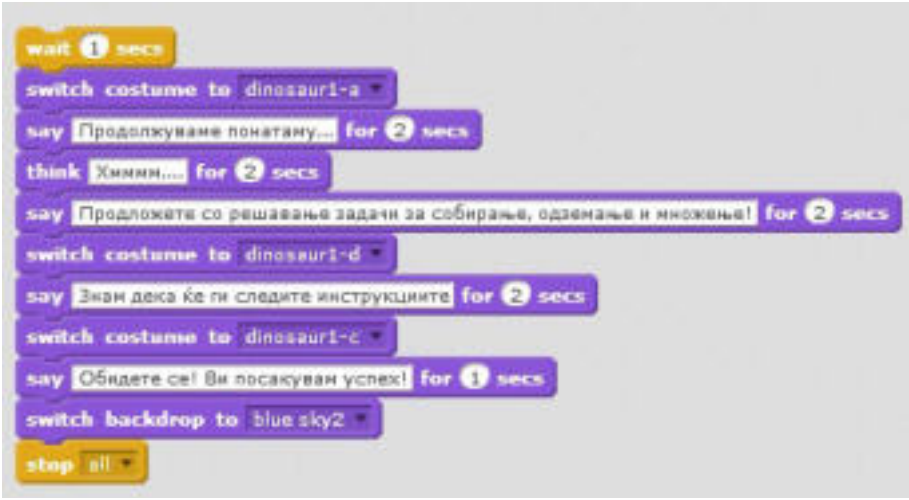


Слика 6: Примена на блок-наредбата If/else

Или, ако вредноста на вториот број е еднаква на нула, да се прикажува порака: „делење со нула не е можно“ и **Dinoaur1** да го менува изгледот. Ако условот не е задоволен, тогаш:

- да се направи пресметка на количникот;
- да се прикаже вредноста на количникот;
- количникот да се прикаже како порака на **Dinoaur1**;
- да се премине во нова тема, односно позадина;
- кога ќе преминеме во новата позадина, варијаблите да не бидат видливи.

Со тоа ја завршивме пресметката на количникот на двата броја. Проектот може да се заврши по желба. На пример, во нашава ситуација **Dinoaur1** продолжува со пораки:



Слика 7: Блок-наредби за приказ на пораки

Постапката за приказ на пораки на екранот ги применувавме и претходно. Пораките зависат од креативноста на корисникот и како тој визуелно го замислил решението на проектната задача. Со наредбата „**stop all**“, означуваме крај на извршувањето на низите од наредби.

Со кликнување на зеленото знаменце започнуваме со извршување на низите од блок наредби, а со тоа вршиме проверка на она што сме го направиле. Овој проект е од интерактивен карактер, бидејќи **Dinoaur1** ги задава инструкциите, ни кажува како да постапиме, а ние ги извршуваме тие напатствија преку преземање активност од тастатурата.

Задача

Dinoaur1 ни помогна да најдеме количник од два внесени броеви. Продолжете ја задачата со тоа што ќе креирате низи од блок наредби кои ќе пресметуваат и други математички операции, како собирање на два броја, разлика на два броја и производ.

Размисли!

Дали може да се креира низа од блок-наредби кои ќе вршат пресметки на математички операции со три броја? Што ни недостига во програмот на Dinoaur1 кој го креиравме за вршење пресметки преку внесување на два броја? Објасни ја постапката! Прикажи каде би направил измени!

Запомни!

Програмите каде корисникот може да внесе вредности на променливите со кои се прикажуваат резултати и зависат од влезните единици се викаат интерактивни програми. Варијаблите се променливи кои добиваат вредности при внес на податок од тастатура или, пак, како резултат на пресметка. Варијаблите/променливите се креираат преку менито Scripts и категоријата Data.

Прашања

1.Кои програми се нарекуваат интерактивни?
2.Што се варијабли? Која е постапката за креирање варијабли?
3.Со која наредба се овозможува внесување вредност на варијаблата?
4.Како корисникот внесува вредност на варијаблата во програмата?
5.Зошто служи наредбата If...Else?

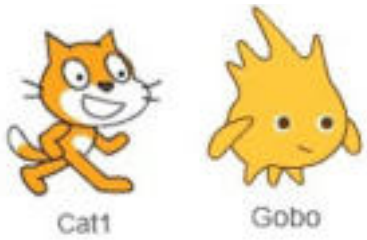
3.3 Изработка на програми со посложени проблемски ситуации

Креирањето игра во **Scratch** е малку покомплексна работа, бидејќи вклучува повеќе **објекти и настани**. На секој објект му се доделуваат инструкции или сетови од инструкции за да се дојде до постигнување на целта, односно до конечна победа во играта. Притоа, задолжително треба да се предивидат и настаните кои можат да се случат. Секако, за нив очекувано е да се понудат соодветни решенија. На пример, ако се освои победа, да се добие порака, или, пак, да се слушне победнички звук, а ако се изгуби да го врати објектот на почеток.

На пример ќе креираме игра во која има два објекти: **Cat** и **Gobo**. Cat ќе се движи според движењето на глувчето – на лева и десна насока. Повеќе **Gobo** паѓаат од небото, а **Cat** треба да собере најмногу од вкупниот број подароци.

Ајде да започнеме нов проект во **Scratch**.

Со кликнување на копчето **Choose Sprite from library**, додаваме нов објект/лик. Од галеријата со објекти/ликови го избираме Cat и Gobo:



Слика 1: Објекти/ликови во проектот

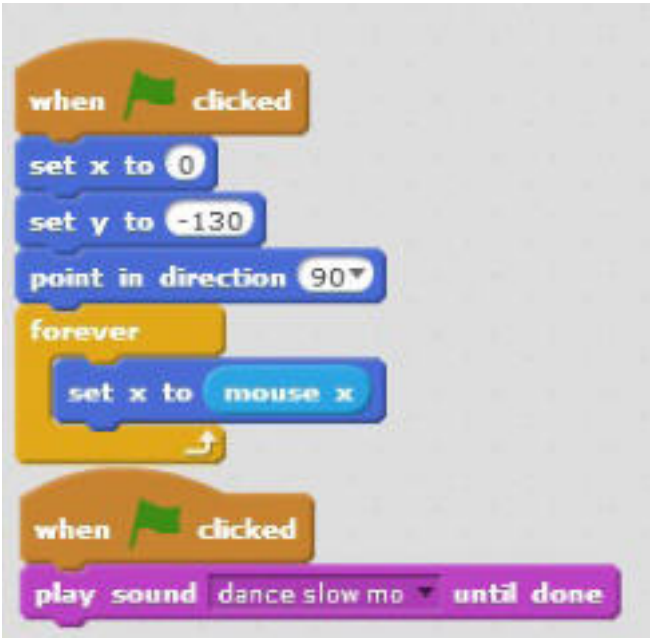
Преку алатката **Shrink**, ги менуваме димензиите на објектите/ликовите. Кликнуваме врз алатката Shrink, а потоа кликнуваме врз објектот, односно ликот, онолку пати колку што сакаме да го намалиме објектот. Во нашиов случај ќе ги намалиме димензиите на двата објекти:

Слика 2: Менување на димензиите преку алатката Shrink

Додаваме тематика или позадина преку алатката **choose backdrop from library**. За нашава игра ќе ја употребиме темата **Gingerbread**:

Слика 3: Тема на проектот

Следно, на секој објект треба да му се доделат сетови или низи од инструкции за да овозможиме објектите да се анимираат и да вршат некакво дејство. На пример, објектот **Sprite1** треба да се движи лево-десно и да ги прибира **Gobo**:



Слика 4: Сет од инструкции за движење на Sprite 1

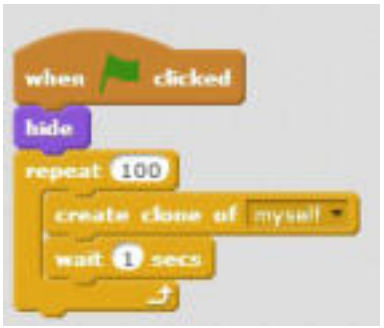
Забелешка!

Од наведените блок-наредби, забележуваме дека постојат блок-наредби претставени во различни бои, а тоа ни покажува дека припаѓаат на различна категорија наредби.

Ајде со наредби да ги објасниме блок по блок!

Со кликнување на зеленото знаменце ќе започне процесот на извршување на инструкциите кои следат. Притоа, им даваме вредности на координатите **x** и **y**, **0** и **-130**. Означуваме насока на движење **90 десно**. Објкетот/ликот ќе се движи по **x оската** според движењето на глувчето. Исто така, со кликнувањето на зеленото знаменце ќе се започне со емитување на **звучен фајл**. Во нашиов случај од галеријата со звуци избран е звукот dance **slow mo**.

Исто така, инструкции треба да му зададеме и на објектот/ликот **Gobo**. Според нашава задача, од небото надолу треба да паѓаат повеќе примероци или **клонови на Gobo**. Иако, ние сме дефинирале само еден објект – подарок, сепак во играта тој објект ќе биде побројен. Да ја видиме постапката:



Слика 5: Креирање клонови на објект

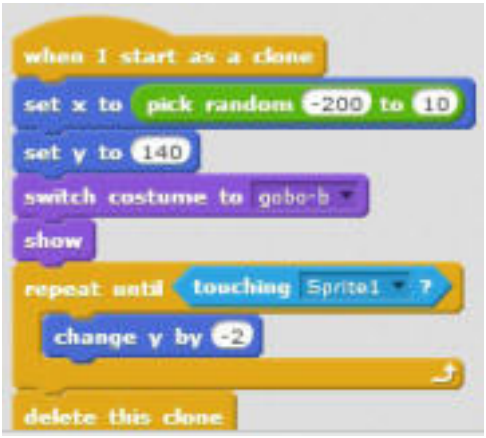
Оваа низа или сет од наредби значи дека со кликнување на зеленото знаменце се означува почеток на извршување на низата команди за креирање на клон, односно копија на објектот/ликот. Со блок наредбата repeat се овозможува креирање клонови на секоја секунда сè додека не се постигне вредноста 100.

Следно што треба да се направи е да се креираат блокови со наредби кои ќе му кажуваат на објкетот – клон како да постапува и кои дејствија да ги изврши:

Слика 6: Инструкции за објектот – клон

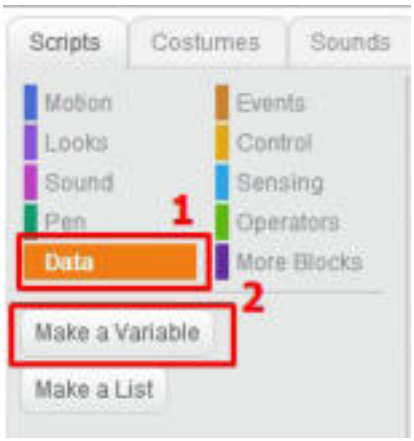
А тоа би значело следново: кога ќе се појави објектот – клон, ќе се движи надолу сè до објектот **Spirit1**. Кога ќе го допре **Spirit1** ќе започне емитување на музички фајл кој се вика **zoop**. Со цикличната наредба **repeat** ќе се менуваат четирите различни облици на објектот **Gobo** на секоја секунда. Веднаш потоа, ќе се избрише, односно ќе го снема објектот – клон.

Во следниот сет од инструкции се укажува дека објектот - клон ќе се појавува на произволни координати во ранг од -200 до 200 по однос на **x оската** и ќе се движи надолу започнувајќи од вредноста 140 по однос на **y оската**. Притоа, ќе го менува својот приказ сè додека не го допре објектот Spirit1. На крајот објектот – клон ќе се изгуби.



Слика 7: Поставување произволни координати за појавување на објектот - клон

Во играта ни недостигаат поени. Со следната низа наредби ќе овозможиме броење. Кога **Gobo** ќе го допре **Spirit 1** објектот, тогаш ќе се менуваат поените. Најпрво, треба да додадеме нова променлива која на почетокот ќе има вредност 0, а потоа ќе се зголемува за 1. Додавањето променлива се врши со кликнување на опцијата **Data** од менито **Scripts**, па потоа на опцијата **Make variable**:

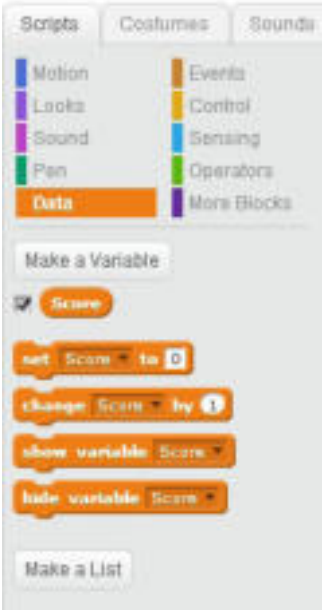


Слика 8: Креирање нова варијабла

Се појавува следниот прозорец преку кој корисникот дава име на варијаблата и означува дека креираната варијабла ќе се употребува за селектираниот лик:

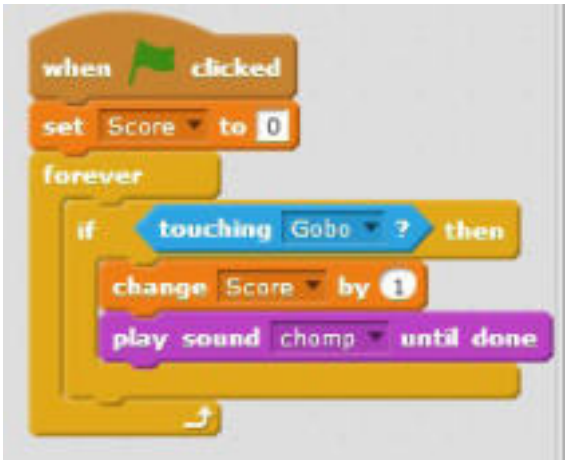
Слика 9: Дефинирање нова варијабла

Со кликнување ОК, се додаваат следните блок-наредби кои се однесуваат на новата варијабла:



Слика 10: Блок-наредби кои се однесуваат на новата варијабла Score

Откако ја креиравме варијаблата, креираме низа од блок-наредби кои треба да се извршуваат, т.е. да ги бројат поените:



Слика 11: Броење поени

Ајде да видиме како функционира играта! Кликнуваме на зеленото знаменце и почнуваме со игра.

Задача

Креирајте едноставна игра во Scratch. Игрите кои ги креиравте споделете ги со другарчињата и играјте следејќи ги инструкциите.

Запомни!

Проектите кои вклучуваат повеќе објекти и повеќе настани се посложени за креирање. Секој објект посебно се програмира, односно му се задаваат низи од блок-наредби за извршување со цел создавање движење, активност или дејство. Во програмите објектите се опишуваат со состојба и начин на однесување. Во реалниот свет објект може да биде: автомобил, велосипед, топка или некој лик кој е дефиниран со состојба и однесување. Настаните се случуваат зависно од состојбата или однесувањето на објектот. Тие треба да се предвидат и за нив да се понуди соодветно решение.

Прашања

1.Што се објекти?
2.Како се програмираат објектите?
3.Која е врската меѓу објектите и настаните?

ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

Задача 1

Обидете се да креирате проектна задача во Scratch, применувајќи ги блоковите со искази (наредби) кои практично ги работевме на часот. Вметнете позадина, додадете објект, придвижете го објектот, додадете звук и сл.

Задача 2

Креирајте едноставна игра во Scratch која вклучува повеќе објекти и настани.

Задача 3

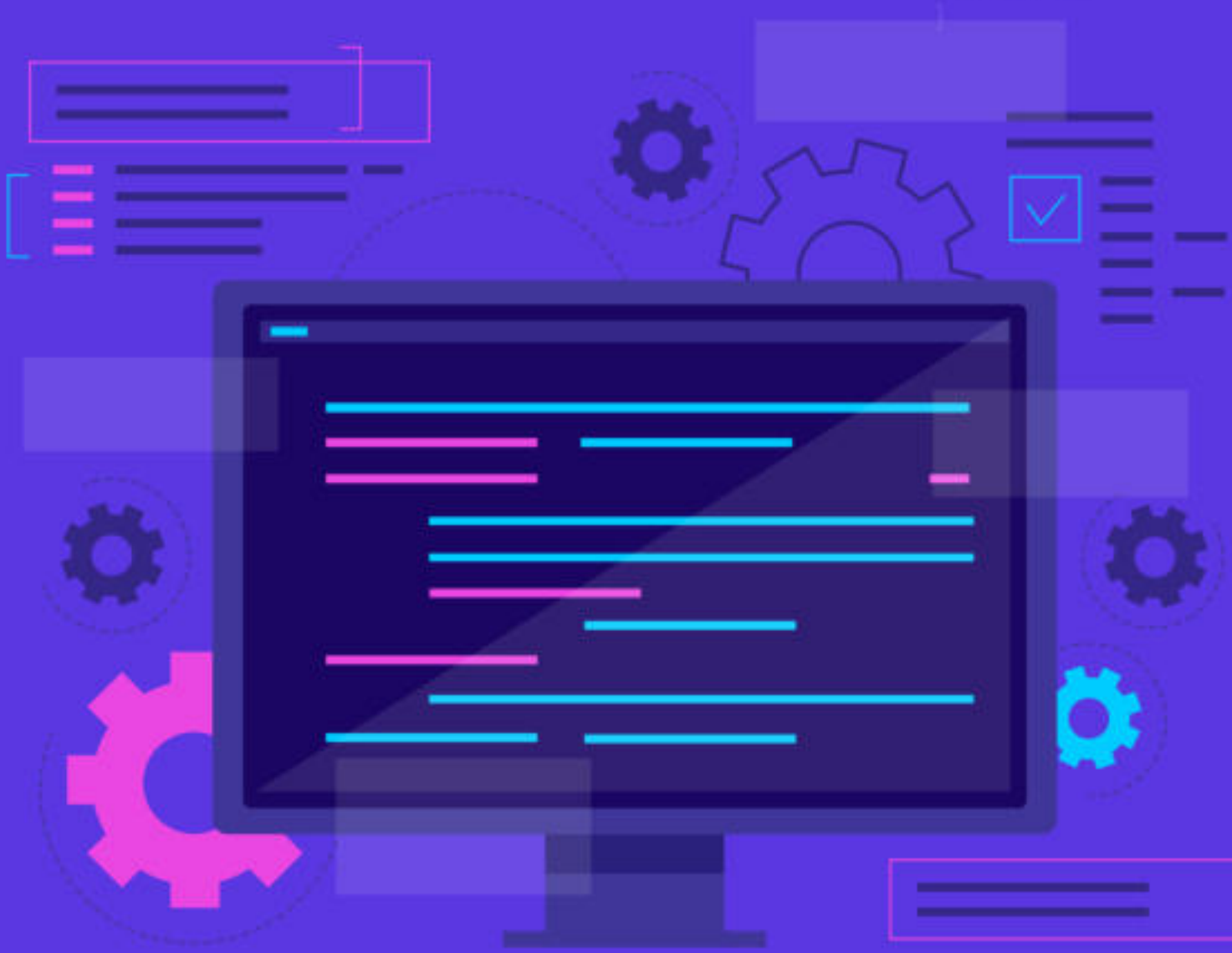
Да се креираат посебни програми кои ќе пресметуваат плоштина и периметар на квадрат, правоаголник и круг.

Задача 4

Креирајте проект во Scratch од интерактивен карактер, во кој при внес на број од тастатура ќе проверува дали е позитивен или негативен.

Програмирање преку стандарден структурен програмски јазик

Вовед во програмирање преку стандарден структурен програмски јазик
Процес на изработка на една програма
Запознавање со основните елементи на интегрирана околина за програмирање
Изглед на готови пример програмски кодови
Извршување на готови пример програми
Основни елементи на програмскиот јазик C++
Искази
Изработка на програми
Аритметички операции и изрази
Константи и променливи
Искази (техники) за внесување на податоци во програмата
Споредбени изрази
Структура за повторување во циклус до исполнување на услов
ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!



4 Вовед во програмирање преку стандарден структурен програмски јазик

Што ќе научиме?

Да работиме во интегрирана околина за програмирање Code::-Blocks и да креираме програма во програмскиот јазик C++.

Живееме во време на брзи промени и уште побрз технолошки развој во сите делови на нашето општество. Речиси не може да се замисли ниту една работа која ќе се изврши без примена на компјутерот. Тргнувајќи од фактот дека компјутерот е електронски уред кој се употребува за прибирање, чување и обработка на податоците со помош на програмите кои се инсталирани на него, заклучуваме дека тој не би бил најмоќната алатка која нуди решение за секоја проблемска ситуација, за различни цели и од различни области.

Во нашето секојдневие употребуваме различни програми, како на пример:

- играме игри за забава;
- пишуваме есеи во програмата за обработка на текст (MS Word, Open Office Writer и др.);
- вршиме пресметки во програмата за табеларни пресметки (MS Excel, Open Office Calc и др.);
- креираме презентации во програмата за креирање и уредување мултимедијални презентации (MS Power Point, Open Office Impress, Prezi и др.);
- гледаме видеофајлови преку соодветни програми (Winamp, VLC Player, Real Player и др.);
- креираме и уредуваме слики во графички едитори (MS Paint, InDesign, Corel Draw, Photoshop и др.);
- комуницираме со роднините и пријателите преку програмите за комуникација (Viber, Messenger, WhatsApp и др.).

Постојат и други програми за специфични намени кои корисникот може да ги нарача да се креираат согласно со неговите потреби. Всушност, програмите претставуваат врската помеѓу корисникот и компјутерот. Според тоа, компјутерските програми претставуваат низи од инструкции кои компјутерот треба да ги изврши. Процесот на осмислување, креирање и поврзување на низите со инструкции се вика програмирање. Луѓето кои ги креираат програмите со помош на програмски јазици се викаат програмери.

