ПРОГРАМИРАЊЕ ВО C++

Клучни зборови

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



4.1 Поим за алгоритми и програми

4.1.1 Алгоритми

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

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

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

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

Чекор 3. Притисни го копчето до името на напитокот;

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

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

Чекор 6. Земи ја чашата со напитокот.

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

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

За љубопитните:

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

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

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

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

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

Чекор 1. Внеси (преку тастатура) вредности за влезните податоци: почетната цена S, цената по километар K и бројот на поминатите километри L;

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

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

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

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

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

Карактеристики на алгоритам се:

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

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

Претставување на алгоритми

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

Најчесто се користат следниве два начина:

- *графички приказ*
- *псевдо јазик*

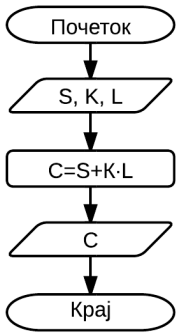
Претставување на алгоритам со графички приказ

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

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

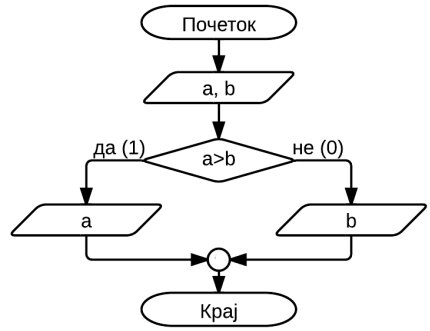
Симболите се цртаат во низа од горе кон долу и меѓусебно се поврзуваат со линии. Оваа насока е логичен редослед на случувањата. Линиите кои означуваат некоја друга насока, на пр. од долу кон горе или наназад, задолжително се означуваат со стрелки.

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



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

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



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

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

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

Претставување на алгоритам со псевдо јазик

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

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

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

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

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

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

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

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

Алгоритамски структури

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

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

Линиска (линеарна) структура

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

Разгранета структура

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

Циклична структура

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

Можни се две ситуации:

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

Резиме

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

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

Прашања:

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

Задачи:

1. Напиши алгоритам за пресметување на поминатот пат на автомобил кој одредено време се движел со одредена средна брзина! Што е влез, а што е излез на овој алгоритам?

2. Алгоритамот од задачата 1 претстави го со блок дијаграм и со псевдо код!
3. Нацртај блок дијаграм и напиши псевдо код за алгоритам со кој ќе се реши линеарната равенка ax+b=0!

4.1.2 Улога на програмите во компјутерот

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

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

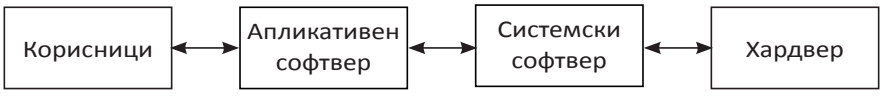
Програма = Алгоритам + Податоци

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

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

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

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



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

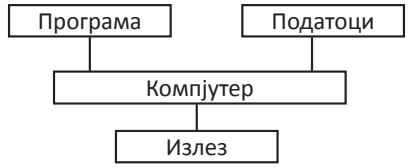
Системските програми ги пишуваат стручните тимови во големите софтверски компании како што се Microsoft, IBM и други, додека апликативните програми може да ги напише и поединец според потребите на корисникот.

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

Поголемиот дел од корисниците на компјутерите користат веќе готови програми кои се инсталирани во компјутерот. Кога програма ќе се покрене од некоја надворешна

меморија, таа се вчитува во оперативната меморија на компјутерот и процесорот започнува со извршување на наредбите од програмата.

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



Сл. 4. 4 Поедноставен приказ на извршување на програмите

Резиме

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

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

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

Податоците претставуваат влез за програма.

Прашања:

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

4.2 Програмирање и програмски јазици

4.2.1 Програмски јазици

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

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

Забелешка:

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

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

- нижи програмски јазици и
- виши програмски јазици.

Главните особини на нижите програмски јазици

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

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

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

Типична наредба во симболички јазик изгледа вака:

```
ADD X Y Z
```

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

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

0110 1001 1010 1011

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

Главни особини на вишите програмски јазици

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

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

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

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

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

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

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

4.2.2 Процес на изработка на една програма

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

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

Креирање на извршна програма

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

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

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

Пишување на изворен код

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

Преведување на изворен код

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

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

Поврзување во извршен код

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

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



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

Тестирање на програма

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

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

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

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

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

L=2*(a+b

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

Но, ако формулата се запише како:

L=2*(a-b)

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

Забелешка:

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

Интерпретери

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

Резиме

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

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

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

Правила со кои се опишува начинот на конструирање на наредби се нарекуваат синтаксата на јазикот.

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

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

Прашања:

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

4.2.3 Интегрирана околина за програмирање

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

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

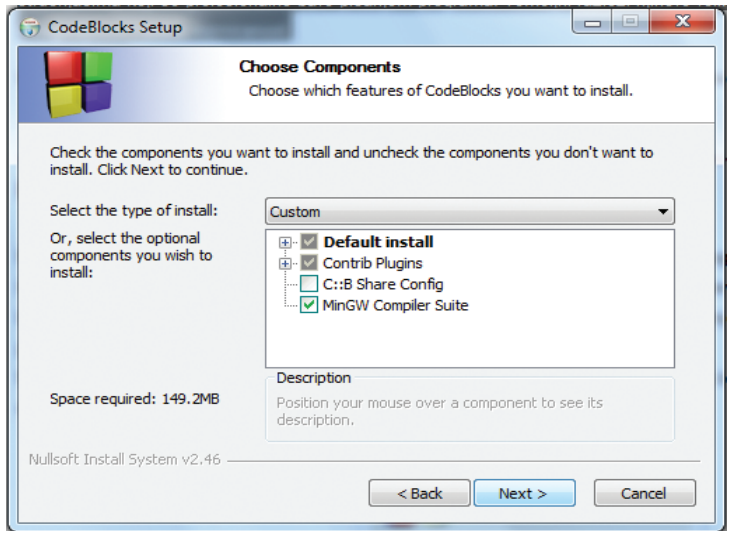
Програмските јазици C и C++ се едни од најпопуларните праграмски јазици за општа намена. Овие јазици имаат големи можности, делотворни се и прилагодливи.

Инсталирање на програмата Code Blocks

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

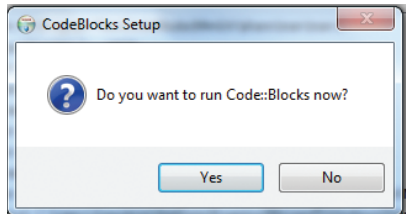
Програмата Code Blocks може да се преземе преку линкот: http://www.codeblocks.org/downloads/binaries/. Во момент на пишувањето на учебников последната верзија на програмата за оперативниот систем Windows е: codeblocks-10.05mingw-setup.exe, а за оперативниот систем Linux: codeblocks-10.05-1-debian-dbg-i386.tar.bz2.

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



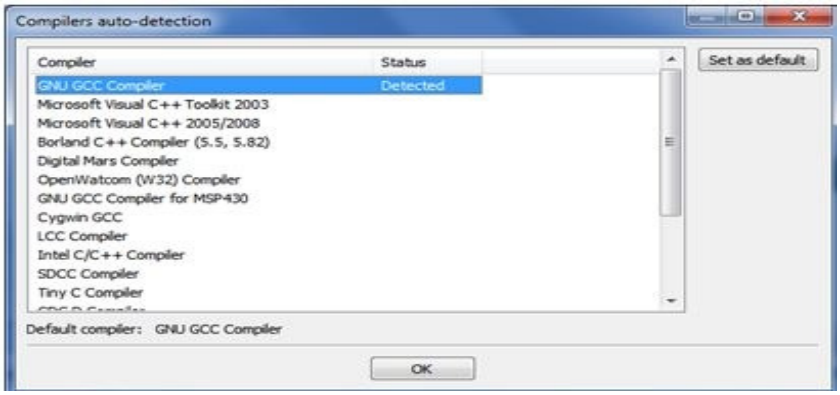
Сл. 4. 6 Инсталација на развојната околина Code Blocks (чекор 1)

Избери програмата веднаш да се стартува:



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

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



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

Прилагодување на работната околина

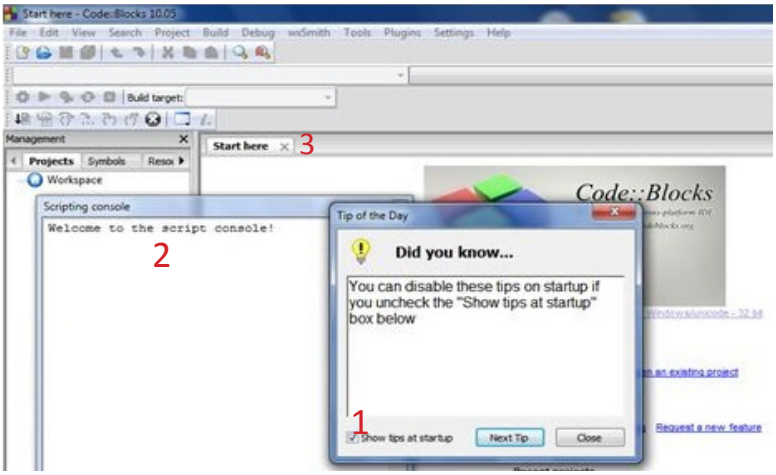
По стартување на програмата, ќе добиеш прозорец во кој, за прилагодување на работната околина:

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

2 Затвори го прозорецот *Scripting Console*;

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

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



Сл. 4. 9 Почетен прозорец на програмата Code Blocks

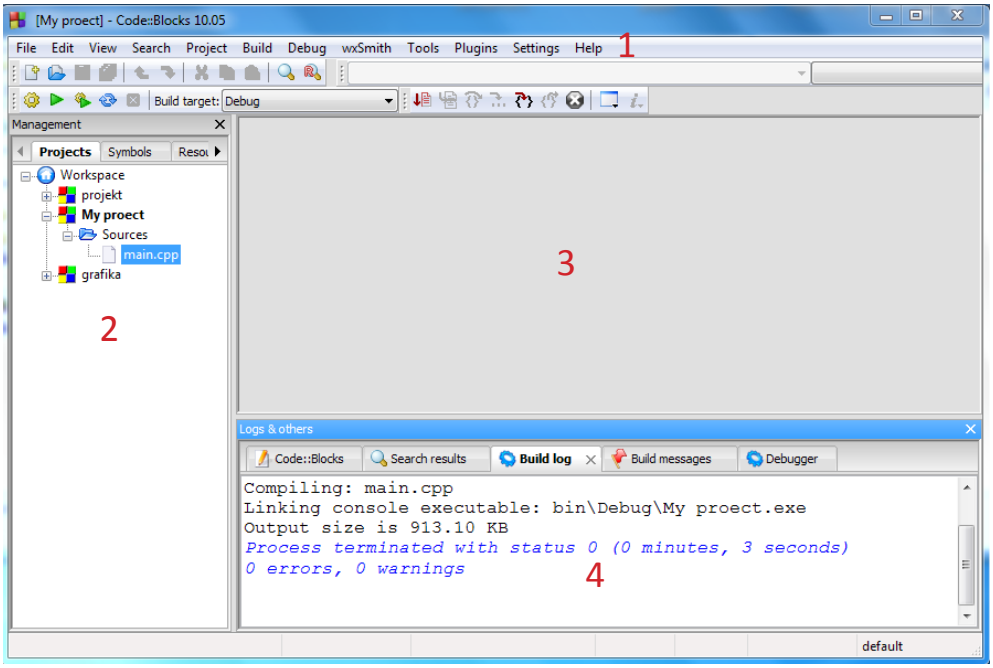
Сега работната околина опфаќа:

1 Мени со наредби,

2 Лентата Management,

3 Текст едитор за внесување на изворниот код,

4 Прозорец за испишување на пораки за грешки.



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

Креирање, преведување и извршување на програма

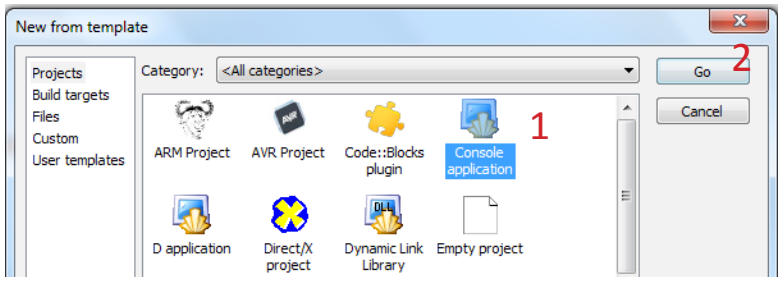
Програмите во C++ можат, но не мора да бидат дел од проект.

Креирање проект

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

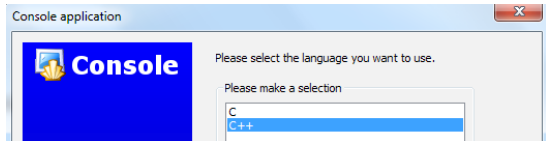
1 Одбери *Console Application* и

2 кликни на копчето *Go*.



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

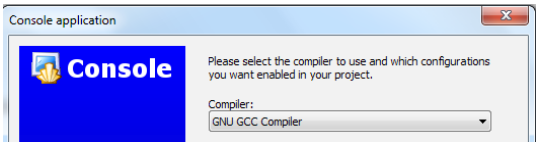
Во следниот прозорец кликни на копчето *Next*, потоа одбери C++ и повторно кликни на копчето *Next*.



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

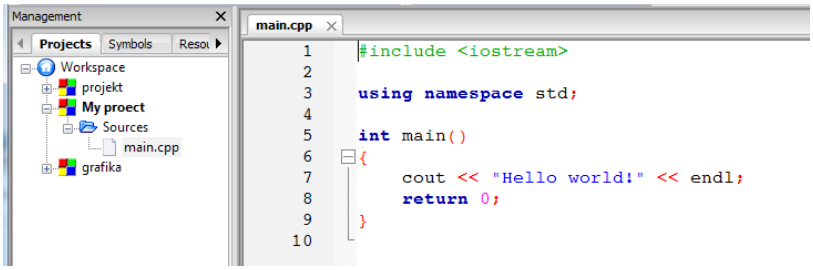
Внеси име на проектот во полето *Project Title*, потоа кликни на копчето *Browse* (означено со три точки) и во прозорецот *Browse for Folder* избери локација каде ќе го зачуваш проектот. Останатите полиња ќе се нагодат автоматски.

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



Сл. 4. 13 Креирање на нов проект (чекор 3)

Кога ќе се креира нов проект тој содржи датотека main.cpp. Во левиот дел на екранот, отвори *Workspace,* потоа твојот проект, па *Sources* и отвори ја датотеката main.cpp (со кликнување двапати на левото копче од глушецот, или со кликнување на десното копче од глушецот и *Open main.cpp*). Тоа е програма која стандардно ја пишуваат сите кои започнуваат со учење на програмирање.



Сл. 4. 14 Датотеката main.cpp

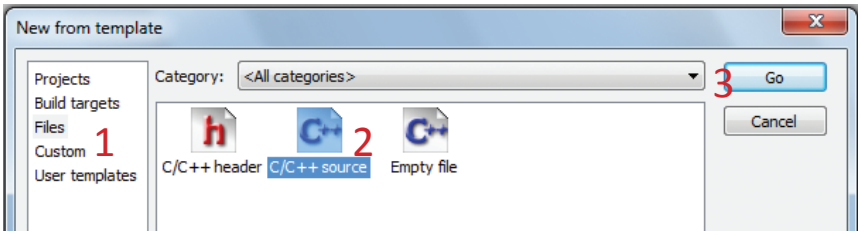
Креирање на нова датотека на изворен код

Нова датотека може да се креира во рамките на постоечки проект или самостојно. Еден од начините за креирање на нова датотека на изворен код е преку наредбата *File→New→File*. Во прозорецот *New from template*

1 избери *Files*,

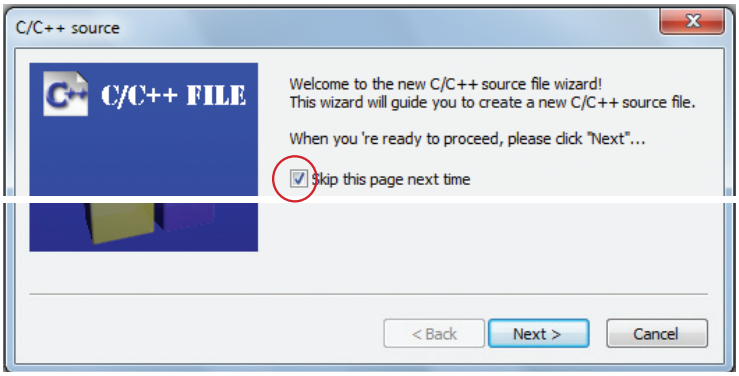
2 потоа избери го типот на датотека *C/C++ source*, и

3 кликни на копчето *Go*.



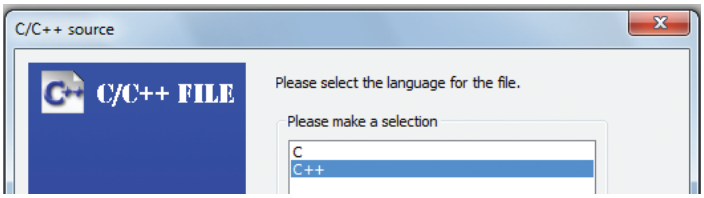
Сл. 4. 15 Креирање на нова датотека на изворен код (чекор 1)

Се покренува помошник за креирање на нова датотека. Доколку не сакаш овој прозорец и понатаму да се појавува потврди ја опцијата *Skip this page next time* и кликни на копчето *Next*.



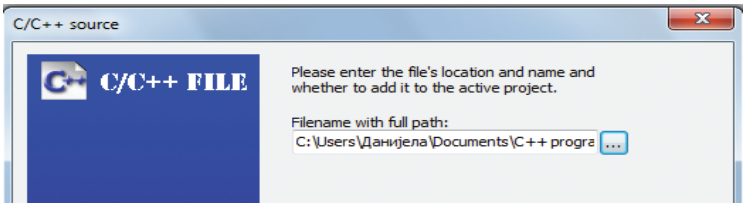
Сл. 4. 16 Креирање на нова датотека на изворен код (чекор 2)

Во следниот прозорец избери го јазикот *C++* и кликни на копчето *Next*.



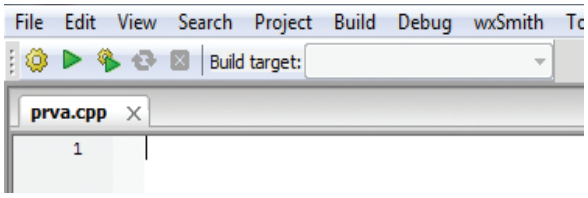
Сл. 4. 17 Креирање на нова датотека на изворен код (чекор 3)

Потоа кликни на копчето *Browse* (означено со три точки) по што ќе се појави прозорецот *Save as...* Избери локација каде ќе ја зачуваш новата датотека и дај ѝ име на датотеката. На крај кликни на копчето *Finish*.



Сл. 4. 18 Креирање на нова датотека на изворен код (чекор 4)

Со ова е креирана нова датотека со името *prva.cpp*:



Сл. 4. 19 Креирање на нова датотека на изворен код (чекор 5)

Совет:

Креирај посебен фолдер во кој ќе ги чуваш твоите програми напишани во јазикот C++.

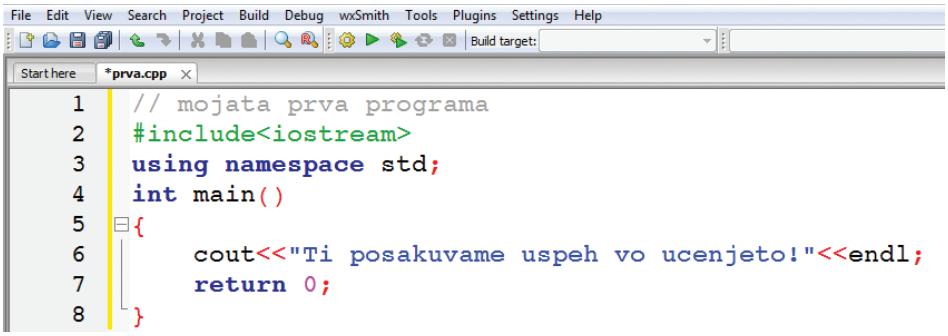
4.2.4 Преведување и извршување на програма

Кога е креирана датотека на изворниот код, може да се почне со пишување на програма. Програмата се пишува во едитор. Препиши ја следнава содржината на датотеката на изворниот код:

Овде е нула

```
// mojata prva programa
#include<iostream>
using namespace std;
int main()
{
    cout<<"Ti posakuvame uspeh vo ucenjeto!"<<endl;
    return 0;
}
```

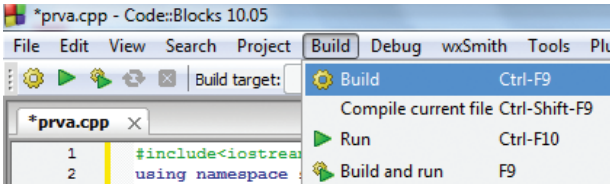
Не грижи се што не ја познаваш структурата на програмата и на поедините наредби. Сега веројатно програмирањето ти изгледа застрашувачки, но набргу сè ќе биде појасно. Изворниот код ќе изгледа како на сликата:



Сл. 4. 20 Изворен код во едиторот на програмата Code Blocks

Линиите се означени со броеви и пред себе имаат жолта вертикална линија. Тоа значи дека тие линии сè уште не се преведени. Преведените лиинии пред себе имаат зелена вертикална линија. Ова е од корист кога се прават измени во програма.

Откако ќе се напише изворна датотека таа треба да се преведе и да се поврзе во извршен код. Тоа се постигнува со наредбата *Build→Build* (Кратенка *Ctrl + F9*).



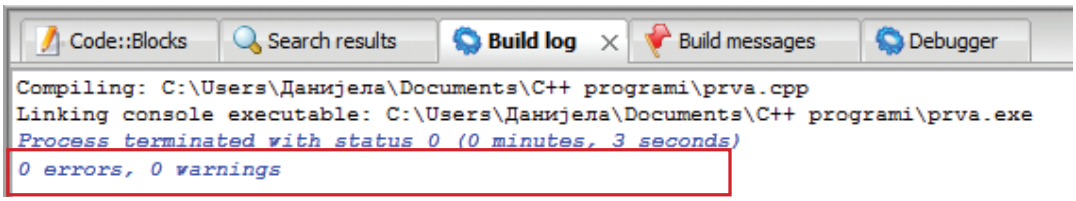
Сл. 4. 21 Наредба за градење на извршна верзија

Забелешка:

Фазите преведување и поврзување се одвиваат како една фаза која се нарекува преведување или компајлирање.

Ако постојат синтаксни грешки или предупредувања, тие ќе се испишат во посебна рамка (*Build log*) во долниот дел на прозорецот. Грешките треба да се исправат

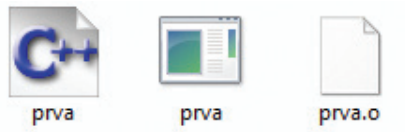
и програмата да се преведе повторно. Кога преведувањето е успешно завршено, преведувачот дава порака како на следната слика:



Сл. 4. 22 Порака за успешно преведена програма

На крајот програмата треба да се изврши со наредбата *Build→Run* (кратенка *Ctrl + F10*).