Програмските јазици претставуваат множество од правила, симболи и

клучни зброви кои се комбинираат на различен начин со цел доаѓање до решение на задачата или проблемската ситуација. Учењето програмирање го развива логичкиот начин на размислување кој е корисен и применлив во секојдневните ситуации.

Клучни поими!

преведувач, изворна програма, извршна програма, едитор, компајлер, дебагер, исказ, споредбен израз, циклус.

4.1 Процес на изработка на една програма

Изработката на една програма било да е наменета за компјутер, телефон, таблет или друг уред претставува сложен процес, бидејќи чекорите за нејзино креирање се запишуваат во некој програмски јазик. Постојат многу програмски јазици и секој од нив има карактеристично множество на инструкции и синтакса на пишување. **Програмските јазици** обично се делат на **виши** и **нижи** програмски јазици.

Нижите програмски јазици се машински ориентирани, бидејќи зависат од машината на која се изведуваат, а тоа значи дека ако таа програма се напише на еден процесор може да не функционира на друг процесор. Затоа, програмирањето во машински код е особено тешко, бидејќи треба добро да се познава структурата и градбата на компјутерот.

За разлика од нижите програмски јазици, **вишите програмски јазици** сè почесто се употребуваат, бидејќи не зависат од машината на која се изведуваат, а тоа значи дека можат да се изведуваат на компјутери со различни процесори. Од друга страна, нивните симболи, знаци, но и нивната синтакса е многу слична со природниот јазик и се лесно разбирливи. Најчесто употребувани виши програмски јазици се: **C, C++, Fortran, Basic, Pascal, Java** и други. Без оглед кој виш програмски јазик употребуваме при креирање на програма, сепак процесот се одвива низ следните **фази**:

Ред.бр.	Фаза	Значење
1	**Пишување изворен код**	Пишувањето изворен код се врши со програмски јазик во интегрирана околина за програмирање. Притоа, оваа програма се вика **изворен код (Source Code)** кој се зачувува во меморискиte уреди на компјутерот како фајл со наставка **.cpp**, на пример: vezba.cpp
2	**Преведување на изворниот код**	Изворниот код со помош на програми за преведување кои се викаат **компајлери** го преведуваат изворниот код. При преведување на изворниот код можно е да се воочат грешки. Овие грешки се викаат синтаксни грешки и најчесто се однесуваат на грешно напишани зборови од програмскиот јазик, неправилно напишани или заборавени интерпункциски знаци и сл. При преведувањето се добива фајл од објектен код кој има наставка **.obj**, на пример: vezba.obj
3	**Поврзување во извршниот код**	Со помош на програми наречени **поврзувачи (Linkers)** се поврзуваат објектните кодови во **извршен код (Executable code)**. Извршниот код го извршува компјутерот. Притоа, се креира извршен фајл кој има наставка **.exe**, на пример: vezba.exe
4	**Тестирање на програмата**	Тестирањето на програмата служи за проверка на програмата за тоа дали ги исполнува поставените услови и доаѓа до точно решение, односно претставува еден вид истражување кое обезбедува информации за квалитетот на производот и евентуално пронаоѓање грешки. Оваа фаза претставува и процес на валидација и верификација на програмата, т.е, ги исполнува барањата наведени во документацијата, работи според очекуваното и може да биде имплементиран со сите негови карактеристики.

Табела 1: Фази при пишување програма

Обидете се!

Потсети се на Offline визуелната програма Scratch. Направи споредба за тоа каде и како ги пишувавте кодовите, како се компајлираше и како се пронаоѓаат грешките при креирање на програмите. Воочи ги сличностите и разликите.

Во процесот за креирање програми во оваа тематска единица ќе креираме програми со C++ програмскиот јазик во Code::Block интегрираната околина за програмирање.

Запомни!

Компјутерски програми се низи од инструкции кои компјутерот ги извршува. Процесот за создавање програма се вика програмирање. Луѓето кои креираат програми со помош на програмски јазици се викаат програмери. Програмските јазици се делат на нижи и виши. Фази при креирање програма се: пишување изворен код, преведување изворен код, поврзување во извршен код, тестирање и верификација на програмата.

Прашања

1.Како се вика постапката за креирање програми?
2.Кој ги креира програмите? Како?
3.Како се делат програмските јазици?
4.Кои се нижи програмски јазици?
5.Набројнекои виши програмски јазици?
6.Која е разликата помеѓу виши и нижи програмски јазици?
7.Дефинирај што е синтаксана програмски јазик!
8.Што е изворна програма? Каква наставка има фајлот на изворната програма?
9.Како се вика програмата која ја извршува компјутерот?
10.Набројги фазите при креирање програма!

4.2 Запознавање со основните елементи на интегрирана околина за програмирање

Интегрираната околина за програмирање (IDE – Integrated Development Environment) претставува софтверски пакет кој ги содржи основните алатки потребни за пишување и тестирање софтвер. Оваа апликација обично нуди многу функции за пишување, модифицирање, составување, распоредување и дебагирање на софтвер. Главната намена на овој пакет алатки е да го поедностави развојот на софтверот и да ги минимизира грешките во кодирањето.

При креирање на програма, корисниците употребуваат бројни алатки за креирање, градење и тестирање на софтверски код. Развојните алатки често вклучуваат **уредувачи (Editor) на изворен код, библиотеки за кодови, компајлери и платформи за тестирање**. Всушност, програмерот пишува и уредува изворен код во уредувачот на кодови (Editor), компајлерот го преведува изворниот код на читлив јазик којшто може компјутерот да го изврши, а дебагерот го тестира софтверот за да реши каков било проблем или грешка.

Обидете се!

Замисли една проблемска ситуација во секојдневните училишни обврски. Обиди се да размислуваш и функционираш како интегрираната околина за програмирање: креирај чекори за решавање, компајлирај ги тие чекори за да ги разберат твоите другарчиња и разговарај со нив за дополнителни можни елементи, обиди се да дознаеш дали е вистинско решение.

Интегрирана околина за програмирање која ќе ја употребуваме при креирање програми со **C++ програмскиот јазик** е **Code::Blocks**.

Code::Blocks е интегрирана околина за програмирање која поддржува компајлирање и тестирање на повеќе програмски јазици. Работи со низа компајлери. Таа е бесплатен пакет алатки за развој на софтвер во C++ програмскиот јазик со отворен код и ги содржи следните алатки: едитор за текст, програми за преведување и поврзување, како и програми за откривање грешки.

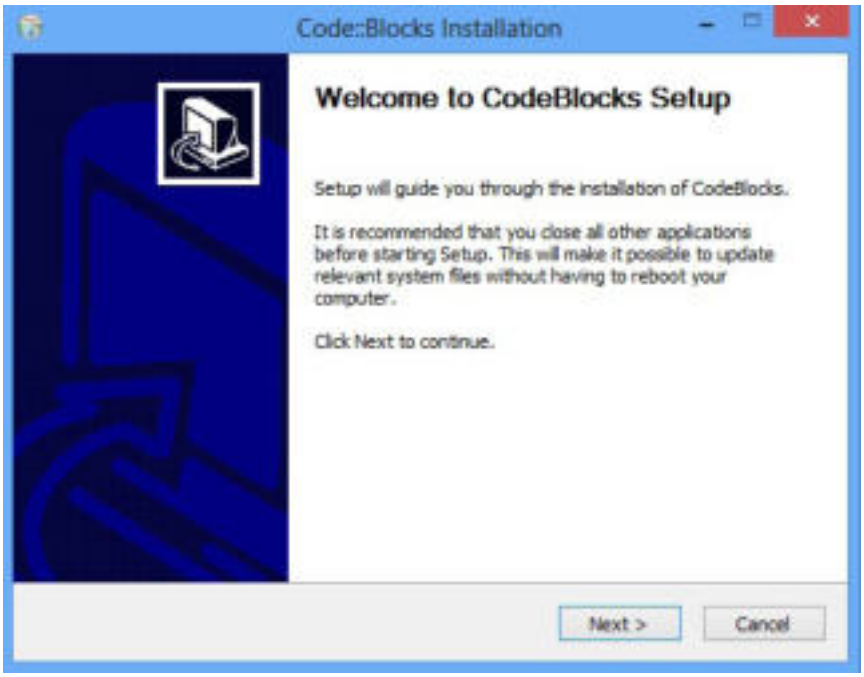
4.2.1 Инсталирање на Code::Blocks

Бидејќи, програмата Code::Blocks е бесплатна програма со отворен код можеме да ја преземеме (Download) од следната страница:

https:// //www.codeblocks.org/downloads/binaries.

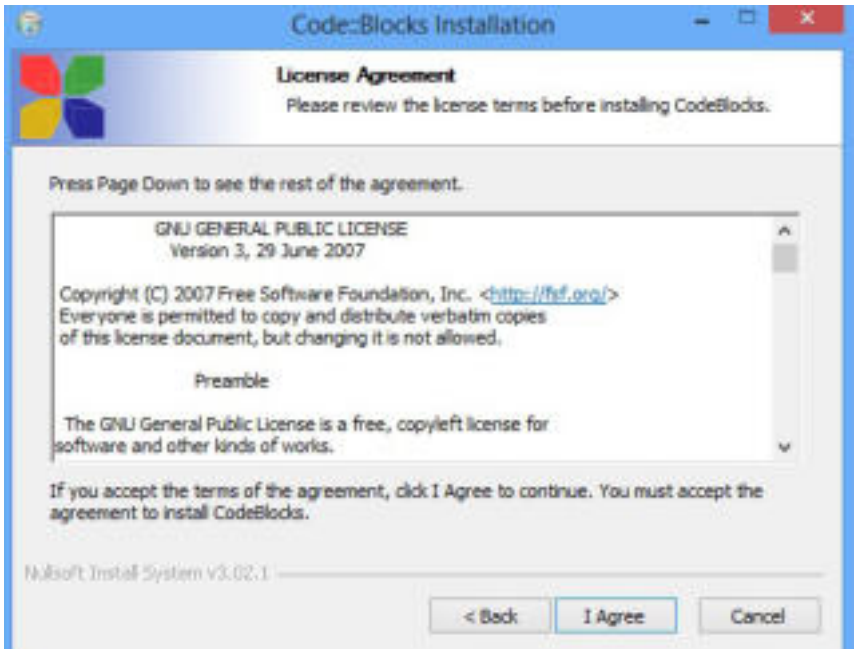
На преземениот извршен фајл му даваме наредба да биде извршен,

односно да започне процесот на инсталација со двоклик врз него. Се појавува волшебникот (**Wizard**) за продолжување на процесот на исталација, како што е прикажано на следната слика:



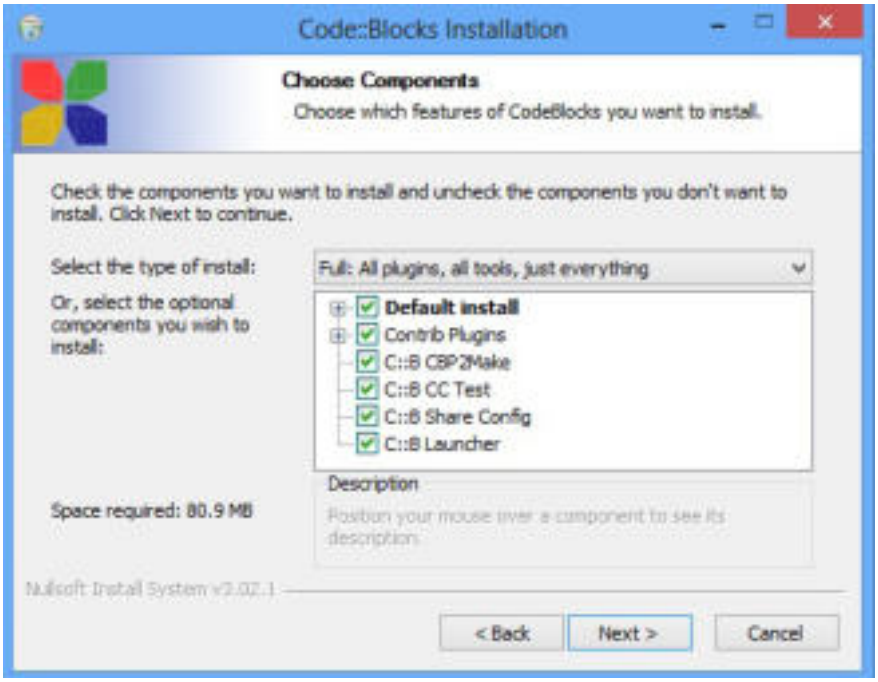
Слика 1: Почеток на процесот за инсталација на Code::Blocks

Со кликнување на копчето **Next** пристапуваме кон следниот чекор од инсталацискиот процес, а тоа е дали ги прифаќаме начелата за употреба на оваа програма:



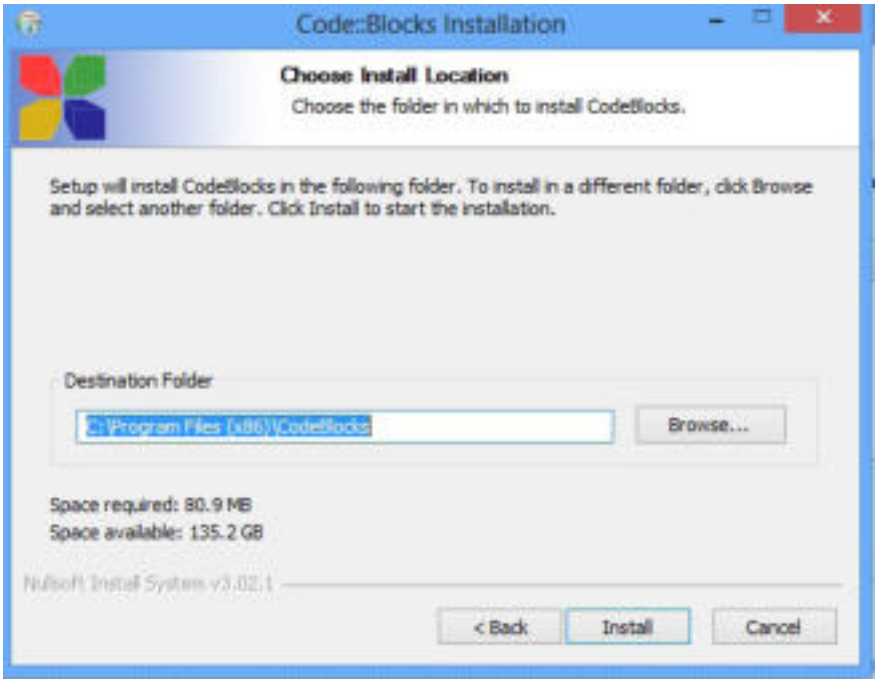
Слика 2: Прифаќање на правилата за инсталација на програмата

За да го продолжиме процесот кликнуваме на **I Agree** и продолжуваме кон следниот чекор, а тоа е избор на компоненти кои сакаме да се инсталираат како пакет на оваа програма:



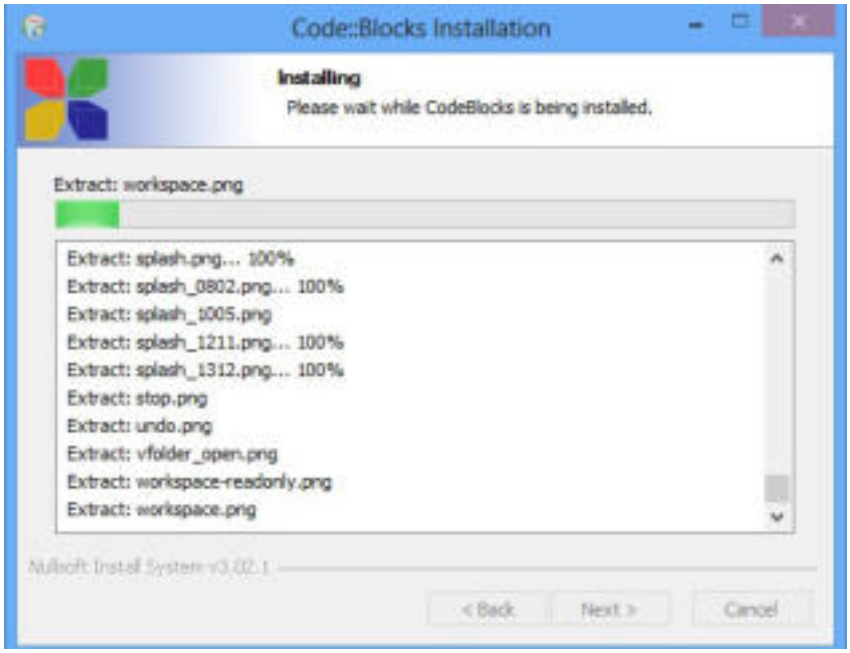
Слика 3: Избор на дополнителни компоненти во работната околина

Со кликнување на копчето **Next**, се отвора следниот прозорец преку кој го избираме фолдерот во кој ќе се сместат фајловите кои се потребни за нормално и непречено функционирање на програмата:



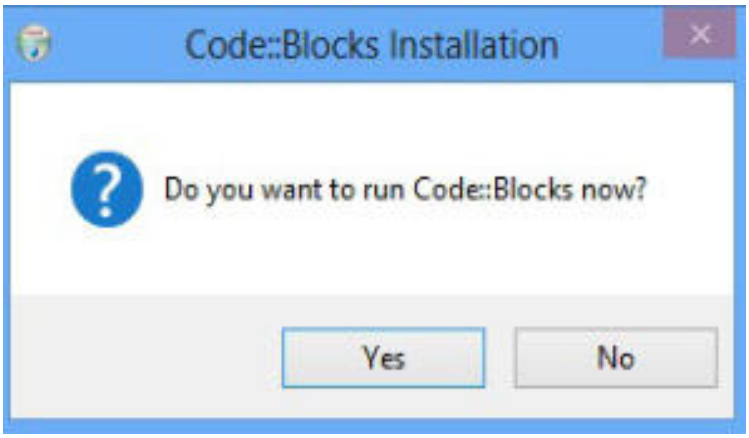
Слика 4: Избор на фолдер каде што ќе се сместат фајловите на програмата

Откако ги завршивме почетните инсталациски чекори, кликнуваме на **Install** при што започнува префрлањето на програмските фајлови во дестинацискиот фолдер:



Слика 5: Инсталација на програмата

Откако ќе се заврши овој процес кликнуваме на опцијата Next за да пристапиме кон следниот чекор, на крајот **Finish**, при што се појавува прозорецот за стартување на Code::Block:



Слика 6: Дијалог-прозорец за стартување на Code::Blocks

Од овој дијалог-прозорец ја избираме опцијата Yes, притоа се стартува програмата и ја гледаме работната површина на интегрираната околина за програмирање Code::Blocks.

4.2.2 Работна околина на Code::Blocks

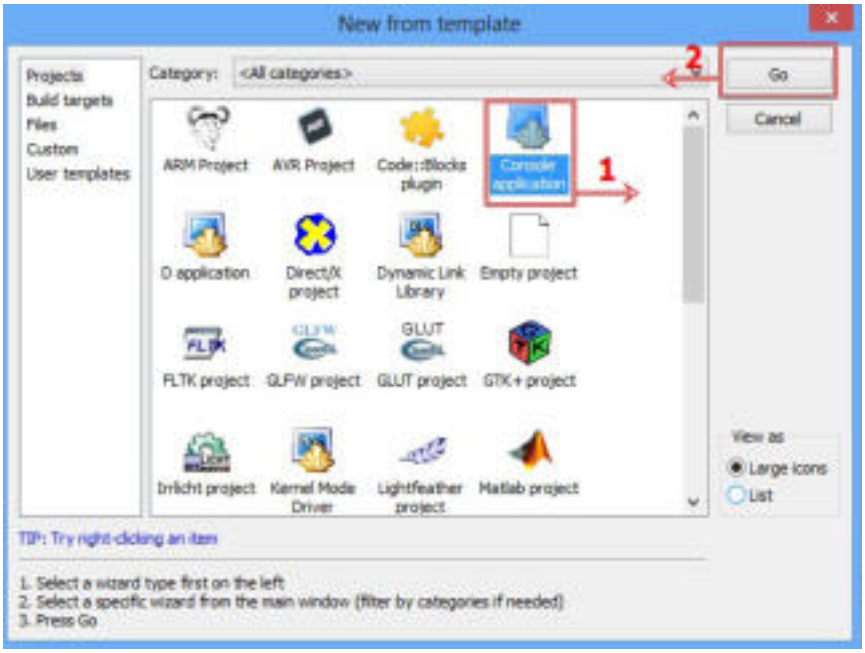
По стартувањето на програмата се појавува работниот прозорец на програмата Code::Blocks. Ајде да ја разгледаме работната околина и да ги опишеме функционалностите на алатките!

Слика 7: Работна околина на Code::Blocks

Работната околина ги опфаќа следните елементи:

1.мени-лента со наредби;
2.лента за менаџирање;
3.едитор за внесување изворен код (**Source Code**);
4.прозорец за испишување пораки за грешки.