Како последица на целиот процес, во папката има три датотеки:

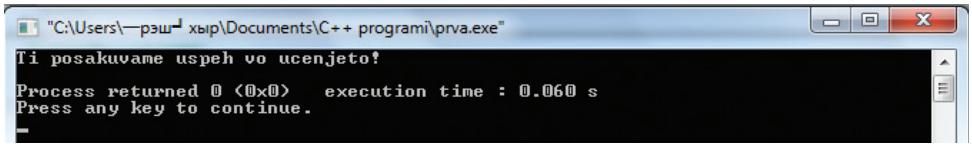


Сл. 4. 23 Икони за изворна, за извршна и за објектна датотека во C++

Важно!

Програмата може да се изврши само ако пред тоа е преведена.

Извршната датотека се извршува во посебен прозорец:



Сл. 4. 24 Прозорец во кој се извршува програма

Нашата програма го испишува текстот „Ti posakuivame uspeh vo ucenjeto!“. Текстот „Press any key to continue“ го генерира самиот компјутер и со тоа не известува дека програмата е извршена и дека со притискање на било кое копче од тастатурата прозорецот ќе се затвори.

Забелешка:

За побрз пристап до наредбите можат да се користат копчињата Build, Run и Build and Run од лентата со алатки за преведување: 

Резиме

Програмите во кои се вградени и обединети програми за пишување на изворен код, програми за преведување, програми за поврзување и програми за наоѓање грешки се нарекуваат интегрирани развојни околини. Ние за програмскиот јазик C++ ќе ја користиме програмата Code Blocks. Кога е креирана датотека на изворниот код, може да се почнеме со пишување на програма. Програма се пишува во едиторот.

Откако ќе се напише, изворната датотека треба да се преведе и да се поврзе во извршен код. Кога преведувањето успешно е завршено, програмата треба да се изврши. Програмата се извршува во посебен прозорец. Како последица на целиот процес, во папката има три датотеки: изворна, објектна и извршна датотека.

Вештини што треба да ги усовршиш:

Да креираш нова датотека.
Да користиш едитор за пишување на изворен код.
Да преведеш изворна датотека во извршна датотека.
Да покренеш извршување на програма.

Прашања:

1. Што е интегрирана развојна околина?
2. За кои развојни околини за програмскиот јазик C++ си слушнал/а?
3. Инсталирај ја програмата Code Blocks и подеси ја работната околина?
4. Со која наредба се креира нова изворна датотека во Code Blocks?
5. Каде се пишува изворен код?
6. Со која наредба се преведува изворна датотека?
7. Кои датотеки се креираат по преведувањето?
8. Како знаеш дека програма е успешно преведена?
9. Со која наредба се извршува програма? Каде програмата се извршува?

Задача:

1. Напиши, преведи и изврши едноставна програма со која ќе се отпечати порака по твој избор! Колку и какви датотеки се креирани во твојата папка?

4.2.5 Извршување и изглед на готови пример програмски кодови

Погледни го кодот на програмата prva.cpp. Дали разбираш некои зборови и реченици? Дали ти е јасно што работи програмата? Дали препознаваш наредба која на компјутерот му кажува да испише соодветен текст на мониторот?

Да го разгледаме и анализираме изворниот код на програмата *prva.cpp*:

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"Ti posakuvame uspeh vo ucenjeto!"<<endl;
    return 0;
}
```

Ова е една од наједноставните програми, но сепак ги содржи основните делови кои секоја програма напишана во C++ мора да ги има.

Изворниот код започнува со следниве линии:

```
#include<iostream>
using namespace std;
int main()
{
```

а завршува со линиите:

```
    return 0;
}
```

Овие линии се пишуваат на почетокот и на крајот на секоја програма. Ќе анализираме линија по линија.

#include <iostream>

Линиите кои започнуваат со знакот „#“ преведувачот не ги преведува и тие се нарекуваат *претпроцесорски наредби*. Овие наредби на преведувачот му кажуваат пред преведувањето да вклучи и некоја додатна датотека. Наредбите во C++ се организирани во посебни датотеки, кои се нарекуваат *библиотеки*. Со изразот `#include` на преведувачот му се најавува од кои библиотеки ќе се користат наредби. Преведувачот тие библиотеки ќе ги вклучи во програмата пред да започне со преведувањето. Името на библиотека се става помеѓу знаците „<>“. Библиотеката `iostream` е стандардна библиотека која овозможува внесување на податоци преку тастатура и испишување на резултати преку монитор.

using namespace std;

Постојат многу библиотеки па може да се случи во две различни библиотеки да се најдат наредби со исто име а со различни намени. Сите елементи на стандардните C++ библиотеки се декларирани во просторот што се нарекува `namespace` (простор со имиња). Со оваа наредба на преведувачот му се кажува дека ќе се користат наредби од просоторот std, што е кратенка од standard.

int main ()

Оваа линија означува дека тука започнува самата програма. Она што е пишувано порано не се смета за програма туку за упатства на преведувачот како да ја преведе програмата, на пр. кои библиотеки да користи при преведувањето.

`main` е име за главната функција која мора да ја има секоја C++ програма. Со оваа функција започнува извршување на сите прогрми во C++ . По зборот `main` стојат загради `()` што значи дека се работи за дефиниција на функција.

{

Голема отворена заграда означува почеток на дел од програма во кој се наоѓаат искази (наредби) на главната функција.

return 0;

Овој исказ е порака до оперативниот систем дека програмата успешно е извршена. По зборот `return` се става повратна вредност, во нашиот случај тоа е нула.

}

Голема затворена заграда се става на крајот на секоја програма и го означува крајот на главната функција.

cout << “Ti posakuvame uspeh vo ucenjeto!“<<endl;

Ова е единствената линија која во нашата програма произведува видлив ефект. Таа во C++ претставува *исказ* или *наредба*. Со овој исказ на мониторот се испишува пораката „Ti posakuvame uspeh vo ucenjeto!“.

Важно!

По секој изказ се става знакот точка и запирка „;“ што е знак дека исказот тука завршува. Претпроцесорските наредби не завршуваат со знакот точка и запирка. Исто така, и по линијата int main () не се става знакот точка и запирка.

Изглед на програма – индентација

Во C++ нема строги правила по кои секој исказ мора да се пишува во нова линија. Така може да се напише и:

```
int main () {cout<<"Ti posakuvame uspeh vo ucenjeto"<<endl; return 0;}
```

За преведувачот оваа линија е исправна бидејќи за крај на исказот се смета знакот „;“, а не крајот на линијата. Исказите се пишуваат во нови линии за програмите да бидат попрегледни. Од тие причини овие искази се малку вовлечени во однос на почетокот и на крајот на функцијата. Ваквата техника на вовлекување на секвенци од искази кои сочинуваат логичка целина се нарекува *индентација*.

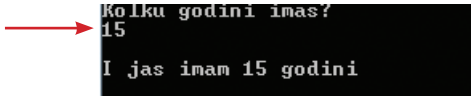
Зад. 4. 5. Следниот код напиши го така да биде полесен за читање. Испиши го на повеќе линии и користи индентација!

```
#include <iostream>
using namespace std;
int main ()
{int x;cout<<"Bravo";return 0;}
```

Пр. 4. 8. Препиши го кодот и зачувај ја датотеката со името *vtora.cpp*:

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      int god;
6      cout<<"Kolku godini imas? ";
7      cin>>god;
8      cout<<"I jas imam "<<god<<" godini"<<endl;
9      return 0;
10 }
```

Преведи ја и изврши ја програмата. На екранот прво ќе ја добиеш пораката „Kolku godini imas?“. Во новиот ред, на местото на курсорот, внеси го бројот на твоите години и притисни го копчето *Enter*. Сега имаш ваков изглед на екранот (црвената стрелка покажува на текстот што корисникот го внесува):



Забележуваш дека во исказот `cout<<"Kolku godini imas?"<<endl;` е додаден изразот `endl` (скратено од end line) што предизвикува поминување на курсорот во нов ред.

Исто така, се јавува и нов исказ:

```
cin>>god;
```

Овој исказ прифаќа податоци внесени преку тастатурата. Во овој случај тоа е бројот 15.

Забелешка:

Изразите cout и cin се декларирани во библиотеката iostream во рамки на просторот std. Тие се причина што мора да се користат првите две линии во програмата.

Ќе разгледаме уште еден исказ кој често се користи во C++ програмите

```
system ("PAUSE");
```

Со овој исказ на компјутерот му се укажува да застане и да чека додека не се притисне некое копче на тастатурата. За да ја разбереш оваа наредба изврши ја програмата *prva.exe* преку иконата од твојата папка; ќе забележиш дека прозорецот во кој се извршува програмата веднаш ќе се затвори пред да ја видиш пораката.

Овој исказ се пишува пред исказот `return 0;` и за негово користење мора да се вклучи билиотеката `cstdlib` преку претпроцесорската наредба `#include <cstdlib>`. Библиотеката cstdlib е библиотека на стандардни наредби.

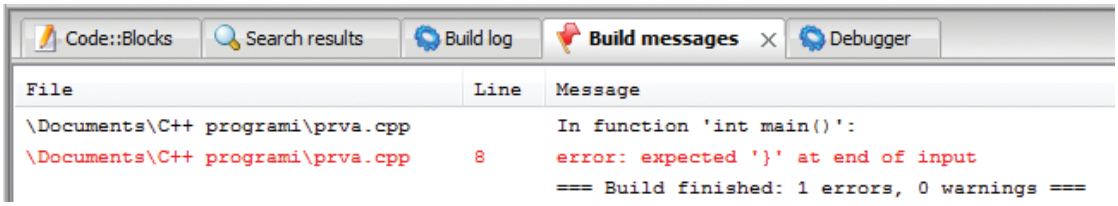
Пронаоѓање и исправање грешки – дебагирање

Во изработка на програми често се јавуваат грешки кои е потребно да се пронајдат и да се исправат. Компјутерските грешки се нарекуваат багови (buggs), па така и процесот на пронаоѓање и на исправање на грешки се нарекува дебагирање (debugging). Овој процес е неопходен во сите нови програми и често е потребно да се повтори повеќе пати. Современите програми се состојат од голем број наредби и често се случува грешките да се откријат дури кај крајните корисници. Од тие причини компаниите кои произведуваат софтвер издаваат пробни (бета) верзии на програмите пред програмата да ја пуштат во продажба.