Првиот чекор при креирањето нов проект во интегрираната околина за програмирање Code::Blocks, започнува со кликнување на менито **File →New →Project**, при што се појавува следниот прозорец:

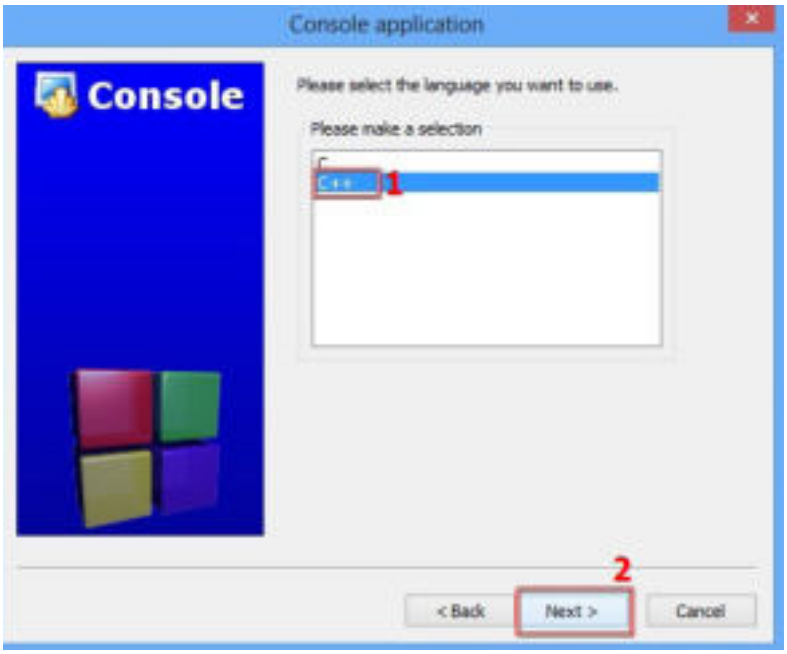


Слика 8: Креирање нов проект

Од прозорецот избираме **Console Application** и кликнуваме на копчето **Go** за да продолжиме кон вториот чекор.

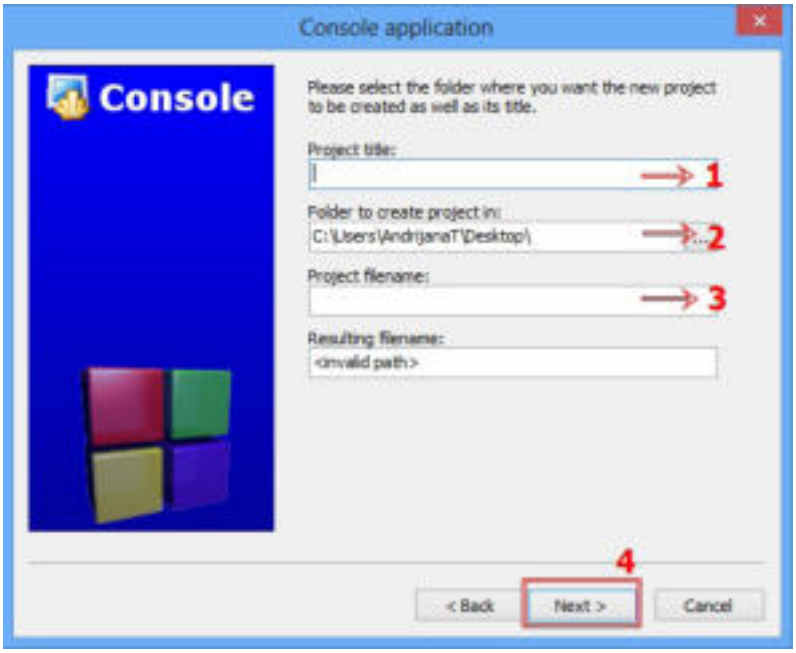
Слика 9: Прилагодување на работната околина

Со кликнување **Next** се појавува прозорец од кој избираме кој програмски јазик ќе го употребуваме, како што е прикажано на следната слика:



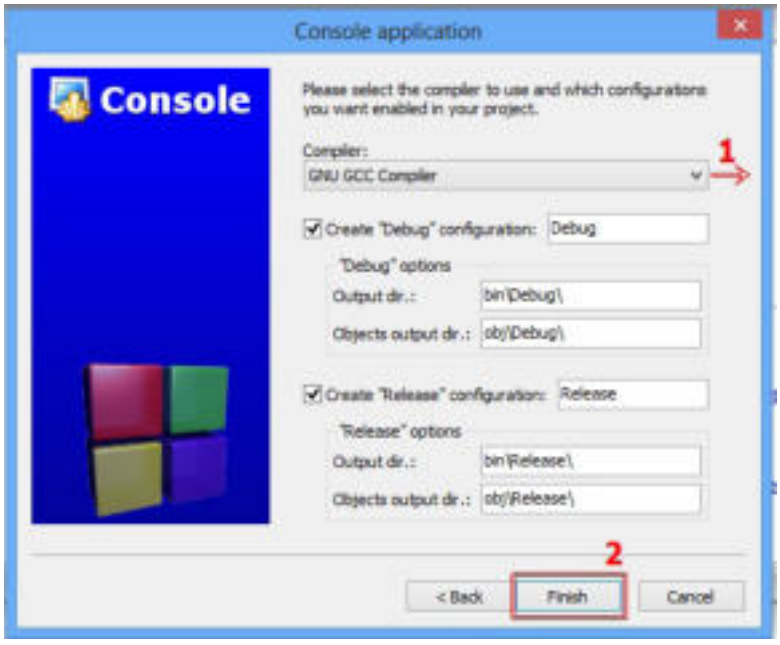
Слика 10: Избор на програмски јазик за пишување

Со кликнување на копчето **Next** пристапуваме кон доделување име на проектот и локација каде што ќе биде сместен, односно зачуван тој проект:



Слика 11: Доделување име и локација на проектот

Најпрво му доделуваме име на проектот, потоа дефинираме во кој фолдер ќе биде сместен проектот, исто така, доделуваме име на проектниот фајл кој се креира и на крајот кликнуваме на копчето **Next**. Со кликнување на копчето **Next** пристапуваме кон избор на компајлер:



Слика 12: Избор на компајлер

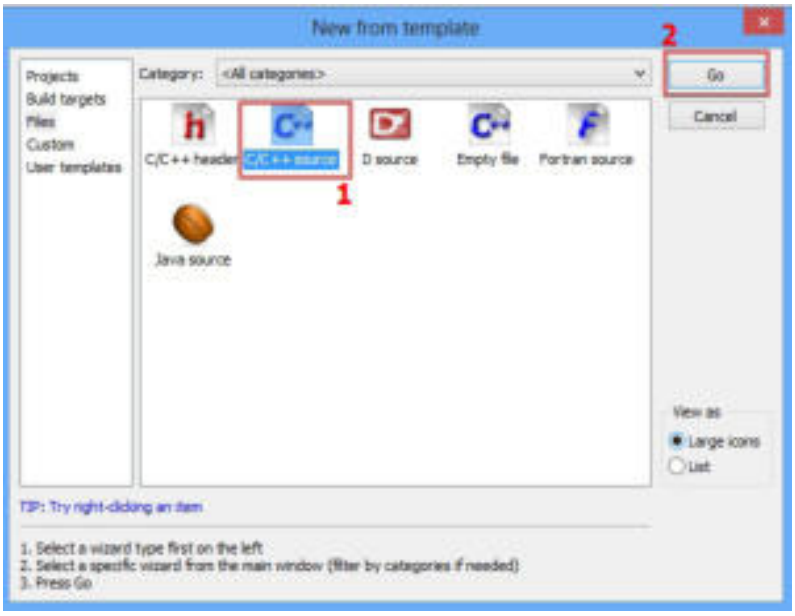
Со кликнување на копчето **Finish** започнуваме со постапката за креирање изворни кодови во едиторот на интегрираната околина за програмирање.

4.2.3 Креирање нов фајл за изворен код

Совет

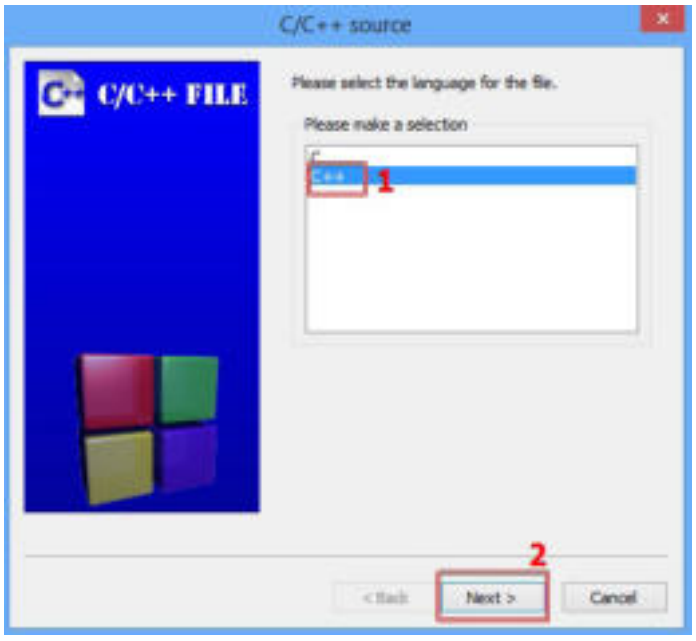
На работната површина на твојот компјутер креирај фолдер во кој ќе ги зачувуваш сите креирани проекти.

Најпрво, да креираме **нов фајл** за да го напишеме изворниот код. Нов фајл се креира преку менито **File → New → File**, при што се започнува постапката чекор, по чекор. Првиот чекор е избор на видот на фајлот:



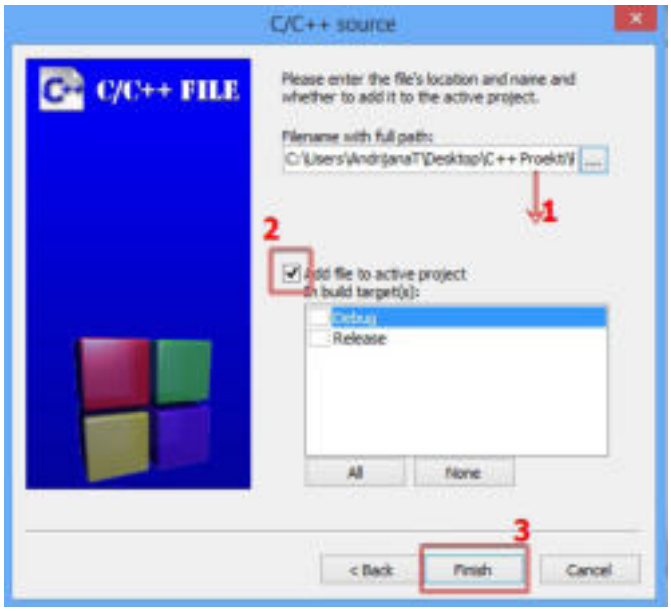
Слика 13: Избор на C++ фајл

Избираме **C/C++ Source** фајл и кликнуваме на **Go**. Потоа, пристапуваме кон вториот чекор, а тоа е избор на програмски јазик:



Слика 14: Избор на програмски јазик

Избираме **C++** програмски јазик и кликнуваме на копчето **Next** за да пристапиме кон следниот чекор од постапката:



Слика 13: Избор на C++ фајл

Во прозорецот кликнуваме на копчето Browse и преку отворениот прозорец му доделуваме име и локација на новокреираниот фајл. Со кликнување на копчето **Finish** влегуваме во едиторот на изворен код (**Source Code**) и започнуваме со пишување на програмата.

Запомни!

Интегрирана околина за програмирање е софтверски пакет кој ги содржи алатките кои се потребни за креирање програма. Такви алатки се: едитор на изворен код, библиотеки за кодови, компајлери и платформи за тестирање. Code::Blocks претставува бесплатна интегрирана околина за програмирање со отворен код.

Прашања

1.Што претставува интегрирана околина за програмирање?
2.Која е главната намена на интегрираната околина за програмирање?
3.Кои алатки ги вклучува интегрираната околина за програмирање?
4.Во кој дел од прозорецот се пишува изворен код?
5.Која е улогата на компајлерот во работната околина за програмирање?
6.Која е улогата на дебагерот?
7.Опиши го работниот прозорец на интегрираната околина за програмирање Code::Blocks!

4.3 Изглед на готови пример-програмски кодови

За да започнеме со пишување изворен код на програма треба да ја знаеме **структурата** и **синтаксата** на програмскиот јазик. Бидејќи, програмскиот јазик претставува множество од правила, симболи и специјални знаци кои се употребуваат при креирање на програмата, постојат правила за **синтакса (граматика)** и **семантика (логика)** кои мора да се почитуваат со единствена цел создавање програма која непречено и точно ќе се извршува.

Изворниот код се пишува во уредувачи за изворен код, односно едитори. На сликата е даден пример за изворен код:

prva.cpp

```
1   // Креирање едноставна програма
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7   cout << "Programiranje vo C++" << endl;
8   return 0;
9   }
10
11
```

Слика 1: Пример на изворен код

Од едиторот на изворниот код забележуваме дека секоја линија е означена со број. Покрај броевите има зелена линија која ни покажува дека овој код е преведен. Претходно, пред да се изврши преведувањето таа линија е жолта. Ајде да го објасниме значењето на кодот во секоја линија!

Број на линија	Значење на изворен код
Л1	Со знакот // се започнува пишување коментар или се дава опис на она што следува
Л2	Наредба за вклучување на библиотеката **iostream**
Л3	**using namespace std** покажува дека ќе се користат стандардните елементи од библиотеките
Л4	Празна линија која се игнорира при извршувањето
Л5	Главната функција **main()**
Л6	Отворена голема заграда која означува почеток и затворена голема заграда која означува крај на извршувањето на наредбите од главната функција **main()**
Л7	Cout е наредба за печатење на екран << се оператори за печатење **„Programiranje vo C++"** е текст за печатење на екран **endl** е наредба за крај на линијата „ ; " означува крај на наредбата
Л8	**return 0**; е порака до оперативниот систем дека програмата е успешно извршена
Л9	Затворање на голема заграда значи затворање на главната функција **main()**

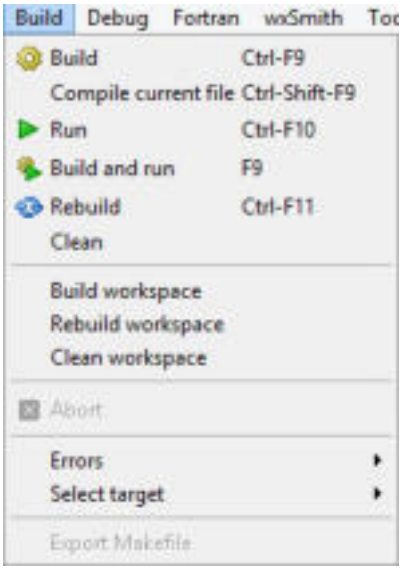
Слика 1: Пример на изворен код

Совет

По секој исказ се става точка и запирка (;) и тоа претставува знак кој ни укажува дека исказот тука завршува, освен во други случаи кои ќе бидат прикажани при креирање на програмите.

При пишување на изворен код, програмерите внимаваат на прегледноста на напишаните наредби со цел да не придонесат кон хаотичност и конфузија при анализирање на изворниот код. На пример, за преведувачот дали ќе се напишат наредбите во една линија или ќе се организираат во посебни линии, не значи ништо, бидејќи знакот точка и запирка (;) јасно укажува дека тука е крајот на наредбата.

Преведувањето и поврзувањето во извршен код се одвиваат како една фаза, а се одвиваат преку менито **Build → Build** или преку комбинација на копчињата од тастатура **Ctrl + F9**:



Слика 2: Наредбата Build

Доколку при оваа фаза се појават грешки, програмерот пристапува кон нивно решавање и повторно врши преведување и поврзување на изворниот код. Пораката дека е пронајдена грешка се прикажува во долниот дел на работниот прозорец во посебна рамка **Build log:**

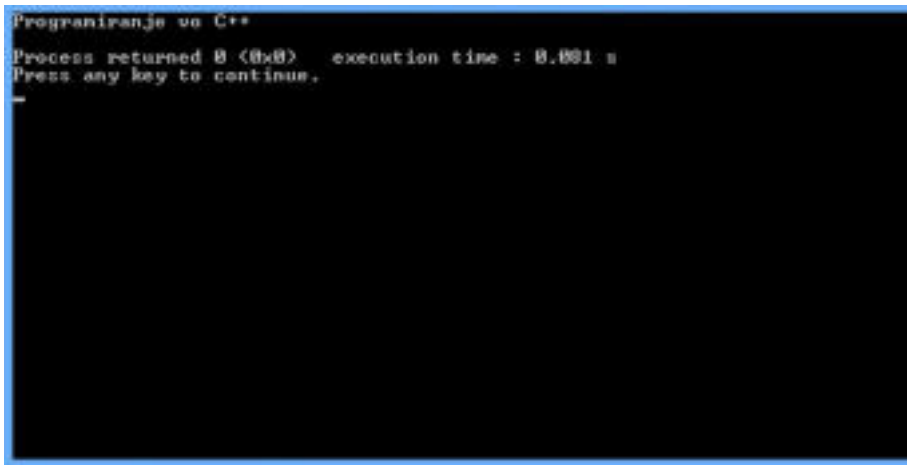
Слика 3: Build log рамка

Програмата ќе ја извршиме со наредбата **Build → Run** или со комбинација на копчињата од тастатура **Ctrl + F10**. По овој процес во креираниот фолдер има три датотеки, односно фајлови: изворна, објектна и извршна датотека.

Повикувањето на наредбата **Build** и наредбата **Run** може да биде и преку лентата со алатки која се наоѓа веднаш под мени-лентата:

Слика 4: Икони на алатките за Build и Run

Со кликнување **Run**, започнува извршување на програмот кој се прикажува и извршува во посебен прозорец, како што е прикажано на сликата:



Слика 5: Прозорец на извршен фајл

Од овој прозорец забележуваме дека пораката е испишана на почетокот на прозорецот, а под неа самиот компјутер генерирал порака „**Press any key to continue**...“, што значи, ако кликнеме на некое копче од тастатурата овој прозорец на програмата ќе се затвори.

Запомни!

При пишување програма секогаш треба да внимаваме на **синтаксата, семантиката и структурата** на програмскиот јазик. Изворниот код се пишува во едитор, во кој секоја линија е означена со број. Изворниот код се преведува и поврзува и тоа се одвива како една фаза. Доколку се појават грешки се појавуваат пораки во **Build log**. Извршувањето на програмата се прикажува на посебен прозорец и како резултат се појавуваат три фајлови: изворен, објектен и извршен.

Прашања

1.Што е изворен код?
2.Со која наредба се преведува изворен фајл?
3.Колку фајлови се креираат по преведувањето на изворниот фајл? Кои се тие?
4.Како се извршува програма?

4.4Извршување на готови пример-програми

Од претходната наставна единица го осознавме значењето и функционалноста на основните и стандардни елементи во едноставната програма која единствено извршува испишување порака на екран. За да запознаеме и други елементи ќе креираме програма во која ќе имаме можност интерактивно да постапуваме при извршување на програмата, односно да внесуваме податоци или вредности од тастатурата. Во едиторот на изворниот код, во интегрираната околина за програмирање **Code::Blocks**, да ги напишеме следните низи од наредби:

```
1   // Presmetuvanje na zbir na dva broja
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int br1, br2, zbir;
8       cout << "Da go presmetame zbirot na dvata broja!" << endl;
9       cout << "Vnesi go prviot broj: " << endl;
10      cin >> br1;
11      cout << "Vnesi go vtoriot broj: " << endl;
12      cin >> br2;
13      zbir=br1+br2;
14      cout << "Zbirot na dvata broja e " << zbir << endl;
15      return 0;
16  }
```

Слика 1: Изворен код на програма за пресметување на збир

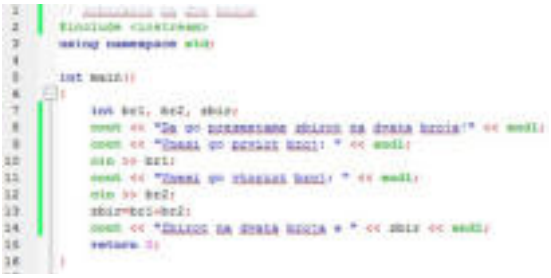
Во првата линија на програмата напишавме **коментар**, односно **објаснување** за тоа каква програма ќе креираме и што ќе пресметува таа програма. Во нашиов случај ќе пресметаме збир на два броја.

Секоја програма започнува со вовед во кој дефинираме кои **библиотеки** ќе ги вклучиме и кои наредби ќе ги употребиме:

```
2   #include <iostream>
3   using namespace std;
```

Слика 2: Вовед во програмата

Во овој случај вклучена е библиотеката **iostream**, а тоа значи дека ќе се применуваат стандардните наредби за внесување податоци и приказ на резултатот на екран. Следува главната функција во која се извршуваат наредбите кои всушност ја сочинуваат програмата:



Слика 3: Главната функција Main

Бидејќи, во оваа програма ќе внесуваме податоци од тастатура, најпрво во главниот програм ги дефинираме варијаблите, односно променливите чии вредности зависат од внесот од тастатура. Со **int** означуваме дека вредностите кои ќе ги внесеме од тастатура ќе бидат **цели броеви (integer)**. На варијаблите им дадовме имиња: **br1, br2 и zbir.**

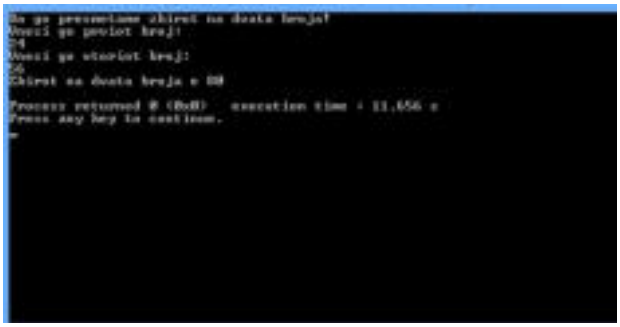
Во линија број осум (8) е испишана наредбата **cout** со која се извршува пишување порака на екран. Зад наредбата се пишуваат операторите за печатење (<<), а потоа пораката која треба да се прикаже на екран запишана во горни наводници. Со наредбата **endl** означуваме крај на линијата, а со знакот точка и запирка (;) крај на исказот, односно наредбата. Иста функционалност и значење има и линија број девет (9).