Да разгледаме што ќе се случи ако во програмскиот код на програмата *prva.cpp* се направи некоја грешка. Напиши го кодот без последната голема заграда:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      cout<<"Ti posakuvame uspeh vo ucenjeto!"<<endl;
7      system("PAUSE");
8      return 0;
9  }
```

При преведувањето, преведувачот ќе воочи дека функцијата `main()` не е правилно затворена и во посебна рамка (Build messages) ќе испише порака за грешка:



Сл. 4. 25 Порака за грешка

Во оваа порака бројот 8 укажува дека грешката се јавила во линијата со реден број 8. Пораката по зборот „error“ ни кажува за каков вид на грешка се работи.

Забелешка:

Во пораки за синтаксна грешка бројот на линија укажува на линија во која преведувачот ја забележал грешката, често тоа е линија по или пред исказот во кој грешката е направена.

Зад. 4. 6. Во програмата *prva.cpp* избриши ја отворената голема заграда и набљудувај што ќе се случи. Истото направи го со малите загради и со наводниците.

Синтаксните грешки можат да ги откријат програмите за преведување за што ќе не известат. Додека не се отстранат сите синтаксни грешки програмата нема да се преведе.

Логичките грешки настануваат како последица на погрешно изработен алгоритам и се причина програмата да не ја извршува задачата точно. Програмерот мора самиот да открие постоење на логички грешки. Компјутерот тука не може да му помогне, бидејќи тој „не знае“ која задача програмерот сака да ја реши.

Забелешка:

За откривање на логичките грешки се користат програми наречени дебагери кои овозможуваат извршување на програмата линија по линија и стопирање на нејзиното извршување на одредено „сомнително“ место.

Пр. 4. 9. Ако програмерот во програмскиот јазик C++ исказот за пресметување на плоштина на круг го напише во следната форма:

```
P=r+r*3.14
```

тој ќе ја добие следната порака за синтаксна грешка: „In function 'int main()' error: expected ';' before...“, бидејќи на крајот на наредбата не го ставил знакот точка и запирка.

Ако програмерот ја исправи грешката и напише:

```
P=r+r*3.14;
```

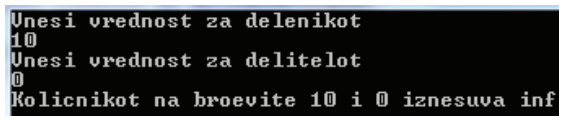
тој нема да добие порака за грешка бидејќи наредбата синтаксно е исправна. Меѓутоа, со оваа наредба плоштината на круг нема точно да се пресмета (формулата за пресметување на плоштина на круг е P=r*r*3.14).

Логичка исправност на програмите се обезбедува со проверки и со тестирање. Програмата воглавно се тестира за влезните податоци за кои однапред е познат резултатот или за кои резултатот може лесно да се провери на друг начин (рачно пресметување). При тоа треба да се води сметка програмата да се тестира за различни податоци.

Пр. 4. 10. Со следнава програма се внесуваат два броја и се пресметува нивниот количник:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    float a,b;
    cout<<"Vnesi vrednost za delenikot"<<endl;
    cin>>a;
    cout<<"Vnesi vrednost za delitelot"<<endl;
    cin>>b;
    cout<<"Kolicnikot na broevite ";
    cout<<a<<" i "<<b<<" iznesuva "<<a/b<<endl;
    system("PAUSE");
    return 0;
}
```

Преведи ја и изврши ја програмата. За деленикот внеси вредност 10, а за делителот внеси вредност 0. Програмата нема да даде точен резултат.



Причина за тоа е што делењето со 0 не е дефинирано и програмата не знае како да се однесува во тој случај. Програмерот направил превид во алгоритамот и не го предвидел овој случај.

Примери за логички грешки се: ако за должината на страна на геометриска фигура се внесе 0 или негативен број, ако за страните на триаголник се внесат должини со кои не може да се формира триаголник, ако за хипотенуза се внесе помала вредност отколку за катета и сл. Ваквите и слични ситуации програмерот мора да ги земе предвид.

Резиме

Линиите кои започнуваат со знакот „#“ преведувачот не ги преведува и нив ги нарекуваме претпроцесорски наредби. Со изразот #include на преведувачот му најавуваме од кои библиотеки ќе користиме наредби. Библиотека iostream е стандардна библиотека која овозможува внесување на податоци преку тастатура и испишување на резултати преку монитор. Со исказот using namespace std; на преведувачот му кажуваме дека ќе користиме наредби од просоторот std, што е кратенка од standard.

main е име на главната функција која мора да ја има секоја C++ програма. Искази кои ја сочинуваат функцијата main се стават помеѓу големите загради „{“ и „}“. Секој исказ завршува со знакот точка и запирка „;“. Исказот return 0; е порака до оперативниот систем дека програмата успешно е извршена.

Техниката на вовлекување на секвенци од искази кои сочинуваат логичка целина се нарекува индентација.

Процесот на пронаоѓање и на исправање на грешки се нарекува дебагирање. Кога преведувачот ќе воочи синтаксна грешка, тој во посебна рамка (Build messages) ќе испише порака за видот на грешката и за линијата во која грешката е воочена. Логичките грешки ги открива програмерот преку тестирање на програмата, во што можат да му помагаат посебни програми наречени дебагери.

Вештини што треба да ги усовршиш:

Да препознаваш структура на програма.
Да употребуваш индентација.
Да внесеш податоци преку тастатурата.
Да препознаваш пораки за синтаксни грешки.

Прашања:

1. Со кои линии започнува, а со кои линии завршува програмскиот код во C++?
2. Како се нарекуваат линиите кои започнуваат со знакот #? Како преведувачот се однесува кон овие линии?
3. За што се користи изразот `#include`?
4. Како се нарекува стандардна библиотека која овозможува внесување на податоци и прикажување резултати?

5. Која е улогата на исказот `using namespace std;`?
6. Како се нарекува главната функција во C++?
7. Како се одвојуваат исказите еден од друг?
8. Како оперативниот систем ќе знае дека програмата успешно завршила со работа?
9. За што се користи исказот `system ("PAUSE");`? Дали е задолжително овој исказ да се користи во програмите?
10. Дали мора секој исказ во програмата да се пишува во нова линија?
11. Што е индентација?
12. Зошто програмерите ја користат техниката на индентација?
13. Што е дебагирање?

Задачи:

1. Пополни ги празните места во следниот код:

```
#include <_________>
_________ <cstdlib>
using _____________ std;
int _____()
{
      cout<<"Zdravo!"___endl;
      ____<<"Kako si?"<<endl;
      system("______");
      return __;
}
```

2. Уреди го следниот код на начин тој да биде прегледен и лесен за читање:

```
            #include <iostream> #include <cstdlib>
      using namespace std;
int main() {int god; cout<<"Hello
            world!"<<endl;
      cout<<"I'm C++"<<endl;
        system("PAUSE"); return 0; }
```

3. Дали во следната програма е направена синтаксна или логичка грешка? Најди ја грешката!

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
      cout<<"Za 100 evra mozes da dobies ";
      cout<<100/61.5<<" denari!"<<endl;
      system("PAUSE");
      return 0;
}
```

4. Најди ги грешките во следнава програма:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
```

```
{
        Cout<<"Kako si?"<<endl;
        cout<<"Mislis deka programiranjeto e tesko?">>endl;
        cout<<"Ne grizi se!;
        cout"Nabrgu ke razberes mnogu poveke";
        system(PAUSE);
        return(0)
}
```

4.3 Програма со редоследна структура

4.3.1 Основни елементи на програмскиот јазик

Градбени делови на програмскиот јазик C++

Програмските јазици се состојат од точно одредена група зборови кои програмерот според одредени правила ги подредува во искази кои му пренесуваат информација на компјутерот.

Програмските јазици, како и природните јазици, имаат своја *азбука*. Азбуката на јазикот C++ е составена од:

- големите и малите букви на англиската абецеда:

  |A | B| C | D| E| F|G|H|I|J|K|L|M|N|0| P|Q| R|S|T|U|V|W|X|Y|Z|

  |a | b | c | d | e| f|g|h| i| j| k|l|m|n|o|p|q|r|s| t|u| v|w|x|y|z|
- арапските цифри:

  |0 |1 |2|3|4|5|6|7|8|9|
- специјалните знаци:

  | + | - | * | / |= | (|) | { \ } \ [|] \ < | > | ‘ | “ | ! | # | \ | % | & | | | | _ |

  | ^ | ¸ | ~ \ , \ ; | : | ?|
- и знакот за празно место | |.

Често се користат и сложени симболи, односно двознаци и тризнаци, на пр.:

|+= | -= | *= | /= | %= | << | >> | ++ | == | /* | */ | && |!= |<<= |>>= |и други.

Со овие знаци се пишуваат сите конструкции од кои програмерот според точно одредени правилата гради искази (наредби). Тие конструкции претставуваат *градбени делови на програмскиот јазик*. Тоа се:

- резервирани зборови
- идентификатори (имиња на величини, функции, операции и сл.)
- оператори (математички, логички, релациони, за доделување)
- интерпункциски знаци (точки, наводници, загради и сл.)
- коментари (произволен текст со кој се објаснува текот на програмата).

Резервирани зборови

Некои од зборовите кои се користат во програмите однапред се дефинирани и преведувачот точно знае што тие значат. Овие зборови се нарекуваат *клучни зборови* и нивното значење не може да се менува. Некои од клучните зборови во јазикот C++ се: delete, int, return, if, then, else, do, while, true, false.

Постојат и предефинирани зборови чије значење исто така е однапред дефинирано али може да се смени (иако е најдобро тоа да не се прави). И едните и другите се *резервирани зборови* и тие не можат да се користат за имиња кои се формираат за да се опишат величини (променливи и константи), функции и слично. Резервираните зборови се пишуваат со мали букви.

Идентификатори

Имињата што ги формира програмерот се нарекуваат кориснички дефинирани зборови или *идентификатори*.

Постојат некои правила при формирање на идентификаторите:

- име се состои од еден или од повеќе знаци кои можат да бидат букви од англиската абецеда (a - z, A - Z), цифри (0 - 9) и знакот долна црта (_) (името не смее да содржи други специјални знаци ниту знакот за празно место),
- првиот знак во името може да биде буква или долната црта,
- се прави разлика помеѓу мали и големи букви (на пр. Zbir, zbir и ZBIR се три различни идентификатори).
- резервираните зборови не смеат да се користат како идентификатори.

Пр. 4. 11. Правилно запишани имиња во C++ се:

```
Broj,
BrojVoDnevnik,
reden_broj,
plostina1,
Ime1_1,
Sobiraj,
```

Пр. 4. 12. Неправилно запишани имиња во C++ се:

`ime i prezime`	содржи празно место
`100procenti`	започнува со цифрата 1
`sto%`	содржи специјален знак %
`ученик`	содржи букви од ќирилицата
`return`	резервиран збор

Совет:

Иако не е пропишано како правило, добра пракса е да се користат имиња кои полесно ќе ги запомниш и ќе имаат некое значење. На пример, подобро е да користиш имиња како што се paralelka, volumen_na_prizma, MestoNaRagjanje и слично наместо a1, a2, a3, xy итн. Ова е особено значајно за снаоѓање во програмите. Ако некој друг, или самиот автор по подолго време, ја чита програмата, полесно ќе разбере за што во неа се работи.

Оператори

Операторите (+, -, *, /, =, %, ++, --, &&, ||, +=,*= и др.) се користат за означување на аритметички, на логички и на други операции кои се изведуваат врз величини (променливи и константи).

Интерпукциски знаци

Интерпукциските знаци (во кои спаѓа и знакот празно место) се користат за раздвојување на елементите на јазикот. На пример, со знакот точка и запирка се разделуваат искази.

Коментари

Програмерите пишуваат коментари за да објаснат што се работи во програмата. На тој начин други програмери полесно се снаоѓаат кога ја читаат програмата, а неретко коментарите им се од корист и на нив самите кога програмата ќе ја читаат по подолго време.

До сега се сретнавме со коментарот:

```
// mojata prva programa
```

Кога преведувачот ќе дојде до двојната коса црта, тој ќе го занемари текстот што следува сè до крајот на таа линија и преведувањето ќе го продолжи во следната линија. По знакот // може да се напише коментар само во една линија. Подолгите коментари (на пр. во заглавието на програмата може да се напише упатство за користење, лиценца и слично) во C++ се пишуваат помеѓу знаците /* и */. На пример:

```
/*   So programava se presmetuva
     perimetarot i plostinata
     na pravoagolen triagolnik */
```

Основни елементи на програмскиот јазик C++

Во табелава се дадени основните елементи на програмскиот јазик C++:

Типови податоци	`int, float, char`
Константи	`0, 123.6, "Hello"`
Променливи	`i, sum`
Декларации	`int x; float a; int fun(int);`
Изрази	`sum + i`
Искази (наредби)	`sum = sum + i; while, for, if-else, switch, break`
Функции	`main(), Hello()`
Модули	`datoteka`

На следната слика е даден пример на програма со која се пресметува плоштина на круг. Веќе кажавме дека програмата Code Blocks различни елементи на програмата ги прикажува со различни бои.

```
1   #include <iostream>
2   #include <cstdlib>
3   using namespace std;
4   int main()
5   {
6
7       float r, L;
8       cout << "Radius r= ";
9       cin>>r;
10      L=2*r*r*3.14;
11      cout<<"Perimetarot L= "<<L<<endl;
12      system ("PAUSE");
13      return 0;
14  }
```

Сл. 4. 26 Изглед на C++ програма

Претпроцесорските наредби се прикажани со зелената боја.

Операторите (<<, >>, *) се прикажани со црвената боја, идентификаторите (main, P, r) и резервираните зборови (cout, cin) се прикажани со црната боја, константите (3.14, 0) се прикажани со виолетовата боја.

Декларациите, за кои подоцна ќе зборуваме, (int, float) се прикажана со темно сината боја.

Функции

Во повеќето програми постојат делови од кои сочинуваат една целина и кои повеќепати се повторуваат. Пишување на тие делови повеќе пати е заморно, па затоа тие се издвојуваат, им се дава име и по потреба се повикуваат во програмата. Деловите од програми кои сочинуваат логичка целина и извршуваат точно одредени задачи се нарекуваат *потпрограми*. Во C++ сите потпрограми се претставени како функции.

Во програмите можат да се користат веќе постоечки функции кои се зачувани во библиотеките или можат да се креираат сопствени функции кои подоцна ќе се користат. Ова се посложени задачи со кои нема да се запознаеш на ова ниво на програмирање. Ќе ја користиш само функцијата main()за која веќе зборувавме.

Функцијата main ()

Тоа е главна функција на секоја C++ програма и означува дел на програма кој треба да се изврши. Таа содржи секвенца од искази (еден или повеќе искази) кои се пишуваат помеѓу големите загради { и }. Секвенците од искази се пишуваат вовлечено заради прегледност.

Структура на програма во C++

Се запознавме со основните елементи кои сочинуваат структура на програма во програмскиот јазик C++. Структурата на една програма изгледа вака:

```
/*
 коментари во заглавието
*/
претпроцесорски наредби
... декларации на променливи
int main()
{
 .... искази (извршни наредби)....
 return 0;
}
```

Резиме

Програмските јазици се состојат од точно одредена група зборови кои програмерот според одредени правила ги подредува во искази.

Градбените делови на програмскиот јазик се: резервирани зборови, идентификатори , оператори, интерпункциски знаци и коментари.

Резервираните зборови не можеме да ги користиме за имиња кои ние ќе ги формираме за да опишеме величини (променливи и константи), функции и слично. Имињата што ги формира програмерот се нарекуваат кориснички дефинирани зборови или идентификатори. Постојат правила при формирање на идентификаторите.

Основните елементи на програмскиот јазик C++ се: типови и класи, константи, променливи, декларации, изрази, искази, функции и модули.

Прашања:

1. Од кои знаци е составена азбуката на програмскиот јазик C++?
2. Наведи ги градбените делови на програмскиот јазик C++!
3. Што се резервирани зборови?
4. Зошто резервираните зборови не можат да се користат за формирање на корисничките имиња?
5. Што се идентификатори?
6. Кои правила мора да се почитуваат при формирање на идентификаторите?
7. Кои од наведените имиња не се идентификатори и зошто?

`tocno?`	`10SoKromid`	`Vozrast`	`noviot_avtomobil`
`a+b`	`AAa`	`x[1]`	`programski jazik`

8. Што се коментари?
9. Дали преведувачот ги преведува коментарите? Зошто?
10. Како се пишуваат коментари во C++?
11. Кои се основни елементи на програмскиот јазик C++?
12. Која функција мора да ја има секоја програма во програмскиот јазик C++?
13. Која е улогата на наредбата `include`?
14. Колку `main` функции може да има во програма?

4.3.2 Искази. Исказ за приказ на екран

Во програмата prva.cpp го видовме исказот

```
cout << "Ti posakuvame uspeh vo ucenjeto!";
```

со кој текстот „Ti posakuvame uspeh vo ucenjeto!“ се прикажува на екранот.

Да ги анализираме составните делови на овој исказ:

- `cout` – функција која насочува податоци кон излезот, најчесто тоа е монитор,
- `<<` – операторот на испишување кој пораката ”Ti posakuvame uspeh vo ucenjeto!” ја праќа кон излезот,
- ” ” – се што е ставено во наводици е порака која се прикажува на екранот,
- `;` – означува крај на исказ (не на линија).

Со користење на исказот за приказ, на мониторот може да се прикажат (отпечатат) какви било податоци. Мониторот е стандардна излезна единица.

Примери за примена на исказот за приказ:

`cout<<25;`	ќе се прикаже бројот 25
`cout<<"a";`	ќе се прикаже вредност буквата a
`cout<<a;`	ќе се прикаже вредност на променливата a
`cout<<2*a;`	ќе се прикаже двојната вредност на променливата a
`cout<<"Tekst";`	ќе се прикаже текстот „Tekst“

Пр. 4. 13. Изврши ја следната програма!

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      cout<<"Zdravo!";
7      cout<<"Jas sum C++ programa.";
8      system("PAUSE");
9      return 0;
10 }
```

```
Zdravo!Jas sum C++ programa!Press any key to continue . . .
```

Забележуваш дека речениците „Zdravo!“ и „Jas sum C++ programa!“ се прикажани во една линија и се прилепени една со друга. Како ќе ги разделиш со едно празно место?

По испишување на податоци наведени во исказот `cout`, курсорот останува во истата линија и следното испишување на податоци продолжува тука.

Пр. 4. 14. За испишување во нова линија, се користи изразот `endl` (кратенка од end line) со кој курсорот се поместува за една линија подолу.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      cout<<"Zdravo!"<<endl<<"Jas sum C++ programa.";
7      system("PAUSE");
8      return 0;
9  }
```

```
Zdravo!
Jas sum C++ programa!Press any key to continue . . .
```

Во исказот `cout<<”Zdravo!”<<endl<<”Jas sum C++ programa.”;` се наведени повеќе оператори на испишување. Во тој случај податоците се испишуваат од лево кон десно.

Во општ случај исказот cout се користи на следниот начин:

```
cout<<podatoci[<<podatoci];
```

при што податоците можат да бидат константи, променливи, изрази или нивната комбинација.

Забелешка:

Деловите на искази кои не се задолжителни ќе ги ставаме во средни загради.

Техника на редоследно извршување

Во главната функција има два искази за приказ на екран:

```
cout<<"Zdravo!";
cout<<"Jas sum C++ programa.";
```

Кој исказ ќе се изврши прв? Што ќе се случи ако на исказите им се заменат местата?

Пр. 4. 15. Програмата од примерот 5.14 може да се напише и вака:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"Zdravo!"<<endl;
    cout<<"Jas sum C++ ";
    cout<<"programa.";
    system("PAUSE");
    return 0;
}
```

```
Zdravo!
Jas sum C++ programa!Press any key to continue . . .
```

Во овој пример има повеќе искази за печатење на екран наредени еден по друг. Сите овие искази ќе се извршат точно по оној редослед по кој се напишани.

Вака напишани искази се нарекуваат *секвенца од искази* или *напластени искази*. Пишувањето на искази во програмскиот код кои се извршуваат сите и редоследно се вика *техника на редоследно извршување*.

Пр. 4. 16. Отвори нова датотека и дај ѝ име C++. Препиши го кодот:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"    **** "<<endl;
    cout<<"  ******       **         **"<<endl;
    cout<<" ***   **      **         **"<<endl;
    cout<<" **            **         **"<<endl;
    cout<<" **          ********  ********"<<endl;
    cout<<" **          ********  ********"<<endl;
    cout<<" **            **         **"<<endl;
    cout<<" ***   **      **         **"<<endl;
    cout<<"  ******       **         **"<<endl;
    cout<<"    ****"<<endl;
    system("PAUSE");
    return 0;
}
```

Преведи ја и изврши ја програмата!

Зад. 4. 7. Напиши и изврши програма со која на екранот ќе се прикаже твоето име напишано со помош на ѕвездички!

Резиме

Исказ или наредба одредува кои дејства е потребно да се извршат за да добиеме резултат. Секој изказ мора да завршии со знакот точка и запирка.

За приказ на екран го користиме исказот cout. Во општ случај исказот cout го користиме на следниот начин: `cout<<podatoci[<<podatoci];` при што податоците можат да бидат константи, променливи, изрази или нивната комбинација.

`cout` е функција која насочува податоци кон излез, најчесто тоа е монитор, << е оператор на испишување кој пораките ставени во наводници ги праќа кон излезот.

Искази кои се извршуваат точно по оној редослед по кој се напишани претставуваат секвенца од искази или напластени искази. Пишувањето искази во програмскиот код кои се извршуваат сите и редоследно се вика техника на редоследно извршување.

Вештини што треба да ги усовршиш:

Да познаваш структура на исказот за прикажување на екран (cout).

Правилно да го употребуваш исказот cout и операторот <<.

Да го користиш изразот endl во исказот за прикажување на екран.

Да напишеш едноставна програма со користење на исказите cout и техника со напластени искази.

Прашања:

1. Што е исказ?
2. Кој исказ се користи за приказ на екран?
3. Во исказот за приказ на екранот, што означува операторот << ?
4. Во исказот за приказ на екранот, кој израз се користи за да курсорот помине во нов ред?
5. Што се означува со изразот напластени искази?
6. Како се нарекува техниката на пишување искази кои се извршуваат сите и редоследно?
7. а) Што ќе се прикаже на екранот со следниов исказ?

```
cout<<"J"<<endl<<"A"<<endl<<"N"<<endl<<"A"<<endl;
```

б) Што ќе се прикаже на екранот со следнава секвенца од искази?

```
cout<<"J"<<endl;
cout<<"A"<<endl;
cout<<"N"<<endl;
cout<<"A"<<endl;
```

Задачи:

1. Напиши програма со која ќе го добиеш следниов приказ на екранот:

```
Zdravo!
Kakov prekrasen den!
Odam na prosetka.
```

2. Напиши програма со која ќе се отпечати твоето име на следниов начин:

   I
   M
   E

3. Напиши програма со која ќе се отпечати твоето име на следниов начин:

   I
    M
     E

4. Напиши програма со која на мониторот ќе се прикаже слика по твој избор со помош на ѕвездички!

4.4 Променливи и искази за доделување

4.4.1 Аритметички операции и изрази

Пр. 4. 17. Напиши ја и изврши ја следнава програма:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       cout<<"5+2"<<endl;
7       system("PAUSE");
8       return 0;
9  }
```

```
5+2
Press any key to continue . . .
```

На екранот се прикажува „5+2“. Овде изразот „5+2” е низа од знаци која се прикажува на истиот начин како што таа е напишана во наводниците.

Напиши ја истата програма, но сега тргни ги наводниците!

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       cout<<5+2<<endl;
7       system("PAUSE");
8       return 0;
9  }
```

```
7
Press any key to continue . . .
```

Програмата ги собира броевите 5 и 2 и резултатот, кој во случајов е 7, го прикажува на екранот. Изразот 5+2 претставува аритметички израз кој има своја бројна вредност.

Зад. 4. 8. Напиши програма со која на екранот ќе се прикаже 5 + 2 = 7!

Аритметички израз е бројна вредност или запис од две или на повеќе бројни вредности и математички оператор(и) меѓу нив. *Оператори* во програмскиот јазик C++ се: плус (+) за операцијата собирање, минус (-) за операцијата одземање, ѕвездичка (*)

за операцијата множење, коса црта (/) за операцијата делење и модул (%) за остаток при делење на два цели броја. Ова се само некои од операторите кои се користат во програмскиот јазик C++.

Аритметичките изрази се градат на ист начин како и во математиката. Примери за аритметичките изрази се: `2*3+8`, `7-2`, `(15-5)*100`, `2/a`, `a-b`. Овде броевите како што се 2, 3, 100 итн. се константи, додека `a` и `b` се променливи.