Во линија број десет (10), испишана е наредбата cin, која проследена со операторите (>>) „>>“ означува наредба за внесување податоци од тастатурата. Секако, наредбата завршува со знакот точка и запирка (;). Истата постапка е повторена во линиите број единаесет (11) и број дванаесет (12).

Во линија број тринаесет (13), напишана е формулата за пресметка на збирот на двата внесени броја или во буквален превод варијаблата zbir ќе добие вредност од збирот на **br1** и **br2**. Но, оваа линија не се прикажува на екран, таа само се преведува. Вредноста на **zbir** ќе се прикаже во следниот ред од низата наредби.

Така, во линија четиринаесет (14) со **cout** и операторите за печатење (<<) ќе се испише реченицата која е напишана во наводниците, а зад неа вредноста на варијаблата **zbir**. Тука со endl завршува наредбата. Главната функција ја завршуваме со **return 0;** со што му укажуваме на процесорот дека програмата е успешно извршена. Со затворање на големата заграда означуваме крај на наредбите во главната функција **main**.

По испишувањето на изворниот код во едиторот на интегрираната околина за програмирање, пристапуваме кон компајлирање и дебагирање, т.е. со преведување и поврзување на изворниот фајл во извршен фајл. Преку менито **Build** и наредбата **Build** ја започнуваме фазата за преведување. Доколку не се појават грешки во **Build log** продолжуваме со наредбата **Run** за извршување на фајлот. Во спортивно ги коригираме грешките и го повторуваме процесот. Извршниот фајл се извршува во следниот прозорец:



Слика 4: Извршување на програмата за пресметка на збир

На овој екран всушност го гледаме резултатот од нашата работа и дали процесот се одвивал според дефинираните чекори.

Но, понекогаш не оди сè според планираното. Можно е да се случуваат грешки при преведувањето за кои програмерот мора да најде решение и процесот да продолжи понатаму. Компјутерските грешки се нарекуваат **багови (Bugs)**, па оттаму од таму доаѓа и името на процесот на пронаоѓање и поправање грешки – **дебагирање (Debug)**. Овој процес практично ќе го извежбаме преку креирање изворен код.

Во едиторот за изворен код во **Code::Block** да ги напишеме следните низи од наредби:

```
1   // пресметка на два броја
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int br1, br2, proizvod;
8       cout << "Da go presmetame proizvodot na dvata broja!" << endl;
9       cout << "Vnesi go prviot broj: " << endl
10      cin >> br1;
11      cout << "Vnesi go vtoriot broj: " << endl;
12      cin >> br2
13      proizvod=br1*br2;
14      cout << "Proizvodot na dvata broja e " << proizvod << endl;
15
16      system ("PAUSE");
17      return 0;
18  }
```

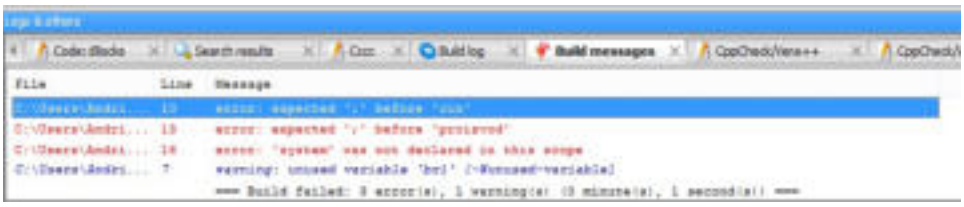
Слика 5: Изворен код за пресметка на производ

Да се обидеме да ги воочиме грешките кои сме ги направиле при креирање на програмата. Ајде да ги наведеме!

При процесот на преведување на изворниот фајл ќе го добиеме следниот резултат:

Слика 6: Порака за појавени грешки

Во полето **Build log** забележуваме дека постојат грешки при преведување на програмата. Со кликнување на опцијата **Build message** ќе ги видиме грешките една, по една:

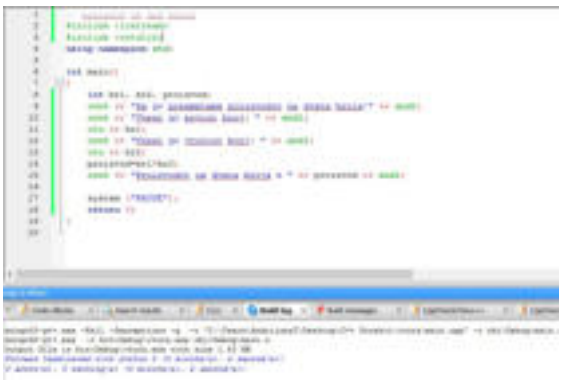


Слика 7: Издвоени грешки по линии

Од прозорецот забележуваме дека имаме грешки:

- во линија број десет (10) ни недостига знакот „;“ пред повикување на наредбата cin;
- во линија број тринаесет (13) повторно недостига знакот „;“ веднаш пред дефинирањето на вредноста на „**Proizvod**“;
- на крајот не ја препознава наредбата **system (“PAUSE”)**, бидејќи истата не припаѓа во библиотеката **iostream**.

Пристапуваме кон решавање на грешките, односно го додаваме знакот „;“ на местата на кои недостига и во воведниот дел ја додаваме библиотеката **cstdlib** со цел да се овозможи функционирањето на наредбата system **(“PAUSE”)**.



Слика 8: Изворен код по направени корекции

Сега веќе штом нема грешки пристапуваме кон извршување, при што на посебен екран ќе се прикаже извршниот фајл и резултатот од програмата.

Запомни!

Претпроцесорските наредби започнуваат со знакот „#“. Преку #Include ги најавуваме библиотеките чии наредби ќе ги употребуваме во изворниот код. Main е главната функција која ја има секоја C++ програма и таа започнува со отворена голема заграда „{“ и завршува со затворена голема заграда „}“. Помеѓу двете загради се дефинираат варијаблите, се пишуваат искази како дел од чекорите на програмата, знаци, симболи и сл. При преведување и поврзување можат да се појават грешки кои се викаат Bugs, а процесот на откривање и поправање на грешките – дебагирање (Debug).

Прашања

1.Како започнува секоја програма во C++ програмскиот јазик?
2.Дефинирај вовед во програма!
3.Како се нарекуваат линиите кои започнуваат со две коси црти (//)?
4.Наведи која е стандардна библиотека!
5.Како се вика главната функција? Што содржи таа?
6.Како се викаат грешките во програмата? Како се отстрануваат?
7.Што значи return 0;?
8.Со кој знак се одвојуваат наредбите една од друга?

4.5 Основни елементи на програмскиот јазик C++

Да се потсетиме!

Со нашето секојдневно комуницирање ние применуваме некој јазик, односно разменуваме информации, податоци, идеи, ставови. Како меѓусебно ќе се разбереме? Кој јазик го употребуваме? Од што е составен јазикот на кој зборуваме? Како комбинираме букви за да конверзираме?

Овие прашања се наметнуваат и при вршење на едноставни работи со компјутерот. И компјутерите комуницираат: корисник – компјутер – корисник или, пак, компјутер – компјутер. Корисникот комуницира со компјутерот преку примена на влезните уреди преку кои тој внесува податоци и информации, дава наредби, употребува програми и сл. Овие активности се преведуваат во **бинарен јазик** кој е разбирлив за компјутерот, и обратно, кога компјутерот треба да даде одговор на некаква активност тогаш бинарните изрази се преведуваат на јазик разбирлив за корисникот и така резултатот го гледаме на екран.

За комуникацијата помеѓу компјутерите можеме да кажеме дека се случува кога тие се мрежно поврзани и можат да разменуваат податоци, информации, документи, програми, заеднички уреди и сл.

Програмските јазици, исто како и природните јазици, имаат своја азбука. **Азбуката на програмските јазици се состои од**:

- големите и малите букви на абецедата;
- цифрите од нула (0) до девет (9);
- специјалните знаци;
- знакот за празно место;
- комбинации од два или три знаци.

Соодветната комбинација и примена на овие елементи креираат **искази**, односно **наредби** и како такви се нарекуваат **градбени елементи на програмскиот јазик**. Според тоа, градбените елементи на програмскиот јазик се:

- резервирани зборови;
- идентификатори;
- оператори;
- интерпункциски знаци;
- коментари.

Секој од наведените градбени елементи има свое место во структурата и синтаксата на програмскиот јазик. Ајде описно да ги објасниме!Резервираните зборови се викаат уште и клучни зборови. Тие се однапред дефинирани и при преведување на изворниот фајл, преведувачот точно знае што значат. Такви се на пример: int, delete, if,

then, else, true, false, while и др.

Идентификаторите во програмскиот јазик ги дефинира програмерот. Тие се комбинации од знаци кои можат да се комбинираат почитувајќи ги следните правила:

- името започнува со буква или долна црта;
- малите и големите букви се разликуваат, на пример: „Zbir“ е различно од „zbir“;
- името може да содржи цифра, но не и да започнува со цифра;
- името не може да биде резервиран збор;
- името не може да содржи специјален знак како !, @, # и сл.

Операторите во програмските јазици се употребуваат за означување аритметички, логички и други операции кои се изведуваат во програмата. Интерпункциските знаци се користат за раздвојување на елементите на програмскиот јазик, како на пример со знакот „;“ се означува крај на исказот, односно наредбата, т.е. се разделуваат исказите.

Програмерот при креирање на програмата може да напише **коментар** или **повеќе коментари**. Пишувањето коментар започнува со **две коси црти „//“**. Всушност, со коментарите се опишува за што е наменета програмата или, пак, се објаснуваат поединечни линии со искази, односно наредби. Коментарите не се извршуваат тие само стојат како насока или објаснување за значењето на елементот, линијата со наредба или програмата во целина.

Различните програмски јазици меѓусебе се разликуваат најчесто по синтаксата, односно различни резервирани зборови, правила за нивна примена, и сл.

Бидејќи ние работиме со C++ програмскиот јазик негови градбени елементи се:

1.**податоци** - со програмските јазици можат да се обработуваат различни типови податоци и сите податоци ги имаат следните карактеристики: име, тип и вредност. Примери за различни типови податоци: **char, int, bool, float, double и сл**.;

2.**константи** претставуваат величини кои не ја менуваат својата вредност;

3.**променливи** се величини кои ја менуваат својата вредност;

4.**декларациите** всушност даваат опис на променливата, односно каков тип на податок ќе содржи променливата, на пример: i**nt x, float a,** и сл.;

5.**изрази** за вршење на некаква пресметка зависно од задачата

која треба да се реши. Пример: sum + i и сл.;

6.**искази** претставуваат наредби кои треба да бидат извршени во програмата, како на пример: **sum=sum+i, while, for, if –else, switch, break** и др.;

3.**функции** - во **C++** главна функција е **main „( )“**. Исказите, односно наредбите се пишуваат помеѓу двете големи загради на главната функција, при што секој исказ завршува со знакот „;“. Постојат функции кои можат да бидат креирани од програмерот, но најчесто се употребуваат готовите постоечки функции;

4.**модули** - при пишување на програмскиот изворен код секој елемент се прикажува во различна боја. Преку бојата утврдуваме дали точно сме го испишале зборот или исказот.

Начинот на кој се комбинираат градбените елементи на програмскиот јазик, нивниот тек и редослед ја дефинираат **структурата на програмскиот јазик**. На пример, структурата на една програма изгледа вака:

Слика 1: Структура на C++

Запомни!

Градбените елементи на програмските јазици се: резервирани зборови, идентификатори, оператори, интерпункциски знаци и коментари. Програмскиот јазик C++ се карактеризира со следните основни елементи: податоци, константи, променливи, декларации, изрази, искази, функции и модули

Прашања

1.Набројги градбените елементи на програмскиот јазик C++!
2.Набројги основните елементи на програмскиот јазик C++!
3.Што се резервирани зборови? Набројнекои од нив!
4.Што се идентификатори? Како се креираат?
5.Наведи неколку примери на точно креирани идентификатори според правилата за нивно креирање!
6.Што се функции? Која функција мора да ја има програмата?
7.Што се коментари? Наведи пример за коментар!

4.6 Искази

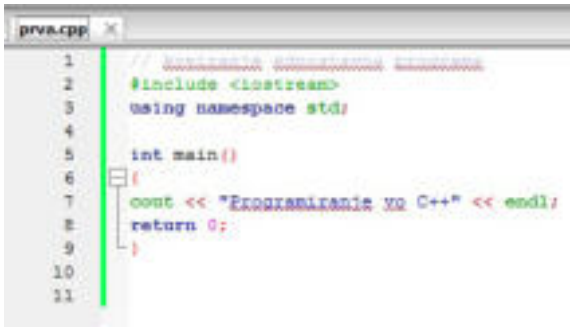
Исказите претставуваат основен елемент на секој програмски јазик. Без исказите компјутерот не би знаел кои чекори треба да се извршат за да се дојде до конечниот резултат. Според тоа, **исказите можеме да ги дефинираме како наредби кои му кажуваат на компјутерот што да извршува, односно кои дејствија да ги преземе за да дојде до решението.**

Исказите во програмата се одвојуваат со знакот ; “ Знакот ;“ ни кажува дека тука е крајот на наведениот исказ.

Во програмските јазици постојат многу искази и сите имаат различно значење, функција и цел. При програмирањето ние најчесто се среќаваме со исказите за приказ на екран и исказите за доделување вредности.

4.6.1 Исказ за приказ на екран

При разгледувањето на готови програми, првата програма со која се соочивме беше едноставна програма која испишува текст или порака на екранот:



Слика 1: Исказ за приказ на екран

По преведувањето и поврзувањето на овој изворен фајл на извршниот прозорец се прикажува пораката „Programiranje vo C++“. Според примерот:

```
cout << "Programiranje vo C++" << endl;
```

исказот претставува исказ кој овозможува приказ на екранот. Ајде да ги разгледаме елементите кои ги содржи овој исказ:

Елемент	Значење
cout	Израз за приказ на екран
<<	Оператор за испишување пораката на екран
“”	Сè што е напишано помеѓу наводниците се прикажува на екран
Programiranje vo C++	Пораката помеѓу двете наводници која ќе се испише на екран
endl	Завршување на линијата
;	Завршување на исказот

Табела 1: Елементи на исказ за приказ на екран

Изразот **cout** можеме да го употребуваме и на други начини. Ајде да видиме што ќе прикаже на екран исказот **cout**, доколку е претставен на следните начини:

Исказ cout	Приказ на екран
cout<<"Brojot a";	На екранот ќе се појави пораката која се наоѓа помеѓу наводниците
cout<<br;	На екран ќе се прикаже вредноста на променливата br
cout<<55;	На екран ќе се испечати бројот 55
cout<<3*br;	На екран ќе се прикаже резултатот од пресметката

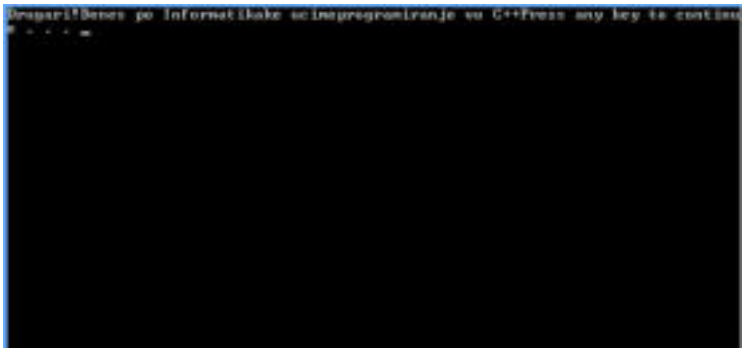
Табела 2: Различни начини на употреба на исказот cout

Преку интегрираната околина за програмирање **Code::Blocks** да креираме програма која ќе ни прикажува на екран порака, како во следниот пример изворен код:

```
1   // programa za prikaz na ekran
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5   int main()
6   {
7   cout << "Drugari!";
8   cout << "Denes po Informatika";
9   cout << "ke ucime";
10  cout <<"programiranje vo C++";
11  system ("pause");
12  return 0;
13  }
14
```

Слика 2: Изворен код за приказ на текст на екран

Резултатот на оваа програма е прикажан на следната слика:



Слика 3: Испишана порака на екран

Од сликата, забележуваме дека пораката е испишана во еден ред, сите делови се споени и вака прикажана не покажува прегледност и јасност. За да ги одвоиме во посебни линии ќе ја употребиме наредбата **endl**:

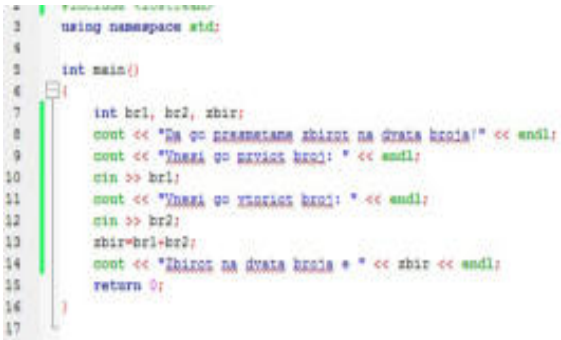
Во изворниот код на примерот, забележуваме дека има повеќе искази за печатење и тие ќе се извршуваат една, по една, како што се наведени. Вака напишаните искази се викаат секвенца или низа од искази.

Слика 4: Примена на наредбата endl

4.6.2 Исказ за доделување вредности

Исказот за доделување вредности го применивме при пресметката на збирот на двата внесени броеви. Доделувањето на вредноста на променливата може да биде со внес од тастатура или, пак, вредност која се добива при пресметка.

Ајде да го разгледаме изворниот код на програмата за пресметка на збир на два броја!



Од сликата, забележуваме дека програмата започнува со коментар како опис на програмата. Под коментарот следуваат предпроцесорските настани и декларирањето дека ќе се употребуваат стандардни искази кои се наоѓаат во **namespace**. Потоа, започнува главната функција **main ( ).**

Слика 5: Доделување вредност на променлива

Во главната функција е прикажана секвенца од искази кои редоследно ќе се извршуваат. Покрај декларирањето на променливите и исказот за приказ на екран, тука е применет и исказот за доделување вредност. Постапката за примена на исказот **cin**, како исказ кој доделува вредност на променливата со внес од тастатура е прикажан на следната слика:

```
cout << "Vnesi go prviot broj: " << endl;
cin >> br1;
```

Слика 6: Доделување вредност од татстаура

Споредтоа,**cin**еисказпрекукојќедоделимевредностнапроменлива преку тастатура, односно ќе внесеме податок преку тастатура. Потоа, следуваат **операторите за внес „>>“** и променливата која е дефинирана за првиот број, односно **br1**.

Променливата **br1** ќе има вредност која е внесена од страна на корисникот од тастатурата. Но, најпрво, пред да се примени овој израз мора да се дефинира променливата **br1**. На претходната слика, променливите **br1, br2 и zbir** се дефинирани како **integer** вредности, односно цели броеви. Според тоа, за да се доделат вредностите на **br1** и **br2** од тастатура, корисникот ќе внесе цели броеви. Тоа, претставува **техника за внесување податоци**. На променливата **zbir** нема да и се додели вредност од тастатурата, туку нејзината вредност ќе биде резултатот од пресметката на збирот на двата броја, како што е прикажано на следната слика:

```
zbir=br1+br2;
cout << "Zbirot na dvata broja e " << zbir << endl;
```

Слика 7: Доделување вредност на променлива како резултат на пресметка

Од приказот, утврдуваме дека вредноста на променливата **zbir** зависи од вредностите на **br1** и **br2** кои корисникот ќе ги внесе од тастатура. Збирот на двата броја ја определува вредноста на варијаблата **zbi**r. На крајот, резултатот се прикажува на екран со исказот **cout**.

Запомни!

Исказите се наредби кои одредуваат кои дејствија треба да ги преземе компјутерот за да се дојде до решение. Знакот точка и запирка (;), после секој исказ означува крај на исказот. Исказот cout се користи за приказ на екран. Тој е проследен со операторите (<<), наводници за испишување на пораката и завршува со точка и запирка. Исказ за доделување вредност на променлива од тастатура, односно за внесување податок е cin. По исказот cin следат операторите (>>) за внес на податоците, променливата и завршува со точка и запирка. Овие видови програми се нарекуваат интерактивни програми. Но, покрај внес од тастатура, променливата може да добие вредност како резултат на пресметка.

Прашања

1.Што е исказ?
2.Кој исказ се користи за приказ на екран?
3.Што овозможува наредбата endl?
4.Кој е исказот за доделување вредност?
5.Како може да се определи вредноста на тастатурата?