Променливи и константи

Со програмите се обработуваат податоци и тие претставуваат *величини* кои можат да бидат:

- константи – величини кои не ја менуваат својата вредност,
- променливи – величини кои ја менуваат својата вредност.

Доделување на вредност на променлива. Оператор за доделување

Во компјутерот за секоја променлива се предвидува и се резервира место во меморијата. Секое резервирано место има своја адреса која е тешка за памтење, па од тие причини на променливите им се доделуваат *симболички имиња*. При тоа, мора да се води сметка да се почитуваат правилата кои важат за сите идентификатори.

Во математиката и во другите науки се среќаваш со многу променливи, на пр. a, b, c – страни на триаголник, P – плоштина, t – температура итн. На сите овие променливи им се доделуваат различни вредности. На пример, на температурата може да ѝ се додели вредност -15, али и 23, 17, 32 итн.

На променливите им се доделува вредност со помош на *оператор за доделување*. Операторот за доделување е знакот еднакво (=).

Важно!

*Знакот еднакво (=) овде не означува еднаквост како во математиката. Со оператор за доделување на **променливата од левата страна** на операторот ѝ се **доделува вредност на изразот од десната страна** на операторот. Од тие причини променливата на која ѝ се доделува вредност мора да биде од левата страна на операторот за доделување. Неточно е на пр. 5 = a или a+b = c;*

Примери за доделување вредности на променливи се дадени во следнава табела:

`x = 5;`	на променливата `x` ѝ се доделува вредност 5. Ова всушност, значи дека во мемориската локација која е означена со симболичко име `x` ќе се зачува бројот 5 или дека бројот 5 е содржина на променливата `x`.
`a = 20;`	на променливата `a` ѝ се доделува вредност 20.
`a = b*3;`	помножи ги содржината на променливата `b` и бројот 3 и производот зачувај го во променливата `a`.
`x = x+3;`	на содржината на променливата x додај број 3 и збирот зачувај го во променливата `x`.
`x=c=b=5;`	на променливата `b` ѝ се доделува вредност 5, на променливата `c` ѝ се доделува содржината на променливата `b`, на променливата `x` ѝ се доделува содржината на променливата `c`. Конечен резултат на овој исказ е: содржината на сите три променливи е иста и е еднаква на бројот 5.

Да ја разгледаме следнава секвенца од искази:

Исказ	Содржина на променливата		
	a	b	zbir
a = 3;	3	?	?
b = 7;	3	7	?
a = a+1;	4	7	?
zbir = a+b;	4	7	11

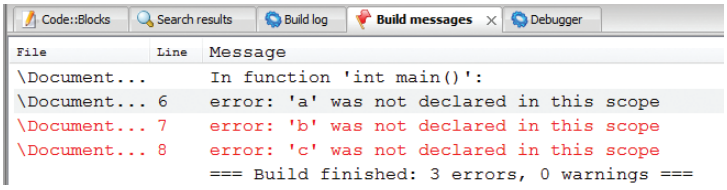
По извршување на оваа секвенца од искази променливата a ќе има вредност 4, променливата b ќе има вредност 7 и променливата zbir ќе има вредност 11.

Посебно внимание ќе му обрнеме на исказот a=a+1; Во математиката ваков исказ не е точен, но во програмскиот јазик C++ знакот = нема исто значење како во математиката. Пред овој исказ, на променливата a ѝ е доделена вредност 3 (со исказот a=3;). Со исказот a=a+1; вредноста на променливата a (бројот 3) се собира со 1 и збирот (бројот 4) се доделува на променливата a. Тоа значи дека во мемориската локација која е означена како променлива a претходната содржина (бројот 3) ќе се избрише, а на нејзиното место ќе се зачува новата содржина (бројот 4).

Сè уште не можеме да напишеме програма во која ќе се користат променливи. Обиди се да ја извршиш следнава програма:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      a = 2;
7      b = 5;
8      c = a+b;
9      cout<<c<<endl;
10     system("PAUSE");
11     return 0;
12 }
```

Преведувачот ќе јави грешки дека променливите кои се користат во програмата не се декларирани:



Очигледно е дека преведувачот бара да се декларираат (најават) променливи кои ќе се користат во програмата. За ова да може да се направи, прво ќе се запознаеме со поимот тип на променлива.

4.4.2 Типови на променливи

Компјутерот мора однапред да знае какви податоци ќе се чуваат во променливите. Причината е едноставна – потребно е повеќе место за зачувување на на пр. една реченица отколку за зачувување на еден знак. Значи, компјутерот треба однапред да знае колкав простор во меморијата ќе резервира за секоја променлива.

Освен тоа, компјутерот не се однесува исто со сите податоци. На пример, со броевите може да извршува аритметичките операции, додека со знаци тоа не може да го прави. Од овие причини за секоја променлива мора да се знае *тип на податокот* кој во неа ќе биде зачуван. Се разликуваат основни и други типови податоци. Основните типови податоци се:

- *цели броеви* – броеви кои немаат децимален дел,
- *реални броеви* – броеви кои имаат цел и децимален дел,
- *знаци* – било кој знак од азбуката на C++,
- *логички податоци* – точно или неточно, односно 0 или 1.

Типот на променлива е одреден со типот на податокот кој во неа може да се зачува. На секој променлива, освен симболичко име, мора да ѝ се додели и *ознака за тип* кој ни кажува колку простор во меморијата треба да се резервира, колкав е опсегот на дозволените вредности на податокот, кои операции се можни со тој податок и слично.

Во табелата се дадени дел од основните типови податоци и нивните ознаки, како и вредностите кои можат да се доделат на променлива од соодветен тип:

Ознака	Опис	Големина	Опсег на вредности
`char`	знаковна променлива	1 бајт	256 знаци
`int`	целобројна променлива	4 бајти	0 - 4294967295 или од -2147483648 до 2147483647
`bool`	логичка променлива	1 бајт	true или false
`float`	реална променлива со	4 бајти	со точност до 7 децимални места
`double`	реална променлива со двојна прецизност	8 бајти	со точност до 15 децимални места

Целобројните типови можат да бидат само позитивни или и позитивни и негативни. Ако во некоја целобројна променлива треба да се зачуваат само позитивни вредности, тогаш пред ознаката `int` се става ознаката `unsigned`. Постои и ознака `signed`, но таа се подразбира, па не мора да се пишува.

Ќе спомнеме уште еден тип податоци кој не се вбројува во основните типови, но во многу работи така се однесува така. Тоа е низа од знаци (карактери) или *стринг* со која веќе се запозна (“Dobar den”, “C++” итн.). За сега е доволно да знаеш дека низа од знаци се пишува помеѓу наводници. Ознака за овој тип податоци е string.

За љубопитните:

Комплетната табела со сите основни типови податоци можеш да ја погледнеш во додатокот А, на крајот на учебникот.

Декларирање на променливи

Секоја променлива која ќе се користи во програма мора да се најави, односно да се *декларира*. При тоа треба да се има предвид следниве карактеристики:

- секоја променлива има свое име
- секоја променлива има тип
- секоја променлива содржи вредност која ѝ се доделува најчесто со операторот за доделување.

Кога се декларира некоја променлива, прво се наведува ознака за тип, а потоа име на променливата:

```
tip imenapromenliva [imenapromenliva];
```

По еден тип може да се наведат и повеќе променливи одвоени со запирка. Така со исказот `int i, j;` се најавува дека ќе се користат две целобројни променливи, нивните имиња се `i` и `j`, на нив може да им се додели вредност која ќе биде цел број.

Примери на декларирање на променливи:

`int a;`	декларација на целобројна променлива a
`float broj;`	декларација на реална променлива broj
`char bukva;`	декларација на знаковна променлива bukva
`int b,c,d;`	декларација на 3 целобројни променливи b, c и d

Пр. 4. 18. Сега имаме сè што е потребно за да се напише програма со која ќе се соберат вредности на две променливи. Пред да се користат променливи, тие треба да се декларираат:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b, c; // deklariranje na promenlivi
    a = 2;
    b = 5;
    c = a+b;
    cout<<c<<endl;
    system("PAUSE");
    return 0;
}
```

```
7
Press any key to continu
```

Наместо исказот `cout<<c<<endl;`

подобро е да се напише `cout<<"c="<<c<<endl;`

со што на екранот ќе се отпечати „c=7“. На тој начин корисникот знае на што се однесува резултатот што е прикажан на екранот.

Истиот излез ќе се прикаже и ако се напише `cout<<"c="<<a+b<<endl;` Во овој случај може да се изостави исказот `c=a+b;`. Зошто?

Пр. 4. 19. Со следната програма ќе се отпечати „c“ наместо 7. Зошто?

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b, c;
    a = 2;
    b = 5;
    c = a+b;
    cout<<"c"<<endl;
    system("PAUSE");
    return 0;
}
```

```
c
Press any key to continue . . .
```

Треба да се внимава кога се доделуваат вредности на променливите. Преведувачот нема да не извести ако на некоја променлива ѝ се доделува податок од несоодветен тип, програмата ќе се преведе но нема да даде точен резултат. Да го погледнеме следниот пример.

Пр. 4. 20. На целобројната променлива a ѝ се доделува реален број:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       int a, b, c;
7       a = 2.5;
8       b = 5;
9       c = a+b;
10      cout<<"c = "<<c<<endl;
11      system("PAUSE");
12      return 0;
13 }
```

```
c = 7
Press any key to continue . . .
```

Во овој пример со изразот `a=2.5;` на променливата a нема да ѝ се додели реален број 2.5. Таа е декларирана како целобројна променлива па во неа се зачувува само целиот дел од доделената вредност (2).

Зад. 4. 9. Во претходниот пример на променливата a додели ѝ податок од тип знак, на пример `a='2';` (знаците се пишуваат помеѓу апострофи). Преведи ја и изврши ја програмата! Што се случува?

Иницијализација на променливи

Пред да се започне со користење со користење на некоја променлива, неопходно е таа да има некоја вредност. Доделување на почетната вредност на променливите се нарекува *иницијализација на променливи.*

На променливите може да им се доделат вредност во истиот исказ со кој тие се декларираат. Така наместо:

```
int a;
a = 5;
```

може да се напише

```
int a = 5;
```

Пр. 4. 21. Плоштина на правоаголникот со страните a = 8 и b = 3:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6       // inicijalizacija na promenliv
7       float a = 8;
8       float b = 3;
9       float P;
10      // dodeluvanje na vrednost na P
11      P = a*b;
12      cout<<"P = "<<P<<endl;
13      system("PAUSE");
14      return 0;
15 }
```

```
P = 24
Press any key to continue
```

Пр. 4. 22. Замена на вредности на две променливи:

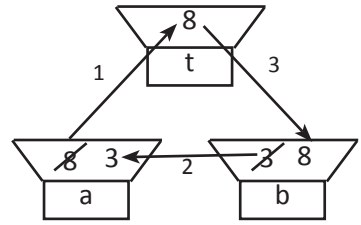
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    // inicijalizacija na promenlivi
    int a = 8;
    int b = 3;
    int t;
    // dodeluvanje na vrednosti
    t = a;
    a = b;
    b = t;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl;
    system("PAUSE");
    return 0;
}
```

```
a = 3
b = 8
Press any key to continue .
```

Замена на вредности на две променливи често се користи во програмите.

Променливата `t` се користи како привремена променлива во која ќе се зачува вредноста на променливата `a` пред да ѝ се додели нова вредност со исказот `a=b;`. Со исказот `b=t;` на променливата `b` ѝ се доделува вредноста од привремената променлива `t` што всушност е претходната вредност на променливата `a`.

	a	b	t
int a = 8;	8	?	?
int b = 3;	8	3	?
t = a;	8	3	8
a = b;	3	3	8
b = t;	3	8	8



Сл. 4. 27 Графички приказ на замена на вредности на променливи

Важно!