4.7.Изработка на програми

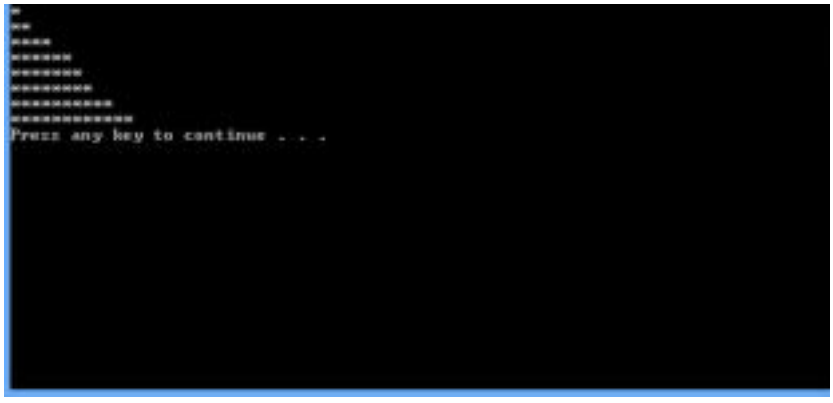
Во следните задачи ќе изработиме програми каде повеќе искази за приказ на екран се **напластени**. Напишаните искази во програмскиот код ќе се извршуваат еден, по еден редоследно и овој начин на извршување се вика **техника на редоследно извршување**.

На пример, со помош на исказот за приказ на екран можеме да отпечатиме **форма на правоаголен триаголник**. Тука ќе ја користиме техниката на редоследно извршување. За таа цел, креираме нов проект во **Code::Blocks** и започнуваме со пишување на програмата:

```
1   //напластени искази за приказ на екран
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5
6   int main()
7   {
8       cout << "*" << endl;
9       cout << "**" << endl;
10      cout << "*****" <<endl;
11      cout << "*******" << endl;
12      cout << "********" << endl;
13      cout << "**********" << endl;
14      cout << "***********" << endl;
15      cout << "*************" << endl;
16      system ("pause");
17      return 0;
18  }
```

Слика 2: Приказ на напластени искази за приказ на екран

Резултатот од следниот изворен код е прикажан на следната слика:



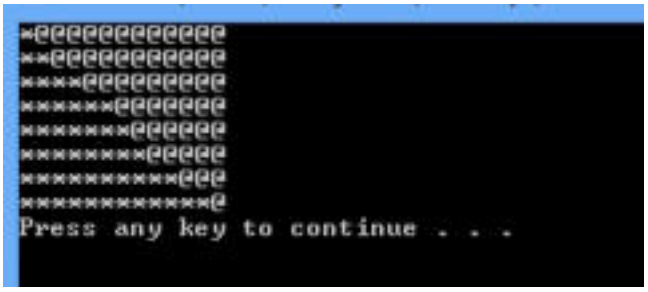
Слика 2: Приказ на напластени искази за приказ на екран

Да се обидеме на истиот начин да креираме квадрат и при приказот на квадратот да биде воочлива дијагоналата. За таа цел, покрај веќе употребените ѕвездички, ќе употребиме друг знак. Изворниот код на оваа програма би изгледал вака:

```
1  //наредбени искази за приказ на екран - квадрат
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
8      cout << "*@@@@@@@@@@@@" << endl;
9      cout << "**@@@@@@@@@@@" << endl;
10     cout << "****@@@@@@@@@" <<endl;
11     cout << "******@@@@@@@" << endl;
12     cout << "*******@@@@@@" << endl;
13     cout << "********@@@@@" << endl;
14     cout << "**********@@@" << endl;
15     cout << "************@" << endl;
16     system ("pause");
17     return 0;
18 }
19
```

Слика 3: Изворен код за приказ на квадрат

Резултатот од извршувањето на програмата е прикажан на посебен екран, како што е претставено на сликата:



Слика 4: Приказ на извршен фајл за приказ на квадрат

На истиот принцип, ајде да се обидеме да бидеме креативни и да исцртаме објекти со помош на исказите за приказ на екран. На пример, можете да ја исцртате првата буква на вашето име, објект срце, и сл.

Запомни!

Пишување искази во програмски код, кои се извршуваат сите и редоследно се вика техника на редоследно извршување.

4.8 Аритметички операции и изрази

При креирањето програми во програмскиот јазик C++ најчесто употребувани се аритметичките операции и изрази. Аритметичките операции и изрази во суштина ги содржат математичките пресметки. Според тоа, **аритметичките изрази во програмскиот јазик C++ можат да се дефинираат како запис од две или повеќе бројни вредности кои се поврзани со математички оператори**.

Математичките оператори се симболи кои претставуваат специфична акција. Основните аритметички оператори кои се користат во C++ се:

Аритметички оператор	Значење
+	Собирање
-	Одземање
*	Множење
/	Делење
%	Остаток од делење

Табела 1: Аритметички операции во C++

Забелешка!

Во програмскиот јазик C++ не постои оператор за експонет. Но, постои вградена функција pow која е дефинирана во библиотеката cmath.

За да видиме како се применуваат аритметичките оператори ќе креираме едноставна програма која ќе врши едноставни математички операции.

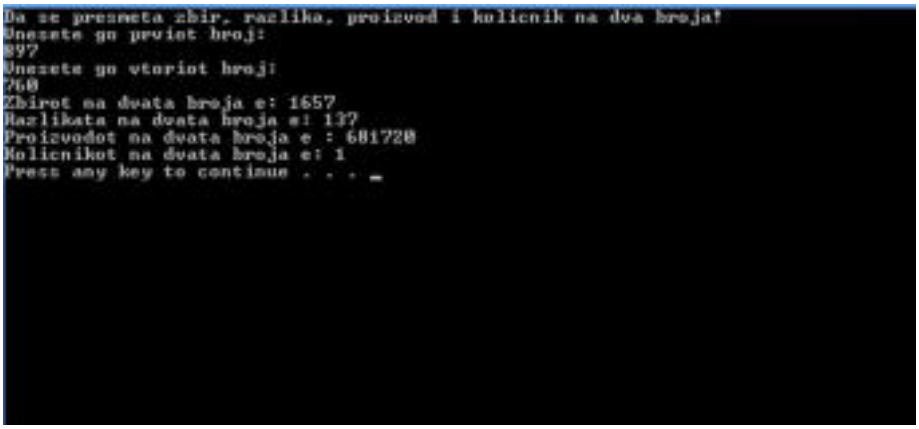
```
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int broj1, broj2, zbir, razlika, proizvod, kolicnik;
    cout << "Da se presmeta zbir, razlika, proizvod i kolicnik na dva broja!" << endl;
    cout << "Vnesete go prviot broj: " << endl;
    cin >> broj1;
    cout << "Vnesete go vtoriot broj: " << endl;
    cin >> broj2;
    zbir=broj1+broj2;
    cout << "Zbirot na dvata broja e: " << zbir << endl;
    razlika=broj1-broj2;
    cout << "Razlikata na dvata broja e: " << razlika <<endl;
    proizvod=broj1*broj2;
    cout << "Proizvodot na dvata broja e : " << proizvod << endl;
    kolicnik=broj1/broj2;
    cout << "Kolicnikot na dvata broja e: " << kolicnik << endl;
    system ("pause");
    return 0;
}
```

Слика 1: Изворен код со примена на аритметички операции

Воведниот дел на програмата, како и во други примери на програми ги вклучува библиотеките и командите кои се дефинирани во **namespace**. **Главната функција main ( )** е всушност главната програма која ги содржи исказите кои ќе се извршуваат за да се дојде до решението.

Како што забележуваме, во главната програма се дефинирани варијаблите или променливите на кои ќе им бидат доделени вредности. Вредностите кои ќе им бидат доделени се **integer**, односно цели броеви. Варијаблите **br1** и **br2** ќе имаат вредност која корисникот ја доделува преку внес од тастатура, додека, пак, варијаблите **zbir, razlika, proizvod и kolicnik** ќе добијат вредност како резултат на пресметката. При пресметките се употребуваат основните аритметички операции. Резултатот од пресметката го гледаме на извршниот прозорец:



Слика 2: Аритметички операции со два броја

Запомни!

Аритметички операции се записи од две или повеќе бројни вредности кои се поврзуваат со математички оператори со цел да се изврши пресметка. Во програмскиот јазик C++ се употребуваат следните аритметички операции: (+) за собирање, (-) за одземање, (*) за множење и (/) за делење. Со комбинација на променливите и операторите се креираат изрази.

4.9.Константи и променливи

Константите и променливите се елементи на програмските јазици. Употребата на константите и променливите при креирање програма бараат почитување на правила за нивна правилна примена. Ајде да се запознаеме со нив!

4.9.1 Константи

Константите претставуваат податоци чија вредност за време на извршувањето на програмата не се менува. Константата во C++ програмскиот јазик се креира на два начина. Едниот начин е со користење на **#define** наредбата, а вториот начин е со додавање на зборот **const** пред типот и името на константата. На пример:

```
1   // definirawe konstanta vo C++
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5   #define pi 3.141
6   int main()
7   {
8       const int n=5;
9       cout << n*pi << endl;
10      system ("pause");
11      return 0;
12  }
13
```

Слика 1: Дефинирање константи

Првата константа која ја дефиниравме во програмата е константата **pi** која има вредност **3.141**. Неа ја дефиниравме со користење на наредбата **#define** и истата е сместена пред да се најави почетокот на главната функција **main ( )**. Бидејќи, оваа наредба е сместена пред почетокот на функцијата **main ( )**, се смета дека се извршува во предпроцесорската фаза и токму поради тоа на крајот не содржи знак точка и запирка, кој всушност го означува крајот на исказот.

Втората константа е **n**. Таа е дефинирана во главната функција **main ( )**. Имено, покрај името n и типот **int** се употребува и зборот **const**, односно **const int n=5**, како што е прикажано во програмата. Во некои случаи, доколку не се наведе типот на константата се подразбира дека е од типот **int**. Резултатот на пресметката всушност е **5*3.141 = 15.705**, кој е прикажан на екран преку исказот cout. Според тоа, можеме да заклучиме дека константите меѓусебе се разликуваат според типот на податокот со која е изразена нејзината вредност.

4.9.2 Променливи

При изучувањето на математиката, физиката, хемијата и други слични науки, честопати се среќаваме со многу променливи кои имаат некакво значење. На пример: периметар се обележува со **L**, плоштина се обележува со **P**, радиус со **r**, диагонала со **d**, температура со **T**, брзина со a, и сл. **Променливите се податоци чијашто вредност може да се менува за време на извршувањето на програмата**. Тие се декларираат со тип и име или, пак, со тип, име и вредност. На пример:

```
int I, j;
int m=15;
float e=2,78;
int a=5, b=10
```

4.9.2.1 Тип на променлива

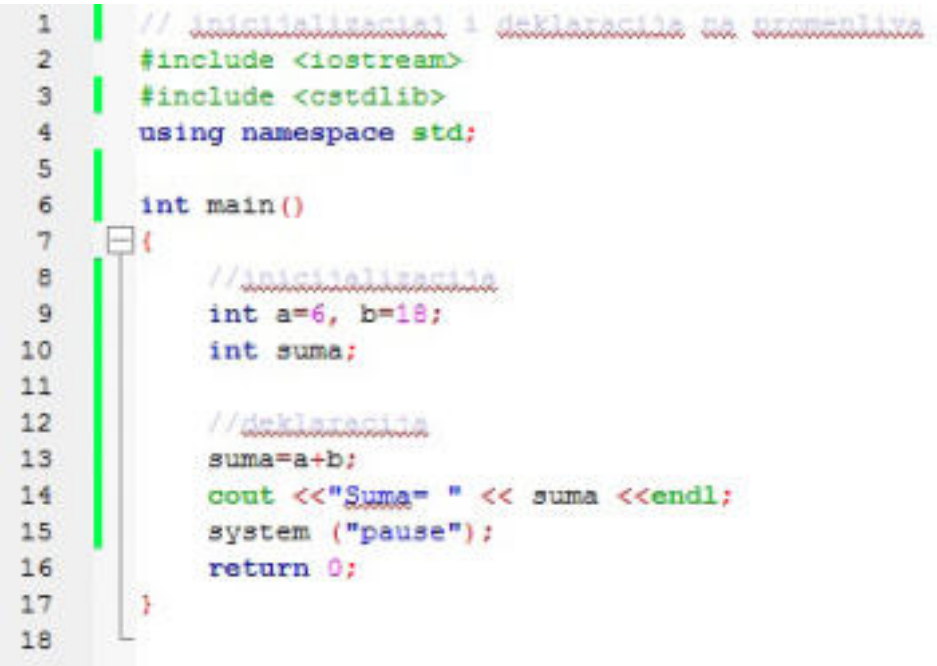
Секоја променлива која ќе се користи во програмата мора да се најави, односно да се декларира, т.е. да се дефинира нејзиниот тип. На следната табела се прикажани најчесто користените типови променливи со нивното значење:

Типови променливи	Значење
Char	Знаковна променлива која може да има вредност знак или целобројна вредност
Int	Целобројна променлива која може да има вредност ...-2, -1, 0,1,2...
Bool	Логичка променлива која може да има една од двете вредности точно (true) и неточно (false)
Float	Реална променлива со обична прецизност
Double	Реална променлива со двојна прецизност

Табела 1: Типови променливи

4.9.2.2 Додавање вредност на променлива

Покрај типот и името, на променливата може да и се додели вредност со помош на операторот „=“, како што е прикажано во следните примери: **c=8, a=10, a=b*3, x=x+6, x=d=f=2** и сл. Оваа постапка се вика **иницијализација** или **додавање почетна вредност**. Тоа, значи, дека на променливата **c** ќе и биде доделена вредност осум (8), променливата a ќе добие вредност што е резултат од пресметката на производот **b*3**, променливата **x** ќе добие резултат од пресметката на збирот на **x+6**, променливата **f** ќе добие вредност два (2), **d** ќе ја има вредноста на **f**, додека, пак, **x** ќе ја има вредноста на **d**. На пример:



```
1  // inicijalizacija i deklaracija na promenliva
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
8      //inicijalizacija
9      int a=6, b=18;
10     int suma;
11
12     //deklaracija
13     suma=a+b;
14     cout <<"Suma= " << suma <<endl;
15     system ("pause");
16     return 0;
17 }
18
```

Слика 2: Иницијализација и декларација на променливи

Во оваа програма во главната функција, во **линија број девет (9)** се дефинирани и иницијализирани променливите **a и b: int a=6, b=18**. Во **линија број десет (10)** дефинирана е променливата **suma**, но, не е иницијализирана. Таа ќе добие вредност по извршување на пресметката: **suma=a+b**, а резултатот ќе биде прикажан на екран.

Запомни!

Променливите и константите при употреба во програма мора да се најават, односно да се декларираат, а тоа значи да им се одреди типот и името. Променливите добиваат вредност со помош на операторот еднакво (=). Доделувањето на почетна вредност на променливата се вика иницијализација.

Прашања

1.Што е константа, а што променлива? Која е разликата меѓу нив?
2.Што е декларација? Дали може да се декларираат повеќе променливи со еден исказ? Пример!
3.Што е иницијализација? Наведи пример!

4.10 Искази (техники) за внесување податоци во програмата

За да ја разбереме суштината на техниката за внесување податоци во програмата ќе креираме неколку програми.

Задача

Да се креира програма за пресметка на квадрат на внесен број!

```
1  // presmetka na kvadrat na vnesen broj
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
8      int broj;
9
10     cout << "************************************************" << endl;
11     cout << " Da se presmeta kvadratot na vneseniot broj" << endl;
12     cout << "************************************************" << endl;
13     cout << "Vnesi eden broj" << endl;
14     cin >> broj;
15     cout << "Kvadratot na brojot " << broj << " e: " << broj*broj << endl;
16     system ("pause");
17     return 0;
18 }
19
```

Слика 1: Пресметка на квадрат на внесен број

При креирањето на програмата за пресметка на квадрат на внесен број, покрај коментарот и предпроцесорските наредби кои се наоѓаат во воведот на програмата, во главната функција употребивме искази за приказ на екран и искази за внесување податоци од тастатура. Всушност, тука ја употребивме техниката на внес на податоци. Променливата за која треба да се внесе податок од тастатура, најпрво се дефинира од кој тип е, а потоа, соодветно се применува. Резултатот од извршувањето го гледаме на посебен екран:

```
************************************************
 Da se presmeta kvadratot na vneseniot broj
************************************************
Unesi eden broj
5
Kvadratot na brojot 5e: 25
Press any key to continue . . .
```

Слика 2: Резултат од пресметка на квадрат на број

Задача

Да се пресмета плоштина и периметар на круг!
Тука станува збор за готови математички формули кои треба да ги употребиме при пресметката. На пример: плоштина на круг се пресметува со формулата P=r*r*pi, додека, пак, периметар се пресметува со следната формула: L=2*r*pi.

```
// presmetka na plostina i perimetar na kruznica
#include <iostream>
#include <cstdlib>
using namespace std;

#define pi 3.141

int main()
{
    float radius, P, L;

    cout << "*************************************************" << endl;
    cout << " Da se presmeta plostina i perimetar na kruznica" << endl;
    cout << "*************************************************" << endl;
    cout << "Vnesi go radiusot na kruznicata: " << endl;
    cin >> radius;
    P=radius*radius*pi;
    cout << " Plostinata na kruznicata iznesuva: " << P << endl;
    L=2*radius*pi;
    cout << "Perimetarot na kruznicata iznesuva: " << L << endl;
    system ("pause");
    return 0;
}
```

Слика 3: Пресметка на плоштина и периметар на круг

Тргнувајќи од готовите математички формули **pi** има константна вредност **3.141** и затоа во претпроцесорскиот дел ја дефинираме како константа употребувајќи ја наредбата **#define**. Променливите во главната функција ги дефиниравме како **float**, односно реални броеви кои содржат цел и децимален дел, бидејќи **pi** има децимална вредност и се очекува резултатот од пресметката да е децимален број. Понатаму, се применуваат исказите за приказ на екран и исказот за доделување на вредност кои редоследно се извршуваат и даваат конечен резултат во посебен прозорец:

```
*************************************************
 Da se presmeta plostina i perimetar na kruznica
*************************************************
Vnesi go radiusot na kruznicata:
6
 Plostinata na kruznicata iznesuva: 113.076
Perimetarot na kruznicata iznesuva: 37.692
Press any key to continue . . .
```

Слика 4: Резултат од пресметката на плоштина и периметар на кружница

Задача

Да се пресмета плоштина на триаголник употребувајќи ја математичката формула: P=(a*h)/2

```cpp
// presmetka na plostina na triagolnik
#include <iostream>
#include <cstdlib>
using namespace std;

#define pi 3.141

int main()
{
    float a, h, P;
    cout << "************************************************" << endl;
    cout << " Da se presmeta plostina na triagolnik" << endl;
    cout << "************************************************" << endl;
    cout << "Vnesi ja goleminata na stranata a: " << endl;
    cin >> a;
    cout << "Vnesi ja visinata na triagolnikot: " << endl;
    cin >> h;
    P=a*h/2;
    cout << "Plostinata na triagolnikot iznesuva: " << P << endl;
    system ("pause");
    return 0;
}
```

Слика 5: Пресметка на плоштина на триаголник

За да ја креираме програмата и да добиеме точно решение, при пресметката ја применивме математичката формула за пресметка на плоштина на триаголник. Го дефиниравме типот на променливите, со помош на исказот за приказ на екран испечативме текст како вовед во задачата, ги внесовме вредностите на променливите преку тастатурата и извршивме пресметка со примена на формулата. Притоа, го добивме следниот резултат при извршување на фајлот:

```
************************************************
 Da se presmeta plostina na triagolnik
************************************************
Vnesi ja goleminata na stranata a:
7
Vnesi ja visinata na triagolnikot:
4
Plostinata na triagolnikot iznesuva: 14
Press any key to continue . . .
```

Слика 6: Резултат од пресметка на плоштина на триаголник

Задача

Креирај програма која при внес на број од тастатура ќе го прикаже неговиот претходник и следбеник.

4.11 Споредбени изрази

Во нашето секојдневие честопати се случуваат работи за кои ние треба да донесеме одлуки и зависно од тие одлуки да одлучиме по кој пат да одиме понатаму. На пример, по завршувањето на задолжително училиште се наоѓаме на еден крстопат на кој треба да одлучиме, дали ќе го продолжиме образованието на факултет или тука завршуваме. Тука имаме две можности. Првата можност е ако го продолжиме образованието на универзитет се отвораат широки можности да го прошириме нашето знаење и со тоа да бидеме конкурентни при изнаоѓање на добро платена и стручна работа. Втората можност е да одлучиме да не го продолжуваме образованието и да останеме на досега стекнатото знаење, а со тоа и можноста за конкурентност на пазарот се намалува. Сепак, ова се тешки животни одлуки, но се добар пример за вовед во она што ќе го зборуваме во оваа наставна содржина.

Забелешка!

Дали некогаш сте се нашле во ситуација да треба да одлучувате? Како одлучивте? Што ве натера така да размислувате? Опишете ја ситуацијата!

Слични работи се случуваат и кај компјутерите. На пример: кога некоја програма одлучува дали условот е задоволен и ако е задоволен која акција да се преземе, и обратно, ако не е задоволен условот што понатаму?!

Ваквите изрази се нарекуваат **споредбени изрази** и нив корисникот ги избира кога има алтернатива. Тоа, значи, дека ќе се преземе една акција доколку тврдењето е точно и друга акција доколку не е точно. На пример, кога започнавме да креираме програми истакнавме дека **исказите (наредбите)** се извршуваат редоследно една, по друга. При примената на ваквите споредбени изрази се губи редоследот на програмата, бидејќи нејзиното извршување зависи од точноста на дефинираното тврдење.

Овие споредбени изрази во програмскиот јазик C++ се нарекуваат **логички изрази** или **Boolean изрази**. При креирањето на ваквите изрази се користат оператори за споредба:

Оператор во C++	Значење	Математички оператор
==	Еднакво на...	=
!=	Нееднакво на...	≠
>	Поголемо од...	>
<	Помало од...	<
>=	Поголемо или еднакво на...	≥
<=	Помало или еднакво на...	≤

Табела 1: Споредбени оператори

4.11.1 Структура за избор од две можности

При креирањето програми со споредбени изрази, најчесто употребувана е **структурата за избор од две можности**. Во зависност од вредноста на изразот, во оваа структура се врши избор помеѓу две можности. Изразот може да има вредност или **точно** или **неточно**, тоа значи, ако условот на изразот е точен, тогаш ќе се изврши една наредба, а ако не е точен, тогаш ќе се изврши друга наредба или тука програмата ќе заврши.

Според горенаведеното, можеме да утврдиме дека постојат два вида разгранување на изразите, зависно дали условот е задоволен или не: **еднократно** и **двократно**.

Еднократното разгранување се применува ако условот на задача е исполнет, тогаш да се изврши некоја наредба, а ако условот не е исполнет, тогаш програмата да заврши, односно **„ако услов, тогаш наредба“**. За таа цел се применува исказот **if...then**.

На пример: кога корисникот ќе внесе години, програмата да прикаже дали е малолетен или не.

```
1  // if...else наредба
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
8      int godini;
9      cout << " Kolku godini imate? " << endl;
10     cin >> godini;
11
12     if (godini <=18)
13         cout << " Maloletni ste!" << endl;
14     else
15         cout << "Polnoletni ste!" << endl;
16
17     system ("pause");
18     return 0;
19 }
```

Слика 1: If...Then исказ

Важно!

По исказот if не се става знакот точка и запирка од причина што во овој исказ треба да се извршуваат наредбите кои следат. Со додавањето на точка и запирка ќе означиме крај на исказот.

Од главниот програм можеме да забележиме дека дефинирана е променлива **godini** од типот целобројна вредност, т.е. **int**. По приказот на екран „**Kolku godini imate?**“, корисникот внесува податок од тастатура и кликнува **Enter**. Тогаш се врши проверка дали внесениот податок е помал или еднаков на осумнаесет (18) и ако е задоволен тој услов, тогаш на екран ќе се прикаже пораката „**Maloleten ste**!“. Доколку условот не е задоволен,

односно се внесе податок поголем од осумнаесет (18), тогаш нема да се изврши никаква акција.

Двократното разгранување, всушност, значи ако некој услов е задоволен, тогаш да се изврши дефинираната наредба, а ако не е задоволен, тогаш да се изврши друга наредба, односно „**ако услов, тогаш наредба1, инаку наредба2**“. За таа цел се применува исказот **If...else**.

На пример, задачата која претходно ја изработивме да ја продолжиме со дефинирање и на друга наредба.

```
1   // if...then naredba
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5
6   int main()
7   {
8       int godini;
9       cout << " Kolku godini imate? " << endl;
10      cin >> godini;
11
12      if (godini <=18)
13          cout << " Maloletni ste!" << endl;
14
15      system ("pause");
16      return 0;
17  }
18
```

Слика 2: If...Else наредба

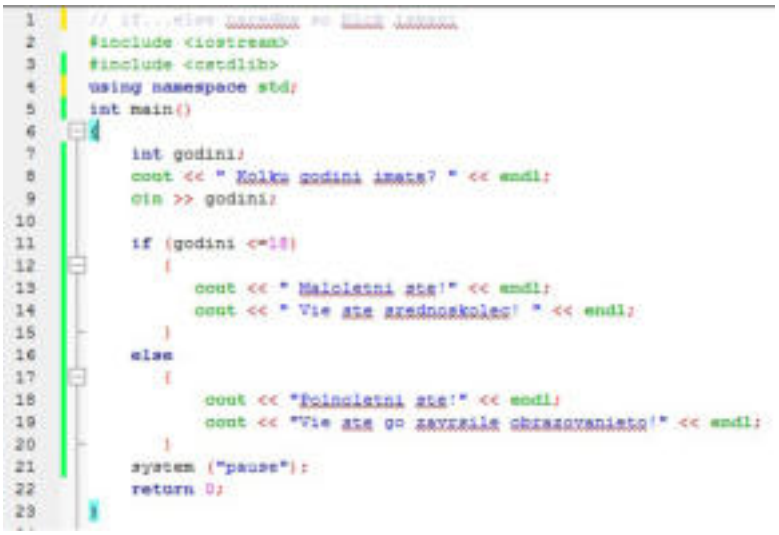
Според тоа, при внесот на податокот „**godini**“, програмата проверува дали внесениот податок е помал или еднаков на осумнаесет (18).. Ако е условот исполнет, односно внесениот податок се наоѓа во рангот измеѓу нула (0)**0** и осумнаесет (18), тогаш на екран ќе се испише пораката: „**Maloletni ste!**“. Доколку условот не е задоволен, односно внесениот податок е поголем од бројот осумнаесет (18), тогаш да се изврши акција, односно на екран да се испише „**Polnoleten ste!**“:

```
 Kolku godini imate?
13
 Maloletni ste!
Press any key to continue . . .
```

```
 Kolku godini imate?
36
Polnoletni ste!
Press any key to continue . . .
```

Слика 3: Резултат од извршен фајл на If...else

При употребата на **If...else** исказот, може да се случи наредбата да содржи повеќе искази. Овој начин на пишување на наредбите се вика блок од искази и тие се стават помеѓу големи загради. На пример, да дадеме блок-искази во програмата во која ја работевме:



```
// if...else наредба со блок наредби
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int godini;
    cout << " Kolku godini imate? " << endl;
    cin >> godini;

    if (godini <=18)
       {
          cout << " Maloletni ste!" << endl;
          cout << " Vie ste srednoshkolec! " << endl;
       }
    else
       {
          cout << "Polnoletni ste!" << endl;
          cout << "Vie ste go zavrshile obrazovanieto!" << endl;
       }
    system ("pause");
    return 0;
}
```

Слика 4: Блок-искази во If...else

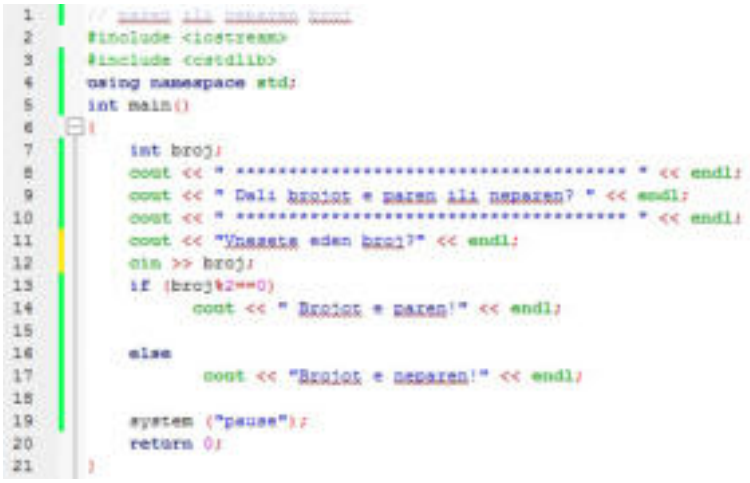
Тоа, значи, дека по внесувањето на податокот за променливата **„godini“** ќе се направи проверка дали внесениот податок е **помал или еднаков** на осумнаесет (18).. Ако условот е задоволен, тогаш на екран ќе се испишат две пораки една под друга, а ако не е задоволен условот, повторно на екран ќе се испишат две други наредби, зависно од исполнетоста на условот. Блокот на искази секогаш се пишува помеѓу две големи загради.

4.11.2 Изработка на програми со структура за избор од две можности

Ајде да креираме неколку програми во кои ќе поставиме услов. Со внесувањето на податоците ќе правиме проверка дали условот е задоволен или не и согласно со одговорот ќе се извршуваат наредби.

Задача

Да се пресмета плоштина на триаголник употребувајќи ја математичката формула: P=(a*h)/2



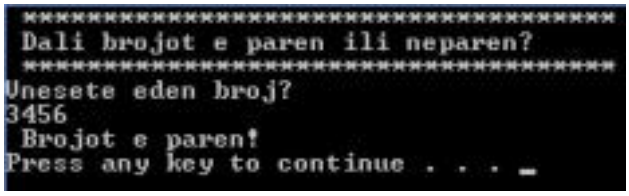
```
// парен или непарен број
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int broj;
    cout << " ************************************ " << endl;
    cout << " Dali brojot e paren ili neparen? " << endl;
    cout << " ************************************ " << endl;
    cout << "Vnesete eden broj?" << endl;
    cin >> broj;
    if (broj%2==0)
          cout << " Brojot e paren!" << endl;

    else
          cout << "Brojot e neparen!" << endl;

    system ("pause");
    return 0;
}
```

Слика 5: Проверка дали бројот е парен или непарен

Во оваа програма по внесувањето на бројот од тастатура се прави проверка дали е парен или непарен број. Според математичкото правило „**секој број кој е поделен со бројот два (2) има резултат цел број и е без остаток, тогаш е парен број**“, го утврдуваме условот. Тука го применуваме **операторот „%“ (mod)** кој го прикажува остатокот од делењето. Внесениот број поделен со бројот два да има остаток еднаков на нула, тогаш на екран да се напише порака „**Brojot e paren!**“. Ако условот не е задоволен да се испише порака „**Brojot e neparen!**“. Резултатот од извршувањето на програмата е прикажан на следниот прозорец:



Слика 6: Резултат од програмата за проверка на бројот дали е парен или непарен

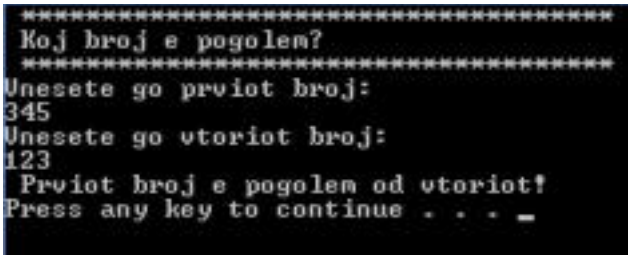
Задача

Да се креира програма во која ќе внесеме два броја. Притоа ќе создадеме услов кој ќе проверува кој број од внесените е поголем.

```
1  // Koj broj e pogolem
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5  int main()
6  {
7      int broj1, broj2;
8      cout << " ************************************ " << endl;
9      cout << " Koj broj e pogolem? " << endl;
10     cout << " ************************************ " << endl;
11     cout << "Vnesete go prviot broj: " << endl;
12     cin >> broj1;
13     cout << "Vnesete go vtoriot broj: " << endl;
14     cin >> broj2;
15     if (broj1>broj2)
16           cout << " Prviot broj e pogolem od vtoriot!" << endl;
17     else
18           cout << " Vtoriot broj e pogolem od prviot!" << endl;
19     system ("pause");
20     return 0;
21 }
```

Слика 7: Програма која проверува кој број е поголем

Во програмата по внесувањето на двата броја го употребуваме исказот **If ... else** во кој креираме услов за проверка дали првиот број е поголем од вториот. Ако е исполнет условот се испишува порака на екран дека првиот број е поголем од вториот, а доколку не е исполнет да пишува порака дека вториот број е поголем од првиот. Резултатот од проверката е следниот:



Слика 8: Резултат од проверката кој број е поголем

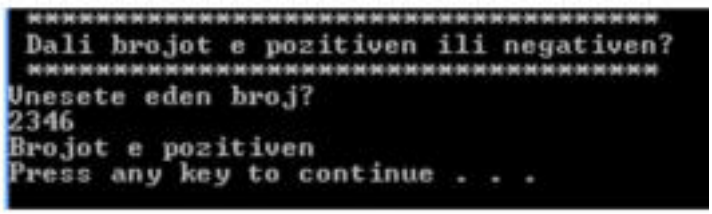
Задача

Да се креира програма која ќе проверува дали еден број е позитивен или негативен!

```
1  // pozitiven, negativen ili ednakov na 0
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5  int main()
6  {
7      int broj;
8      cout << " ************************************ " << endl;
9      cout << " Dali brojot e pozitiven ili negativen? " << endl;
10     cout << " ************************************ " << endl;
11     cout << "Vnesete eden broj?" << endl;
12     cin >> broj;
13     if (broj<0)
14           cout << " Brojot e negativen!" << endl;
15     else
16           if (broj==0)
17                cout << "Brojot ne e nitu pozitiven nitu negativen!" << endl;
18           else
19                cout <<"Brojot e pozitiven" << endl;
20
21     system ("pause");
22     return 0;
23 }
```

Слика 9: Проверка со if...else дали бројот е позитивен, негативен или еднаков на нула

Од оваа програма, можеме да забележиме дека по внесувањето на број од тастатура креиравме услов, ако внесениот број е помал од нула, тогаш да се испише порака дека тој е негативен број. Во спротивно, ако условот не е задоволен, да се пристапи кон проверка на условот дали внесениот број е еднаков на нула и тогаш ако овој услов е задоволен на екран да се испише дека бројот не е ниту позитивен, ниту негативен, а во спротивно да се испише порака дека внесениот број е позитивен. Забележуваме, дека, тука имаме применето **двапати исказ If...else**, кои се извршуваат зависно од исполнетоста на претходниот услов. Оваа техника се вика **техника на вгнездување на изкази**. Резултатот од пресметката го гледаме на екранот:



Слика 10: Резултат од проверката дали бројот е позитивен, негативен

Запомни!

Споредбените изрази се користат при избор од две можности. Тие се нарекуваат логички или Boolean изрази и при нивното креирање се користат операторите за споредба. Структурата на програмата која содржи избор од две можности може да биде еднократно разгранета и двократно разгранета. Во првиот случај се користи изказот if, додека во вториот случај исказот if...else. Овие искази во својата структура можат да содржат повеќе искази и тие се запишуваат во големи загради.

Прашања

1.Дефинирај ги споредбените изрази!
2.Наведи ги операторите кои се употребуваат за споредба!
3.Какво може да биде разгранувањето во програмата во која може да се направи избор од две можности? Која е разликата меѓу нив?
4.Напиши ги синтаксите на If и If...Else!

4.12 Структура за повторување во циклус до исполнување на услов

Досега сега креиравме многу програми кои имаат различна структура, различни искази, различна цел и решение. При креирањето програми во програмскиот јазик C++ исказите се извршуваат една, по една, редоследно. Овој начин на извршување на исказите се вика **физички редослед** на извршување.

Кога креиравме програми со примена на исказот **If...Else**, редоследот на извршување на исказите се менуваше зависно од исполнетоста на условот кој е дефиниран. Но, честопати е потребно некои искази да се повторуваат, односно да се извршат повеќепати. Ваквото повторување на исказите претставува **циклус**, а овие структури се викаат **циклични структури или структури за повторување**. Редоследот на извршување на исказот во циклус се вика **логички редослед**, бидејќи тој зависи од исполнетоста на условот.

Според тоа, циклусот извршува една или повеќе инструкции повеќепати сè додека еден или група услови не се исполнети. Најчесто употребувани искази кои извршуваат циклуси се: **while**, **do – while** и **for** циклусот.

При примената на структурите за повторување можни се две ситуации и тоа:

- однапред се знае колку пати циклусот ќе се повтори;
- бројот на повторувања зависи од некој услов и тој број не е однапред познат. Условот може да се најде на почетокот на циклусот или, пак, на крајот на циклусот.

Циклусот **while** е наједноставниот циклус кој завршува со исполнување на зададениот услов, а тоа значи дека while инструкцијата тестира услов при кој се извршуваат низа од наредби сè до негово исполнување. Неговата синтакса е следната:

```
While (услов)
{
наредба 1;
наредба 2;
наредба 3;...
        }
```

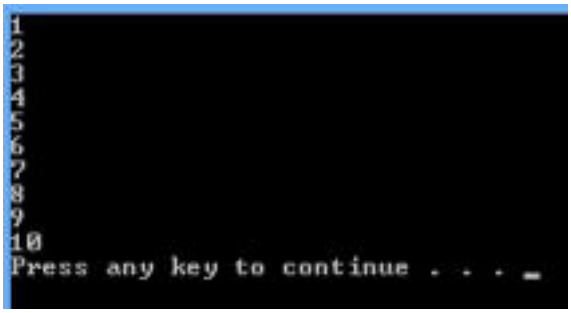
Од прикажаната синтакса, **while** инструкцијата е контролна структура која се вика **циклус** или **loop**. Инструкцијата, односно наредбата која треба да се изврши секојпат при поминување на циклусот се вика **тело на циклусот**. Притоа, пред да започне извршувањето на циклусот, прво се проверува дали условот е задоволен. Ако условот е задоволен се започнува со извршување на исказите, а ако не е задоволен, воопшто и не се започнува.

На пример:

```
1  // испишување на броеви од 1 до 10
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5  int main()
6  {
7      int i=1;
8      while (i<=10)
9      {
10         cout << i <<"\n";
11         i++;
12     }
13     system ("pause");
14     return 0;
15 }
```

Слика 1: Програма за испишување на низата броеви од еден (1) до десет (10)

Од програмата, забележуваме дека во главната функција променливата „**I**“ е дефинирана со типот „цел број“ и истата е иницијализирана со вредност еден (1). Со примена на структурата за повторување **while** да се испишуваат броевите на низата од еден (1) до десет (10), сè до исполнување на условот зададен во циклусот **while**. На екран ќе се прикаже следното:



Слика 2: Испишување на низата броеви од еден (1) до десет (10)

Забелешка!

Во оваа програма низата од броеви е вертикално претставена. За да се претстават броевите хорионтално го бришеме изразот „\n” и во наводниците додаваме запирка.

Слична вакава програма која на екран испишува знаци до определен број е прикажана во следната програма:

Слика 3: Испишување низа од знаци

Циклусите кои користат **while** инструкција можат да бидат:

- циклуси контролирани од бројач;
- циклуси контролирани од настан.

Циклусите контролирани од бројач користат променлива која го контролира циклусот. За таа цел се иницијализира контролната променлива на бројачот и при секој круг на извршување, бројачот се зголемува. На пример:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i=1;
    while (i<6)
    {
        cout << " Programiranje vo C++" << endl;
        i++;
    }
    system ("pause");
    return 0;
}
```

```
Programiranje vo C++
Programiranje vo C++
Programiranje vo C++
Programiranje vo C++
Programiranje vo C++
Press any key to continue . . .
```

Слика 4: Циклус контролиран од бројач

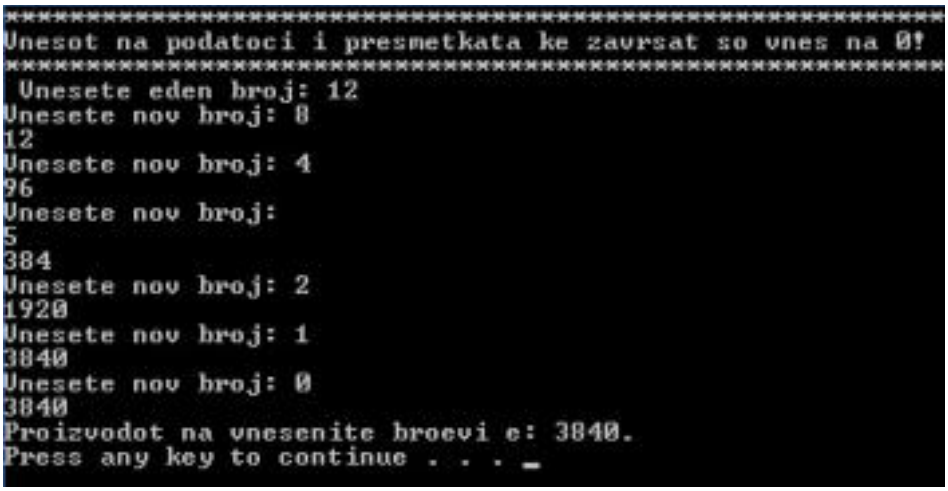
Според задачата, реченицата „**Programiranje vo C++**“ ќе се испишува сè додека не се задоволи условот: **да се напише пет пати**. На секој круг бројачот i се зголемува за еден (1).

Кај **циклусите кои се контролирани од настан** условот за прекин зависи од некој настан кој се случува додека телото на циклусот се извршува. На пример: да се пресмета производ на внесените броеви, а внесот ќе прекине со внесување нула (0) од тастатура:

```
#include <iostream>
#include <cstdlib>
using namespace std;
main()
{
    int broj, p=1, br=1;
    cout << "*************************************************************" << endl;
    cout << "Vnesot na podatoci i presmetkata ke zavrsat so vnes na 0!" << endl;
    cout << "*************************************************************" << endl;
    cout <<" Vnesete eden broj: ";
    cin >> broj;
    while (broj!=0)
    {
        p=p*broj;
        br++;
        cout << "Vnesete sled broj: ";
        cin >> broj;
        cout << p << endl;
    }
    if (br==0)
    p=0;
    cout << "Proizvodot na vnesenite broevi e: " << p << "." << endl;
    system ("pause");
    return 0;
}
```

Слика 5: Циклус контролиран од настан

Оваа програма на циклусот While ќе се извршува сè додека не се внесе нула (0) од тастатура. А, тоа значи дека внесувањето нула (0) претставува настан кој ќе го прекине, односно заврши циклусот. Затоа се вели дека ваквите циклуси се контролирани од настани. Резултатот од извршувањето на кодот е прикажан на следната слика:

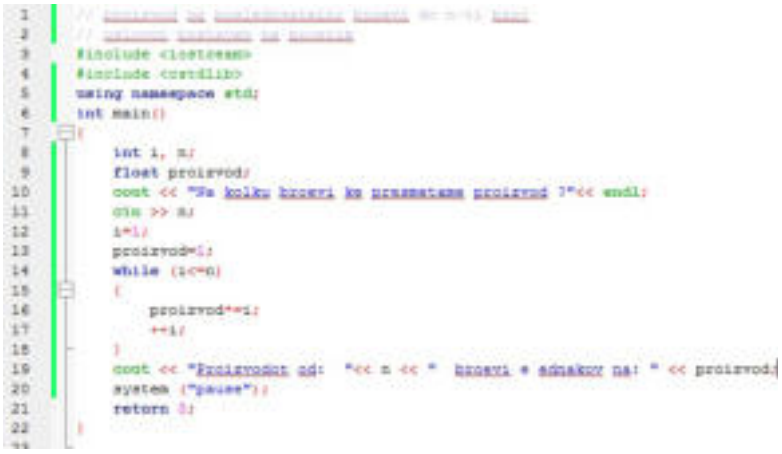


Слика 6: Резултат од извршување на циклус

4.12.1 Изработка на едноставни програми со структура за повторување на циклус дури е исполнет услов

Кај структурите со повторување до задоволување на услов, условот може да биде дефиниран на **почетокот од циклусот** или на **крајот од циклусот**. Кај првите, најпрво се проверува дали условот е задоволен за да можат да се извршат низите со искази, а кај вторите се извршуваат исказите додека не се задоволи условот. На пример, ќе пресметаме производ на последователни броеви до **n-ти** број, на двата начина:

Прв начин: условот се наоѓа на почетокот на исказот за повторување на циклус:



Слика 7: Уловот во циклусот е дефиниран на почеток

Втор начин: ќе се извршат низите од искази додека не се задоволи условот:

Слика 8: Програма во која се извршуваат исказите до задоволување на условот

Задача

Применувајќи ги структурите за повторување во циклус сè додека не се задоволи условот, да се креираат следните програми:

1. Да се испишат на екран парните броеви до n-ти број!
2. Да се пресмета збирот на непарните броеви до n-ти број!
3. Да се пресмета производот на парните броеви до n-ти број!

Запомни!

Редоследот на извршување на исказите може да биде физички и логички. Повторувањето на исказите неколку пати претставува циклус, а структурите циклични структури или структури за повторување. Најчесто употребувани искази за циклуси се: while, do-while и for. При употреба на while прво се проверува условот, а потоа се извршуваат исказите. Кај do-while се извршуваат исказите сè додека не се постигне условот.

Прашања

1.Што се структури за повторување? Дефинирај!
2.Кои се најчесто користените искази за циклуси?
3.Наведи ги синтаксите на while и do-while!
4.Како се контролираат циклусите при примена на while и do-while?
5.Дали се знае колку пати ќе се повторува циклусот? Објасни!

ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

Задача 1

Напиши програма со која ќе го добиеш следниов приказ на екран:
Prv cas: INFORMATIKA
Programiranje vo C++

Задача 2

Кои се вредностите на променливите a и b, по извршување на слендите наредби?