Ако на променливите не им се додели некоја вредност тие сепак ќе имаат некоја случајна вредност која преведувачот автоматски ќе и додели. Линиите a = 5; и b = 2; стави ги како коментари, преведи ја и изврши ја програмата. Ќе видиш дека програмата ќе работи поинаку, односно ќе даде некој произволен резултат. ***Секогаш доделувај вредности на променливи!***

Константи

Освен променливите, во C++ се користат и константи. Со константите во програмите се изразуваат одредени вредности. Како и променливите, и константите се разликуваат според типот на податок со кој нејзината вредност е изразена. Така има:

- *целобројни константи* (15, 100, -50)
- *реални константи* (2.34, 105.5, -12.2)
- *знаковни константи* (‘a’, ‘M’, ‘+’, ‘$’)
- *логички константи* (true/false, 1/0)
- *стринговни константи* (“Dobar den”, “C++”).

Забележуваш дека знаковните константи се запишуваат помеѓу апострофи, додека стринговните константи се запишуваат помеѓу наводници.

И константите можат да се декларираат на сличен начин како и променливите. Единствената разлика е во тоа што се користи клучниот збор const. На пр.:

```
const float DDV = 0.18;
const float BRZINA_NA_SVETLINATA = 2.997925e8;
const char ZNAK='$';
```

Реалната константа DDV е прикажана со децимална точка, а константата `BRZINA_NA_SVETLINATA`, која исто така е реална, е прикажана со т. н. научно означување. Во овој начин на означување се користи малата или големата буква „e“ со која се означува експонентот на бројот `10`. Со `2.997925e8` е претставен бројот $2.997925 \cdot 10^8$.

Забелешка:

Програмерите имињата на константите вообичаено ги пишуваат со големи букви иако такво синтаксно правило во јазикот C++ не постои.

Пр. 4. 23. Следнава програма пресметува плоштината на круг со радиусот r = 2.5:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      const float PI = 3.14;
7      float r = 2.5;
8      float P;
9      P = PI*r*r;
10     cout<<"P = "<<P<<endl;
11     system ("PAUSE");
12     return 0;
13 }
```

```
P = 19.625
Press any key to continue . . .
```

Во програмава се користи реалната константата `PI` на која ѝ се доделува вредност 3.14.

Резиме

Аритметички израз е бројна вредност или запис од две или повеќе бројни вредности и математички оператор(и) меѓу нив. Операторите во програмскиот јазик C++ се: + - * /и %. Секоја променлива или константа која ќе се користи во програма мора да се најави, односно декларира. Под ова се подразбира одредување на имети и на типот на променливата.

На променливите им се доделува вредност со помош на операторот за доделување (=). Ако на променливите не им доделиме вредност, тие сепак ќе имаат некоја случајна вредност која преведувачот автоматски ќе ѝ ја додели.

Основни типови и ознаки на променливите и на константите се: целобројни, реални, знаковни и логички променливи.

Доделување на почетната вредност на променлива се нарекува иницијализација на променлива.

Вештини што треба да ги усовршиш:

Да пишуваш и да користиш аритметички изрази во C++.

Да доделуваш вредности на променливи со помош на операторот =.

Да препознаваш типови на променливи и правилно да ги декларираш.
Да напишеш едноставна програма со доделување вредност на променлива преку едноставни математички изрази.

Прашања:

1. Што е аритметички израз? Напиши неколку аритметички изрази?
2. Кои величини се користат во програмските јазици?
3. Направи разлика помеѓу константи и променливи?
4. Што се подразбира под декларација на променлива?
5. Дали е можно повеќе променливи да се декларираат со еден исказ? Ако е, наведи пример!
6. Која од наведените декларации е исправна?
   а) `int n=-10;`
   б) `unsigned int i =-10;`
   в) `signed int = 3.8;`
7. Наведи примери за секој од типовите на константи!
8. Што ќе се прикаже на екранот со следниот програмски код?
```
char a,b,c;
a='b';
b='c';
c=a;
cout<<a<<b<<c;
```
9. Што ќе се прикаже на екранот со следниот програмски код?
```
bool x=2>1;
cout<<x;
```

Задачи:

1. Ако во програма се користат променливите int kolicina=24 и float cena=12.5, со наредбата cout испиши ги:
   а. нивните имиња
   б. нивните вредности
   в. извештај во форма „kolicina = x, cena = y“, така што наместо x и y се испишуваат вредностите на променливите kolicina и cena.
2. Напиши програма за пресметување на плоштина на правоаголник со страните a=8.5 и b = 3.2! Излезот да се прикаже на екранот во следнава форма:
   `Plostinata na pravoagolnikot so stranite a=8.5 i b=3.2 iznesuva 23.4.`
3. Напиши програма за пресметување на плоштина на круг со радиус r = 2.5! Излезот да се прикаже на екранот во следнава форма:
   `Plostinata na krugot so radiusot r = 2.5 iznesuva 19.625.`
4. Напиши програма со која ќе пресметаш плоштина и периметар на квадрат со страна a = 8.5!
5. Напиши програма со која ќе се најде аголот на врвот на рамностран триаголник чиј агол на основата е 30 степени!
6. Напиши програма со која ќе најдеш решение на линеарната равенка чии параметри се a=3, b=-18!

4.5 Дополнителни специфики на јазикот

4.5.1 Исказ за внесување на податоци во програма

Програмите кои прикажуваат излез како информација која е непроменлива и е независна од корисникот се неинтерактивни компјутерски програми. Во таквите програми на променливите им се доделува некоја вредност во самата програма. За на променливите да им се доделат поинакви вредности, мора да се измени програмата, повторно да се преведе и повторно да се изврши. Ова не е она што се очекува од програмите. Програмите треба да бидат интерактивни каде корисникот ќе може да внесе вредности на променливи според неговите потреби и да се прикажат резултати зависно од влезните податоци на корисникот. Секако дека во програмските јазици тоа е овозможено.

Во C++ преземањето на податоци се извршува со операторот >> во комбинација со исказот `cin`. На пример, со исказот

```
cin>>broj;
```

се прифаќа вредност која корисникот ја внесува преку тастатурата и ѝ се доделува на променливата `broj`.

Да ги анализираме составните делови на овој исказ:

- `cin` – функција која „извлекува“ податоци од влезот, стандарден влез е тастатурата,
- `>>` – оператор за читање кој презема податок од влезот и го праќа во соодветна променлива,
- `broj` – име на променлива на која ѝ се доделува вредност прифатена од тастатурата,
- `;` – означува крај на исказ (не на линија).

Примена на исказот за внесување на податоци:

`cin>>x;`	Програмата очекува внесување на податок кој ќе се зачува во променливата x.
`cin>>a>>b;`	Можно е со еден исказ за внесување да се внесат повеќе податоци кои во променливите се зачувуваат по редоследот од лево кон десно.

Корисникот податоците ги внесува преку тастатурата. Исказот `cin` ја чита внесената вредност и ја доделува на променливата чие име е наведено како параметар во исказот.

По внесувањето на податоците, корисникот мора да го притисне кочето Enter, инаку програмата нема да ги преземе податоците и нема да продолжи со работа.

`cin>>a>>b;` е исто што и

```
cin>>a;
cin>>b;
```

Во двата случаи корисникот треба да внесе два податока, првиот за променливата `a` и вториот за променливата `b`. При тоа, по внесување на секој податок може да го притисне кочето Enter, но податоците може да ги одвои и со притискање на копчињата Space или Tab.

Во општ случај исказот `cin` го користиме на следниот начин:

```
cin>>promenliva[>>promenliva];
```

Пр. 4. 24. Програмата за собирање на два броја е преуредена така што корисникот ќе внесува податоци за променливите a и b.

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int a, b, c; //deklariranje na promenlivi
7      cin>>a>>b;   //vnesuvanje na promenlivi
8      c = a+b;     //dodeluvanje na vrednost
9      cout<<"c = "<<c<<endl;
10     system ("PAUSE");
11     return 0;
12 }
13
```

внесува корисник
внесува корисник

```
8
3
c = 11
Press any key to continue
```

4.5.2 Техника за објаснувања за податоците кои се очекуваат од корисникот

Програмата од претходниот пример нема грешки, сепак нешто недостасува. Кога ќе ја покренеш програмата добиваш празен екран, програмата чека да внесеш некој податок. Програмата сам си ја напишал и знаеш што треба да направиш, но што ако некој друг ја користи програмата?

Проблемот може да се случи и ако корисникот внесе погрешен податок. Изврши ја програмата и за променливата a внеси некој реален број. Програмата ќе даде неточен резултат. Од овие причини на корисникот треба да му се даде појаснување во вид на порака што од него се очекува. Пред секој `cin` исказ треба да се напише и `cout` исказ со кој на корисникот ќе му се прикаже порака каков податок од него се очекува.

Значи наместо

```
cin>>a;
```

се пишува

```
cout<<"Vnesi cel broj: "<<endl;
cin>>a;
```

Пр. 4. 25. Програмата во која се користат пораки за објаснување какви податоци се очекуваат од корисникот:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      // deklariranje na promenlivi
7      int a, b, c;
8      //vnesuvanje na promenlivi
9      cout<<"Vnesi cel broj: ";
10     cin>>a;
11     cout<<"Vnesi uste eden cel broj: ";
12     cin>>b;
13     c = a+b; // dodeluvanje na vrednost
14     cout<<"Zbirot na broevite ";
15     cout<<a<<" i "<<b<<" iznesuva "<<c<<e
16     system ("PAUSE");
17     return 0;
18 }
```

```
Vnesi cel broj: 6
Vnesi uste eden cel broj: 4
Zbirot na broevite 6 i 4 iznesuva 10
Press any key to continue . . .
```

Пр. 4. 26. Програма за пресметување на претходник и следбеник на цел број.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a;
    cout<<"Vnesi cel broj: ";
    cin>>a;
    cout<<"Prethodnik na brojot "<<a<<" e "<<a-1<<endl;
    cout<<"Sledbenik na brojot "<<a<<" e "<<a+1<<endl;
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 8
Prethodnik na brojot 8 e 7
Sledbenik na brojot 8 e 9
Press any key to continue . . .
```

Резиме

Програмите каде корисникот може да внесе вредности на променливи и со кои се прикажуваат резултати зависно од влезните податоци на корисникот се нарекуваат интерактивни програми.

Во C++ преземањето на податоци се извршува со операторот >> во комбинација со исказот cin. Во општ случај исказот cin се користи на следниот начин: `cin>>promenliva[>>promenliva];`

Пред секој cin исказ треба да се напише и cout исказ со кој на корисникот му се прикажува порака каков податок од него се очекува.

Вештини што треба да ги усовршиш:

Да го користиш исказот за внесување на податоци во програма (cin).

Правилно да го користиш операторот >> во исказот за внесување на податоци.

Да користиш техника со која на корисникот ќе му објасниш што од него се очекува.

Да напишеш едноставна програма со исказот за внесување на податоци во програма.

Прашања:

1. Кои програми се нарекуваат интерактивни програми?
2. Со кој исказ се овозможува внесување на вредности за променливи?
3. Како корисникот внесува вредности во програма?
4. Напиши исказ со кој преку тастатурата се внесува реален број и се придружува на променливата t!
5. Зошто е потребно на корисникот да му се прикаже порака пред тој да внесе некоја вредност?
6. Кои вредности ќе им се доделат на променливите a и b со исказот cin>>a>>b; ако корисникот по ред ги внесе следниве вредности: -1, 1?
7. Што ќе се прикаже на мониторот по извршување на следниот код:

```
int i,j,k;
cin>>j>>k>>i;
cout<<i<<", "<<j<<", "<<k;
```

ако корисникот по ред ги внесе следните вредности: 1, 2, 3?

Задачи:

1. Напиши програма со која ќе се пресметаат плоштината и периметарот на правоаголник! Овозможи корисникот да внесе вредности на страните на правоаголникот.