```
a=15
b=23
a=a+b
b=a-b
a=a-b
```

Задача 3

Што ќе се прикаже на екран по извршување на следниот код:

```
int a, b,c;
cin >> a;
cin >> b;
cin >> c;
cout << a << “,” << b << “,” << c << “. “ << endl;
```

ако корисникот ги внесе следните вредности: пет (5), шест (6) и седум (7) последователно?

Задача 4

Што ќе се испише на екран по извршувањето на следната низа од наредби?

```
int a=8;
int b=10;
cout << (a+b)/2
```

Задача 5

Колку пати ќе се изврши следниот циклус?

```
int i=2
do
{
cout << i << “ “ ;
i ++;
}
While (i<=10)
```

Задача 6

Напиши програма за пресметка на плоштина на правоаголник. Излезот на екран да ги прикаже димензиите на страните и резултатот од пресметката на плоштината. Формула за пресметка на плоштина на правоаголник е: p=a*b.

Задача 7

Напиши програма која ќе пресметува аритметичка средина на два броја. Како би се извршила пресметка на аритметичка средина на осум броја?

Задача 8

Да се креира програма која проверува дали внесениот број е парен или непарен.

Задача 9

Да се креира програма која го пресметува периметарот на квадрат, доколку внесената вредност за страната „a“ е позитивна. Во спротивно да се испише порака „Пресметката не може да се изврши! Внесете позитивен број!“.

Задача 10

Со примена на циклусот do – while, да се пресмета збирот на парните броеви до n-ти број

Задача 11

Да се пресмета збирот на броевите кои циклично се внесуваат од тастатура, с додека не се внесе бројот нула (0).

Онлајн живеење

Вовед во веб дневници (блогови)
Поим за блог и негова примена
Изработка на блог
Коментирање содржини на блоговите
ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

5.Вовед во веб-дневници (блогови)

Да се потсетиме!

Што се програми? Наведи дефиниции за програмирање! Како се изработува програма? Кои програмски јазици ги познаваш?

Модерното општество и трендот на масовната глобализација ја наметнуваат потребата од примена на **Интернет технологијата** во процесот на образованието, но и во секојдневието воопшто. Со примената на интернет технологијата на корисниците им се овозможува употреба на различни сервиси, како што се: комуникација, размена на пораци, споделување информации, купување преку интернет, симнување слики, музика, видео содржини и сл.

Според тоа, **Интернет** е најпознатата светска мрежа составена од милиони компјутери низ целиот свет, кои меѓусебно се поврзани на најразлични начини, со цел меѓусебно да комуницираат и да разменуваат информации. Сепак, најупотребуван сервис на интернет е комуникацијата, односно размената на податоци и информации преку е-маил, онлајн-сервиси за комуникација, социјални мрежи, но во поново време и примената на **веб-дневници** или **блогови**. Бидејќи, тука станува збор за комуникација, корисникот треба да ги почитува правилата за етичко користење на интернет:

- не споделувај лични податоци, како и податоци од членовите
- на твоето семејство, како што се: име и презиме, адреса, телефон или фотографии со непознати соговорници на интернет, бидејќи не знаеме кој стои од другата страна на компјутерот. Не се добронамерни сите луѓе што комуницираат на интернет;
- никогаш не го внесувај бројот на кредитната картичка на родителите, на интернет, за да купиш или порачаш нешто без нивна дозвола;
- не ги откривај лозинките што ги употребуваш за пристап до сервисите на интернет, за да не бидат злоупотребени;
- не употребувај навредливи и непристојни зборови при разговор на интернет;
- при комуникација на интернет почитувај ги ставовите на другите луѓе, иако можно е да се спротивни од твоите. Однесувај се пријателски при онлајн-комуникација;
- при пишување текст на интернет, не пишувај со големи букви, тоа е исто како да викаш.

Во поново време со актуализацијата на социјалните мрежи се појавува терминот **дигитален отпечаток**, како резултат на објавите кои корисникот ги публикува на интернет. При нашето секојдневно сурфање убедени сме дека сме зашитени и анонимни, но не е така. Тоа покажува дека активностите на корисниците на социјалните мрежи се следат од многу

повеќе луѓе отколку што мислиме. Преку преземените активности од страна на корисникот: допаѓање на страници и објави, споделување линкови, статуси, слики, видеа, коментирање содржини и сл., се формира **дигиталниот отпечаток** на корисникот, односно се креира база на активности на секој корисник, преку кој се осознава за каква личност станува збор, кои се неговите инстереси и занимливости и како да се влијае врз него.

Според тоа, **дигитален отпечаток се сите интернет информации (и позитивни и негативни) кои се случајно или намерно оставени**.

Дигиталниот отпечаток има и позитивни и негативни страни, но тие треба да се контролирани. На пример како позитивна страна на дигиталниот отпечаток е кога бараме некој рецепт на Интернет и преку тој рецепт можеме да дојдеме до многу кулинарски страници. Но има и негативни страни како последица на неодговорно употребување на Интернет. На пример фотографиите за кои сме мислеле дека се приватни, станале јавни, личните податоци во одредени моменти не се добро заштитени и сл. Затоа, пред да се објави нешто на интернет, корисникот треба добро да размисли кои се придобивките од објавата и како истата може да му наштети. Пред секоја објава корисникот треба да ги измери позитивните и негативните вредности за да направи правилна одлука пред да објави. Најчести совети за позитивен дигитален отпечаток се следните:

- одјавете се при напуштање на социјалнаите мрежи или маил;
- не ги кажувајте лозинките;
- не ги запишувајте јавно лозинките;
- при креирање лозинка коитстете букви, бројки и знаци;
- често бришете ја архивата;
- не ги делете личните податоци на Интернет.

Во оваа тема ќе ставиме акцент на **веб-дневниците**, односно **блоговите**, како моќна онлајн-алатка која овозможува врска меѓу наставниците и учениците, споделување на наставни единици, вежби, видеозаписи, додавање на дополнителен материјал за оние кои сакаат да научат нешто повеќе од она што се нуди во учебниците и сл.

За да ги воочиме функционалностите и можностите кои ги нудат блоговите со помош на пребарувач, да се обидеме да пронајдеме блогови од различни автори и за различни теми. Постојат многу блогови кои имаат содржини токму за материјата која ја изучуваме, **информатика**.

Клучни поими

веб-дневник, блог, блогер, логирање, корисничко име, лозинка, запис, коментар, регистрација, дигитален отпечаток.

5.1 Поим за блог и негова примена

Блог или **веб-дневник** претставува онлајн-дневник, каде што содржината се прикажува по хронолошки редослед, односно најновите вести или содржини се прикажуваат први, а останатите одат подолу, т.е. сортирани се така што од најновите се оди кон постарите објави.

Зборот блог потекнува од зборовите **web** што значи мрежа и **log** што значи внес.

Всушност, блоговите претставуваат комуникациска алатка која ги поврзува корисниците, бизнис-групите и другите веб-истомисленици.

Авторот на блогот се вика **блогер**. Тој не само што креира блог, туку постојано ја обновува и уредува неговата содржина. Креирањето, обновувањето и уредувањето на содржината на блогот се вика **блогирање**. Блогер може да биде секој којшто има познавања за работа со компјутер и интернет технологии, и е подготвен да дискутира на одредена тема.

Содржината на веб-дневникот може да биде различна: текстови, белешки, дискусии, фотографии, видеоклипови, адреси кон други локации на интернет и сл. Блогерите уште од самиот почеток знаат што сакаат да пишуваат, да коментираат, односно да блогираат. Најчести теми за блогирање се следните:

- прирачници за образование и теми од образованието;
- познати личности: биографии и нивни постигнувања;
- спорт;
- здравје и убавина;
- теми за родители и деца;
- компјутерски игри;
- домашни миленичиња и нивно одгледување;
- рецепти;
- политика;
- водич за различни области;
- животни искуства;
- услуги и производи;
- патувања;
- мода;
- уредување на домот и сл.

Секако, постојат и многу други теми кои се од интерес на корисниците. Блогерот одлучува за која тема ќе креира блог. Најчесто, блогерите се одлучуваат за теми кои им се познати, за кои имаат предзнаења и со помош на дискусиите да го прошират и продлабочат знаењето, наменети се за јавноста, или пак, за одредена целна група која дискутира во врска со темата или темите што се поставени на блогот. Дискусиите можат да бидат забавни и поучни, исто така, можат да се најдат и содржини кои се

од голема важност за интернет популацијата.

За да ги осознаеме можностите кои ги овозможува блогирањето, ќе се оспособиме да креираме блог и да бидеме дел од блог-заедницата.

Запомни!

Блог или веб-дневник претставува локација на Интернет во форма на електронски весник во кој содржините се прикажуваат по обратен временски редослед. Блогер е авторот на веб-дневникот, односно блогот. Блогирање е процесот на креирање, уредување, коментирање и одржување на блогот. Почитувањето на правилата за етичко однесување при комуникација преку интернет важат и при блогирање и при оставање на дигитален отпечаток. Дигитален отпечаток претставува трагата која ја оставаме по примена на различните сервиси на интернет.

Прашања

1.Што е блог?
2.Што се блогери?
3.Кој може да биде автор на блог?
4.Што е блогирање?
5.Какви теми можат да се употребат при креирање блог?
6.Кој ги пишува содржините на блогот?
7.Набројнеколку правила за етичко користење на блоговите!
8.Дали блоговите може да ги применувам како алатка за училишни цели? Како?
9.Што е дигитален отпечаток?

5.2 Изработка на блог

Креирањето на блоговите е едноставен процес. Наједноставен, најбрз и најефтин начин за примена на неговите можности, специјалности и функционалности е преку бесплатните блог-сервиси. Постојат многу блог-сервиси за изработка на блогови:

1.Blogger;
2.Drupal;
3.Ghost;
4.Joomla;
5.Magento;
6.Medium;
7.Squarespace;
8.Tumblr;
9.Typepad;
10.Weebly;
11.Wix;
12.WordPress.

Најпопуларни блог-сервиси се **Blogger** и **WordPress**.

Blogger претставува една од најстарите платформи на **Google** и бидејќи е дел од **Google** пакетот лесно пристапуваме со употреба на нашата лична сметка (**account**) на **Google**, односно **Gmail**. Според тоа, за да започнеме со блогирање на Blogger, најпрво треба да имаме свое корисничко име и лозинка (**Gmail account**).

Слика 1: Лого на Blogger

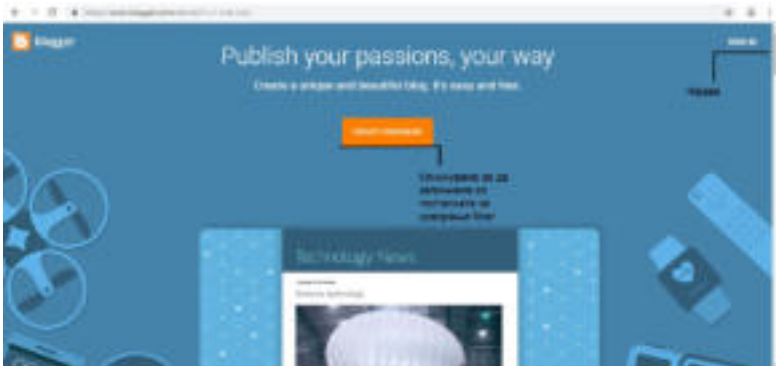
Бидејќи **Blogger** е хостиран онлајн, покрај него корисникот добива и поддомен кој изгледа вака: именасајтот.blogspot.com.

Можностите на **Blogger** можат да се користат за: пишување дневници, поставување слики, пишување на свои трудови, портал со вести и новости, прирачници и сл. Притоа, не е ограничен бројот на блогови кои авторот може да ги креира и одржува. Покрај тоа, корисникот може да оствари и заработка со блогот преку рекламирање, промоција на разни производи и сл.

5.2.1 Чекори за креирање блог

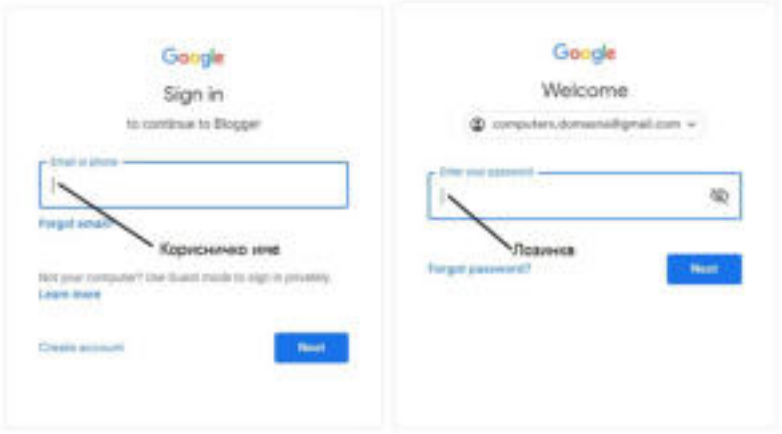
За да ја започнеме постапката за креирање блог на бесплатниот сервис Blogger, најпрво треба да креираме сметка на Gmail, потоа го стартуваме веб-прелистувачот (Browser), како на пример: Google Chrome, Internet Explorer, Mozilla Firefox, Opera и сл. Во адресната лента на прелистувачот ја пишуваме адресата на бесплатниот блог-сервис,

www.blogger.com, при што се појавува следниот прозорец:



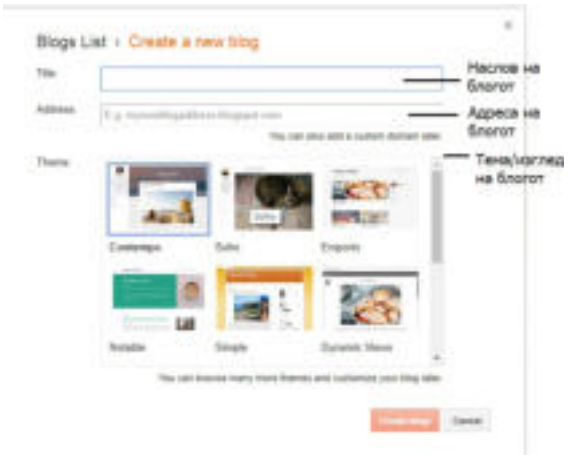
Слика 2: Почетен екран при повикување на страницата за креирање блог

Доколку корисникот нема своја сметка (**Account**), кликнува на опцијата **Sing in** и ја започнува постапката за регистрација на **Gmail**, и обратно, доколку има сметка на Gmail, тогаш ја избира опцијата **Create New Blog**, при што се отвора прозорецот за најава:



Слика 3: Прозорец за најава

По најавата корисникот му доделува име на блогот, адреса и избира тема, односно изглед на блогот. Тоа се прави со пополнување на полињата од следната слика:

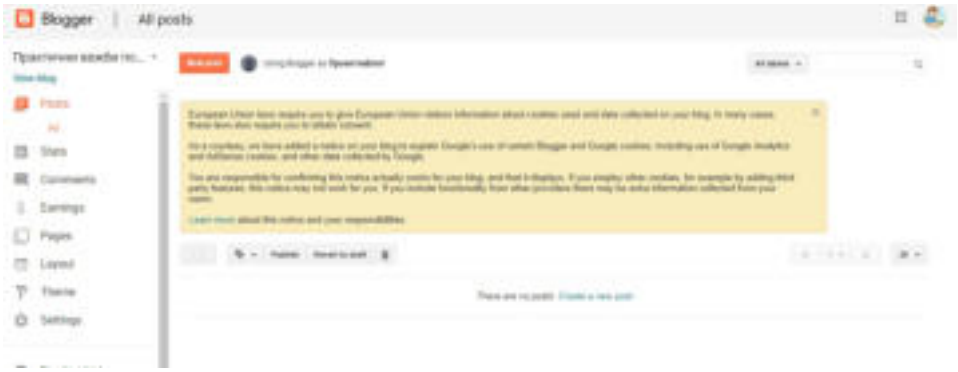


Слика 4: Креирање блог

Забелешка!

Покрај темите, изгледите кои се дадени како опција, корисникот може да додаде тема надвор од листата на понуди.

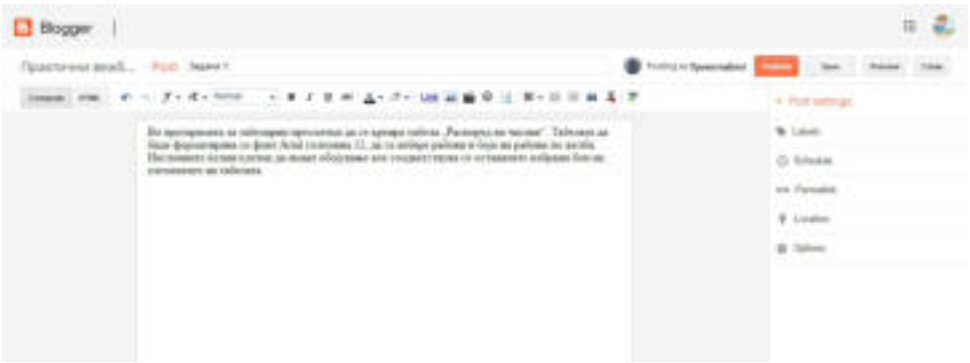
Кога корисникот ќе ја запише адресата на блогот, тогаш на левата страна во близина на рамката за внес се појавува порака:
Checking Address Availability, а тоа значи дека се врши проверка, дали внесената адреса е достапна, т.е. дали не е употребена од друг корисник. Потоа, се избира тема или изглед на блогот која што е соодветна со содржината на блогот. На крајот, се избира копчето **Create Blog** и се пристапува кон контролната табла (**Dashboard**) на блогот. Преку оваа контролна табла корисникот ја уредува содржината на блогот, додава нови содржини, менаџира со коментарите и сл.



Слика 5: Контролна табла (Dashboard) на блог

5.2.2 Уредување содржина на блог

По појавата на контролната табла (**Dashboard**), корисникот може да започне со уредување на содржината на блогот. За да се креира објава (**Post**), се кликнува на копчето **New post**, при што се прикажува следниот прозорец:



Слика 6: Креирање објава (Post) на блог

Со кликнување на копчето **Publish** се прави објава на содржината на блогот. Објавата може да биде зачувана со кликнување на копчето **Save**,

прегледана со кликнување на копчето preview, како и опција за затворање на објавата, односно **Close**. Од десната страна на прозорецот забележуваме дека има опции за дополнителни уредувања на поставките (**Post settings**) на објавата (**Post**), како на пример: додавање ознаки, дали објавата автоматски ќе се прикаже со кликнување на копчето **Publish** или, пак, корисникот ќе постави датум и време на објавата, додавање на локацијата од каде што која е направена објавата и сл.

Забелешка!

Кога ќе се публикува објавата повторно како главен екран се појавува Dashboard. Во контролната табла (Dashboard) пристап има само администраторот на блогот. Со кликнување на линкот Post може да се види целата листа на објави.

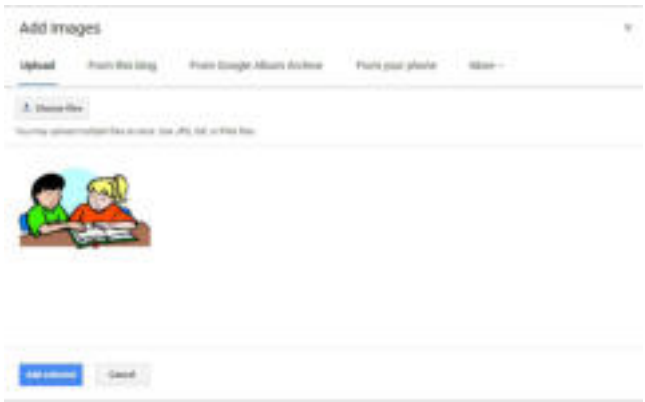
Од самиот прозорец за креирање објава се забележува дека пишувањето, едитирањето и форматирањето на содржините на објавата е исто како во **MS Word**, што значи дека корисникот може да избере фонт, големина на фонт, боја на фонт, стилови на фонт, додавање на линкови, додавање на слики и сл.

5.2.3 Додавање слика и видео во блог

Додека е отворен прозорецот за креирање објава, најпрво, се сместува курсорот на местото каде што треба да се додаде слика и се избира алатката за додавање слика:

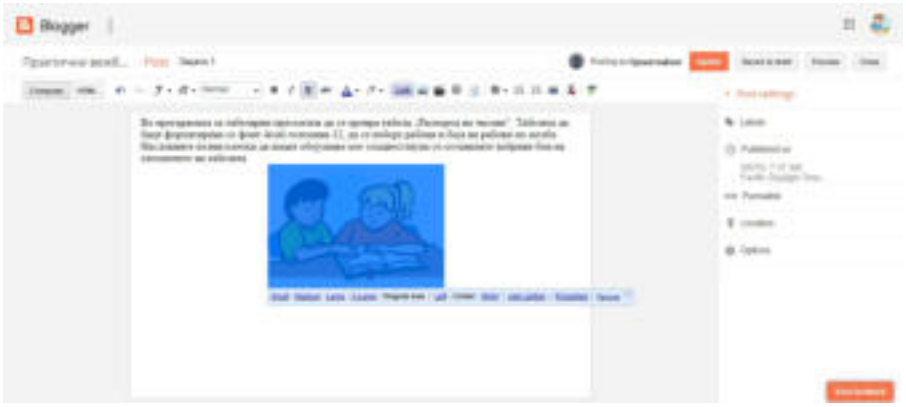
Слика 7: Алатка за додавње слики

Потоа, се отвора следниот прозорец, преку кој се прикачува слика:



Слика 8: Избор на слика за објава

Преку опцијата **Choose Files** се избира слика која ќе се прикачи од компјутерот. Селектираната слика се објавува со кликнување на копчето **Add Selected**, кое се наоѓа во левиот долен агол на прозорецот:



Слика 9: Уредување слика во објава

Забелешка!

Покрај тоа што може да се прикачи слика од компјутер, при креирање објава со слика може да се употребуваат и сликите кои се веќе употребени во блогот, слики од архивата на Google, како и слики од телефонот.

Од прозорецот, при селектирање на додадената слика се појавуваат опции за нејзино уредување, како на пример: уредување на димензиите на сликата, подредување на сликата, додавање ознака на сликата, дополнителни поставки и нејзино бришење.

Постапката за додавање видео во блогот е иста како и додавање слика. За да се додаде видео во објавата, потребно е да има симнато видео и зачувано во компјутерот. Со кликнување на опцијата **Choose video to upload**, видеото почнува да се прикачува на обајвата. По завршувањето на прикачувањето се кликнува **Publish**.

Обидете се!

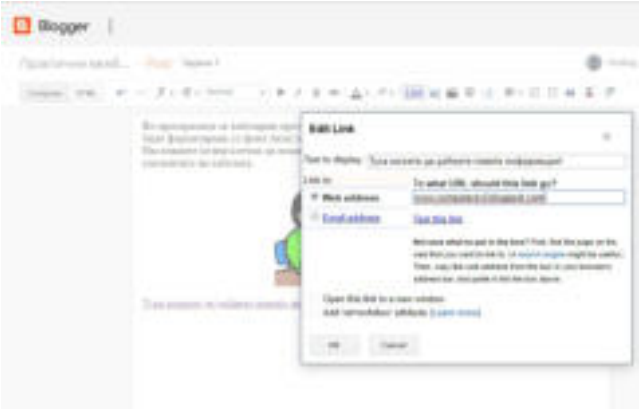
Што се програми? Наведи дефиниции за програмирање! Како се изработува програма? Кои програмски јазици ги познаваш?

5.2.4 Додавање линкови во блог

Да се потсетиме!

Што се линкови? Како може да се препознае дали еден текст е линк? Наведи неколку карактеристики на линковите!

Во објавата (**Post**) на блогот, корисникот може да додаде и линкови, односно да се поврзе со други страници поставени на интернет. За таа цел, се селектира текст во објавата кој треба да биде линк и се кликнува на алатката од прозорецот за додавање линкови:

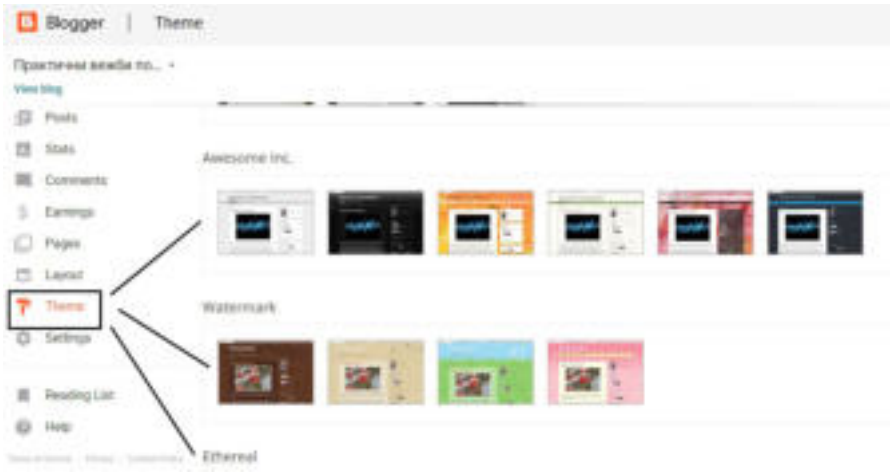


Слика 10: Додавање линкови во објава на блог

Во прозорецот за уредување линк (**Edit Link**), во полето **Text to Display**, се појавува текстот кој е селектиран во објавата за кој корисникот утврдил дека сака да биде линк. Потоа, се селектира копчето **Web Address** и во полето се запишува адресата на страницата до која корисникот сака да се поврзе. Доколку треба да се постави врска до е-маил адреса, тогаш се означува копчето **Email Address** и на крајот се кликнува на копчето **OK**. За да се публикува објавата се кликнува **Publish**.

5.2.5 Менување на темата/изгледот на блогот

Во текот на уредувањето на содржините на блогот, корисникот иако веќе на почетокот избрал тема за блогот, истата може да ја измени. Постапката за менување на темата/изгледот на блогот е преку контролната табла (**Dashboard**) и опцијата **Theme**:



Слика 11 :Избор на тема/изглед на блог

Откако темата/изгледот е избран, се прикажува како би изгледал блогот и доколку изгледот е прифатлив кликнуваме на копчето **Add** за додавање на темата.

На крајот, на контролната табела се кликнува на линкот **View Blog** за да може да се направи преглед на објавите и изгледот на блогот.

Запомни!

Најчесто користен блог-сервис е Blogger и тој му припаѓа на Google. Корисникот мора да има корисничка сметка (account) на Gmail, а тоа подразбира корисничко име и лозинка за најава. Објави или постови се содржините кои се објавуваат на блогот. Преку контролната табла (dashboard) авторот на блогот може да креира поставки, да додава објави, да уредува објави, да менува теми/изглед на блогот, да прави преглед на посетеност на блогот и сл.

Прашања

1.Набројте неколку блог-сервиси!
2.Кој е најпопуларен блог-сервис?
3.Зошто служи контролната табла (dash table) при креирање блог?
4.Што се постови?
5.Која е постапката за креирање пост?
6.Што сè може да биде содржина на блог?

5.3.Коментирање содржини на блоговите

Како што беше кажано на почетокот на темата, блоговите ни овозможуваат да бидеме дел од заедниците со определена тематика. Всушност, можеме да учествуваме во дискусија за која било тема доколку покажеме интерес за проширување на знаењата, споделување на искуствата, изнаоѓање на решение за проблем, приказ на добра пракаса и сл. Поради наведените причини можеме да коментираме на објавите, односно постовите во блогот.

За да можеме да коментираме на содржината на блогот, најпрво треба да ја повикаме содржината на блогот на разгледување со впишување на адресата на блогот во адресната лента на веб-прелистувачот. Притоа, ја разгледуваме неговата содржина. За да додадеме коментар на објава кликнуваме на **Comment**, при што се појавува формулар за додавање на коментар, како што е прикажан на сликата:

Слика 1: Додавање коментар на објава на блог

Во празното поле се пишува текстот на коментарот и се избира профил преку кој ќе се коментира. Потоа има две опции: **Publish** и **Preview**. Доколку се избере **Publish** коментарот ќе биде објавен под содржината на постот, а со **Preview** се прави преглед на коментарот.

Администраторот или креаторот на блогот, за да може да одговори на поставените коментари или, пак, да се вклучи во започнатата дискусија, најпрво прави преглед пристапувајќи преку контролната табла (**Dashboard**) и опцијата **Comment**.

Слика 2: Преглед на коментари на содржини на блог

Со кликнување на коментарот се овозможува опција **Reply** која овозможува администраторот на блогот да даде одговор на коментарот. Во празното поле се пишува одговорот и се кликнува **Publish** за да биде објавен.

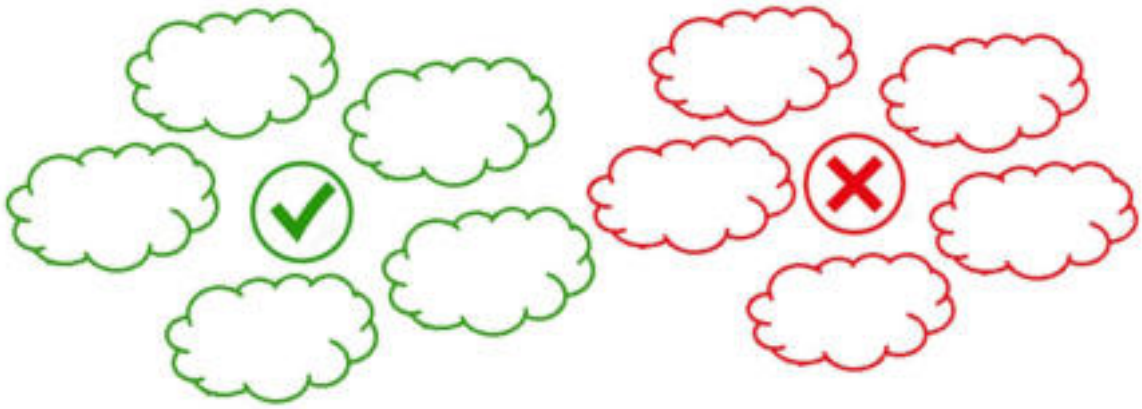
Запомни!

При блогирање и корисиците и администраторот додаваат коментари. Преку коментарите корисниците започнуваат разговор, споделуваат искуства, прашуваат за непознати нешта, одговараат на прашања. Коментарите овозможуваат да се биде дел од заедницата на различна тема.

ДА ПОВТОРИМЕ! ДА УВЕЖБАМЕ!

Задача 1

Пополни го следниот дијаграм така што дозволените правила за етичко користење на сервисите на Интернет да се напишат во зелениот дел, а недозволените во црвениот дел.



Задача 2

Набroј ги советите за оставање на позитивен дигитален отпечаток.

- _____
- _____
- _____
- _____
- _____
- _____

Задача 3

Објасни што претставува веб-дневник!

_____

_____

_____

Задача 4

Дополни ги реченициите за да добиеш точни тврдења.

Луѓето кои ги уредуваат веб-дневниците се нарекуваат ----------
а постапката на уредување на блоговите се нарекува ----------.

Задача 5

Во веб-дневникот вестите/содржините се објавуваат по:

а) според бројот на сликите објавени во блогот;
б) без некој посебен редослед;
в) според хронолошки редослед, односно по датумот на објавата.

Задача 6

Пред да се започне со постапка за изработка на блог,

а) неопходно е корисникот да има е-маил адреса, т.е. електронска пошта;
б) корисникот може да има е-маил адреса, но тоа не е неопходно;
в) не е потребна е-маил адреса.

Задача 7

Корисничкото име на сопственикот (авторот на блогот) се користи како:

а) име за потпис;
б) име за најава во веб-дневникот;
в) тема за дискусија.

Задача 8

Записите на блогот се нарекуваат:

а) статуси;
б) постови;
в) твитови.

Задача 9

Посетителот на блогот, може да остави порака за еден или повеќе постови на блогот преку опцијата за додавање:

а) линк;
б) блог-архива;
в) коментари.

Задача 10

Што е потребно да се направи за да се објави ново напишан запис во блогот?

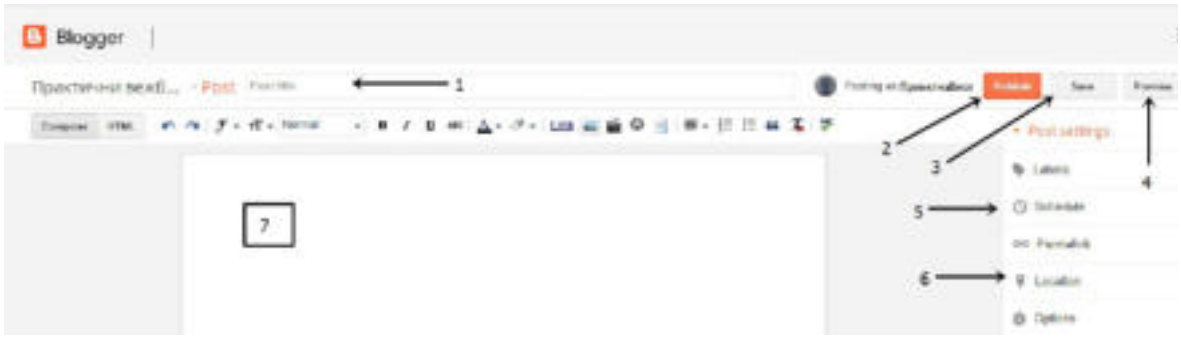
_____

_____

_____

Задача 11

Објасни го значењето на секој елемент!



1._____
2._____
3._____
4._____
5._____
6._____
7._____
8._____

Задача 12

Како треба да се однесуваш при пишување или посета на одредени блогови на Интернет?

_____

_____

_____

Проектна задача

Креирајте веб-дневник/блог на некој од бесплатните сервиси за креирање на блогови. Изберете тема која соодветствува на некоја од проектните активности во училиштето или, пак, кои соодветствуваат на теми од наставните содржини, како на пример: еко-интеграција, меѓуетничка интеграција и интеркултурност во училиштето, спорт и спортски активности, ликовни творби и настани, играње, програмирање и сл. Внимателно изберете тема/изглед на блогот кој соодветствува со содржината на блогот. Блогот да содржи различни типови на содржини: текст, линкови, видеоклипови, слики и сл. Адресите на блоготвите разменете ги со вашите другарчиња за да може да оставате коментари на објавите, потоа, одговорете на следните прашања:

1.Која е адресата на твојот блог?
2.Кој е насловот на твојот блог?
3.За кого е наменет блогот?
4. На која дата е напишан последниот запис во блогот?
5. На која дата е запишан првиот запис во блогот?
6. Освен записи кои други информации ги има на блогот?

РЕЧНИК НА КЛУЧНИ ПОИМИ (стручна терминологија)

Активна ќелија	Ќелијата која во моментот е означена и е подготвена за внесување податоци се вика активна ќелија.	Тема 1
Анализа	Анализа е разложување на проблемската задача на посебни елементи.	Тема 3
Бинарни броеви	Бинарни броеви се 0 и 1. Тие го сочинуваат бинарниот јазик во кој податоците, информациите и командите се преведуваат во низи од 0 и 1.	Тема 2
Блог	Блог или веб-дневник претставува онлајн дневник каде што содржината се прикажува по хронолошки редослед, односно најновите вести или содржини се прикажуваат први, а останатите одат подолу.	Тема 5
Графикон	Графикон претставува графички приказ на податоците.	Тема 1
Графички приказ	Сликовит приказ.	Тема 3
Дебагер	Дебагер е пронаоѓач и отстранувач на грешки (bugs).	Тема 4
Дигитален отпечаток	Дигитален отпечаток се сите интернет информации (позитивни и негативни) кои се случајно или намерно оставени.	Тема 5
Едитор	Уредувач на изворен код е едитор.	Тема 4
Елиминација	Елиминација е процес на исфрлање на фактите кои не се значајни и кои не нè водат кон конечното решение на проблемската ситуација.	Тема 3
Запис	Записи се објавите на блогот.	Тема 5
Изворна програма	Изворна програма е листа на текстуални команди што треба да се преведат во извршна компјутерска програма.	Тема 4
Извршна програма	Извршна програма е датотека која може да се извршува како програма на компјутер.	Тема 4
Иницијализација	Доделување вредност на променливата при нејзино дефинирање се вика иницијализација.	Тема 4

Интерактивни програми	Интерактивни програми се програмите во кои постои некаков вид комуникација (интеракција) помеѓу субјектите.	Тема 3
Искази (наредби)	Исказите претставуваат наредби кои му кажуваат на компјутерот што да направи.	Тема 3
Кодирање	Преведувањето на податоците, информациите и програмите во јазик разбирлив за компјутерот и обратно се вика кодирање.	Тема 2
Колона	Колони се вертикалните делови на табелата.	Тема 1
Коментар	Коментарите се наједноставната алатка за интеракција на блогот, односно за воспоставување комуникација со заедницата.	Тема 5
Компајлер	Компајлер е преведувач на изворниот фајл во извршен.	Тема 4
Константи	Константа е постојана и непроменлива величина.	Тема 3
Координати	Геометриски елементи кои се употребуваат за точно утврдување на положбата на некоја точка.	Тема 3
Криптографија	Криптографијата е научна дисциплина која се занимава со проучување на методите за испраќање и примање пораки на начин кој е разбрилив само на оние за кои е наменета.	Тема 2
Логичко размислување	Логичко размислување е начин за доаѓање до решение преку почитување на вистинитоста на тврдењата, односно исказите.	Тема 3
Настан	Настан во програмски јазик претставува нов градбен елемент кој овозможува интерактивност во програмата преку активности со глувчето или притискање на соодветно копче од тастатурата.	Тема 3
Непоследователни ќелии	Непоследователни ќелии се несоседни ќелии.	Тема 1
Последователни ќелии	Се ќелии кои се наоѓаат една над друга или една покрај друга.	Тема 1
Програмирање	Пишување програми со помош на програмски јазици се вика програмирање.	Тема 2

Променливи	Променливи или варијабли кои се менуваат зависно од внесениот податок.	Тема 3
Работна тетратка	Работни тетратки се документите кои се креираат во програмата за табеларни пресметки. На пример: MS Office Excel, Open Office .org, Apple I work Numbers.	Тема 1
Работен лист	Секоја работна тетратка во програмата за табеларни пресметки се состои од работни листови.	Тема 1
Редица	Редици се хоризонтални делови на табелата.	Тема 1
Редослед	Подредување на податоците согласно со некој услов е редослед.	Тема 1
Слободен софтвер	Слободен софтвер е софтвер што може да се користи, проучува и изменува без ограничувања, што може да се копира и редистрибуира во изменета или неизменета форма без ограничувања или со минимални ограничувања.	Тема 2
Споредбен израз	Споредбен израз е наредба или исказ кој овозможува компарација меѓу вредностите.	Тема 4
Структури на податоци	Структури на податоци се групи податоци дефинирани под едно име.	Тема 2
Табела	Табела е начин на организирање на податоците во колони и редици.	Тема 1
Ќелија/клетка	Пресекот на колона и редица се вика ќелија или клетка.	Тема 1
Формули	Формули се аритметички и логички изрази кои ги креира корисникот со цел да изврши пресметка.	Тема 1
Функции	Функциите се готови формули кои содржат име и аргументи за пресметка.	Тема 1
Циклус	Повторувањето на низа наредби до исполнување на услов е циклус.	Тема 4

КОРИСТЕНА ЛИТЕРАТУРА

Davis, S.(2014). Beginning Programming with C++ for Dummies. Wiley Publishing

Дејтел П, Дејтел Х. (2010). C++ Како се програмира-седмо издание. Арс Ламина

Freun S, Starks J, Schmieder E. (2016). Microsoft Office 365 Excel 2016 comprehensive. University Indianapolis

Gardner S. (2005). Buzz Marketing with Blogs for Dummies. Wiley Publishing

Marji M. (2014). Learn to program with Scratch. William Pollock

www.bebras.org

https://education.microsoft.com

www.bbc.co.uk

www.code.org