Пример 1: Vnesi gi stranite na pravoagolnikot: 5 3 Plostinata na pravoagolnikot e 15, a negoviot perimetar e 16	Пример 2: Vnesi gi stranite na pravoagolnikot: 4.5 2.8 Plostinata na pravoagolnikot e 12.6, a negoviot perimetar e 14.6

2. Напиши програма со која ќе се пресметаат плоштината и периметарот на квадрат! Овозможи корисникот да внесе вредност на страната на квадратот.

Пример 1: Vnesi ja stranata na kvadratot: 5 Plostinata na kvadratot e 25, a negoviot perimetar e 20	Пример 2: Vnesi ja stranata na kvadratot: 7.2 Plostinata na kvadratot e 51.84, a negoviot perimetar e 28.4

3. Напиши програма со која ќе се пресметаат плоштината и периметарот на круг со даден радиус!

Пример 1: Vnesi go radiusot r= 5 Plostinata na krugot e 78.5, a negoviot perimetar e 31.4	Пример 2: Vnesi go radiusot r= 3.2 Plostinata na krugot e 32.1536, a negoviot perimetar e 20.096

4. Напиши програма која и помага на касирка да пресмета кусур, така што од неа бара да внесе цена на артикал, количина на артикал и износ кој зема од купувачот!

Пример: Cena: 100 Kolicina: 4.5 Iznos: 1000 Kusur: 550

4.5.3 Дополнителни спецификации на јазикот C++ (прв дел)

Операцијата делење со целобројни и со реални променливи

Да ја разгледаме следнава програма:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout<<"Vnesi dva celi broja!"<<endl;
    cin >>a>>b;
    cout<<"Zbirot na broevite "<<a<<" i "<<b<<" iznesuva "<<a+b<<endl;
    cout<<"Razlikata na broevite "<<a<<" i "<<b<<" iznesuva "<<a-b<<endl;
    cout<<"Proizvodot na broevite "<<a<<" i "<<b<<" iznesuva "<<a*b<<endl;
    cout<<"Kolicnikot na broevite "<<a<<" i "<<b<<" iznesuva "<<a/b<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dva celi broja!
5       2
Zbirot na broevite 5 i 2 iznesuva 7
Razlikata na broevite 5 i 2 iznesuva 3
Proizvodot na broevite 5 i 2 iznesuva 10
Kolicnikot na broevite 5 i 2 iznesuva 2
Press any key to continue . . .
```

Гледаме дека операторите работат главно тоа што од нив се очекува, освен операторот за делење. Наместо очекуваниот резултат 2.5, програмата дава резултат 2. Ова се случува затоа што променливите a и b се декларирани како целобројни променливи па програмата како резултат враќа цел број (го занемарува децималниот дел на резултатот).

Истото ќе се случи и ако резултатот ќе се зачува во нова променлива која ќе се декларира како реална променлива. Со секвенцата од искази:

```
int a = 5;
int b = 2;
float kolicnik = a/b;
cout<<kolicnik;
```

на екранот ќе се прикаже 2 иако променливата `kolicnik` е декларирана како реална променлива.

Зад. 4. 10. Во математиката изразот 1/3*3 има вредност 1. Напиши програма со која ќе провериш која вредност ја има овој израз во C++! Објасни!

Ваков начин на делење не е грешка на програмскиот јазик C++. Вака дефинираната операција на делење на целобројните константи и променливи се нарекуваат *целобројно делење,* а резултатот се нарекува *целоброен количник.* На пр.: `10/2=5, 10/3=3, 100/75=1, 75/100=0` итн.

За целобројните константи и променливи е дефинирана уште една операција – *остаток при целобројното делење*, за која се користи операторот „%“ (modul). Така `5%2 = 1`, бидејќи остатокот на делењето на броевите 5 и 2 е 1, `10%2=0, 10%3=1, 100%75=25, 75%100=75` итн.

Зад. 4. 11. Напиши програма со која се пресметува количникот и остатокот на делење на два цели броја!

Пр. 4. 27. Програма со која се наоѓаат цифри на даден двоцифрен број:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Vnesi dvocifren broj!"<<endl;
    cin>>n;
    cout<<"Prvata cifra na brojot ";
    cout <<n<<" e "<<n/10<<endl;
    cout<<"Vtorata cifra na brojot ";
    cout <<n<<" e "<<n%10<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dvocifren broj!
38
Prvata cifra na brojot 38 e 3
Vtorata cifra na brojot 38 e 8
Press any key to continue . . .
```

Зад. 4. 12. Напиши програма со која ќе го најдеш збирот на цифрите на даден двоцифрен број!

Претварање на типови податоци

Видовме дека ако и деленикот и делителот се цели броеви, тогаш и количникот ќе биде цел број без разлика дали броевите се деливи или не. На пример, со исказот

```
float kolicnik = 5/2;
```

променливата `kolicnik` ќе добие вредност 2 (цел број).

Секој цел број може да се запише како реален број, на пр. бројот 2 може да се запише како 2.0 или скратено 2. Ако претходниот исказ се запише како

```
float kolicnik = 5./2; или float kolicnik = 5/2.;
```

променливата `kolicnik` ќе добие вредност 2.5 (реален број)

Доволно е децималната точка да се стави само после еден операнд (величина врз која се извршуваат операции).

Во C++ е дозволено претварање на типови податоци, тоа се одвива според одредени правила:

- Ако операндите се од ист тип, од тој тип е и резултатот.
- Ако операндите се од различнин типови, двата операнди и резултатот се претвораат во посложениот тип. Така ако еден операнд е цел број, а друг реален, двата операнди ќе се претворат во реален број, а таков ќе биде и резултатот.

Примери на претварање разни типови податоци:

`int a,b;`	резултатот на изразот `a/b` ќе биде цел број
`int a;` `float b;`	резултатот на изразот `a/b` ќе биде реален број
`int a,b;` `int c;`	резултатот на изразот `a*b/c` ќе биде цел број
`int a,b;` `float c;`	резултатот на изразот `a*b/c` ќе биде реален број

Пр. 4. 28. Програма која пресметува дадени метри во километри.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int m;
    cout<<"Vnesi broj na metri!"<<endl;
    cin>>m;
    cout<<m<<" metri e "<<m/1000.<<" km "<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi broj na metri!
100
100 metri e 0.1 km
Press any key to continue . . .
```

Се поставува прашање како да се постапи за да се изврши делење на две целобројни променливи и да се добие резултат со децимални места. Решението е во користење на *операторот за доделување на тип* со кој променливите привремено се однесуваат како променливи од друг тип. Така целобројната променлива a во изразот `(float) a` се однесува како реална променлива.

Со секвенцата од наредби:

```
int a = 5;
int b = 2;
float kolicnik = (float)a/b;    //ili float kolicnik = a/(float)b;
```

на променливата kolicnik ѝ се доделува вредност 2.5.

Со употреба на операторот `float`, вредностите на променливите a и b се претвораат во реални броеви пред операцијата делење. Самите променливи a и b не го менуваат својот тип така што со повторното користење на исказот

```
kolicnik = a / b;
```

на променливата `kolicnik` ќе ѝ се додели вредност 2.

Редослед на операции

Во математиката постои утврден редослед на извршување операции: операциите се извршуваат од лево кон десно ако сите операции се од ист приоритет. Овој редослед се менува доколку има операции кои не се од ист приоритет. Опеарациите множење, делење и остаток при делењето имаат повисок приоритет што значи дека прво ќе се извршат овие операции па дури тогаш ќе се извршат операциите собирање и одземање. Заградите го менуваат приоритетот на начин што прво се пресметува изразот во заградите:

`6/3+3 =5` `6/(3+3)=1`

Истиот редослед е усвоен и во програмскиот јазик C++. Така со исказот `a=5+7/2` променливата a ќе добие вредност 8, затоа што операцијата целобројно делење има предност.

За `a=12, b=3, c=4` вредноста на на изразот `a/b*c` ќе биде `12/3*4=4*4=16`, додека вредноста на на изразот `a/(b*c)` ќе биде `12/(3*4) = 12/12 = 1`.

Со кој од овие два изрази е претставен математичкиот израз $\frac{a}{bc}$?

Совет:

Ако не си сигурен/на која операција прва ќе се изврши, тоа што сакаш да се изврши прво стави го во загради. Наместо a = 5+7/2 можеш да напишеш и a = 5+(7/2).

Скратување на изрази. Оператори +=, -=, *=, /=, %=, ++, --

Често се јавува потреба од промена на вредноста на една променлива, за што се користат изрази во кои се врши некоја математичка операција врз истата таа променлива. На пр. со изразот `a=a-1;` вредноста на променливата a се намалува за 1, со изразот `s=s+a;` вредноста на променливата s се зголемува за вредноста на променливата a.

C++ поддржува пет оператори (по еден оператор за секоја аритметичка операција: `+=, -=, *=, /=, %=`) кои овозможуваат скратено пишување на ваквите изрази. Така, наместо изразот `a=a+1;` може да се напише `a+=1;`, наместо `s=s+a;` може да се напише `s+=a;`, наместо `a=a*b;` може да се напише `a*=b;` итн. Следнава табела ги содржи сите потребни информации за овие оператори:

Примери за скратување на изрази:

Скратен израз	Еквивалентно со
`a*=5`	`a=a*5`
`a/=5-b`	`a=a/(5-b)`
`a*=(b-c)+5`	`a=a*((b-c)+5))`

Пр. 4. 29. Демонстрација за скратување на изрази:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a=12;
    cout<<"a="<<a<<endl;
    a+=5;
    cout<<"po izrazot a+=5, a = "<<a<<endl;
    a-=5;
    cout<<"po izrazot a-=5, a = "<<a<<endl;
    a*=5;
    cout<<"po izrazot a*=5, a = "<<a<<endl;
    a/=5;
    cout<<"po izrazot a/=5, a = "<<a<<endl;
    a%=5;
    cout<<"po izrazot a%=5, a = "<<a<<endl;
    system("PAUSE");
    return 0;
}
```

```
a=12
po izrazot a+=5, a = 17
po izrazot a-=5, a = 12
po izrazot a*=5, a = 60
po izrazot a/=5, a = 12
po izrazot a%=5, a = 2
Press any key to continue . . .
```

Операциите *зголемување за 1* и *намалување за 1* можат да се напишат уште пократко. Во C++ постојат оператори со кои се дефинирани овие операции:

- ++ – зголемување за 1 (increment)
- -- – намалување за 1 (decrement)

Изразите:

```
a=a+1;        a+=1;        и        a++;
```

имаат иста функција, сите ја зголемуваат вредноста на променливата a за 1.

За љубопитните:

C++ го добил името токму по операторот ++.

Всушност, овие два оператора (++ и --) можат да се користат на два начина: како префикс или како постфикс, зависно од тоа дали операторот се наоѓа пред променливата (++a) или по променливата (a++). И во двата случаи вредноста на променливата се зголемува, односно намалува за 1, но кога резултатот од изразите се користи во некои искази, важно е да се обрне внимание на положбата на операторот во однос на променливата. Операторот како префикс прво делува на променливата од неговата десна страна и ја менува нејзината вредност, потоа враќа нова (променета) вредност. Операторот како постфикс прво ја враќа постоечката вредност, потоа делува на променливата од неговата лева страна и ја менува нејзината вредност.

Израз	Значење
a++	ја враќа вредноста која се содржи во променливата a, па ја зголемува вредноста на a за 1
++a	ја зголемува вредноста на a за 1, па ја враќа вредноста која се содржи во променливата a
a--	ја враќа вредноста која се содржи во променливата a, па ја намалува вредноста на a за 1
--a	ја намалува вредноста на променливата a за 1, па ја враќа вредноста која се содржи во променливата a

Пр. 4. 30. Демонстрација на операцијата a++:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int x,a;
    cout<<"Vnesi cel broj: ";
    cin>>a;
    x=a++;
    cout<<"Po izvrsuvanje na operacijata a++:"<<endl
    cout<<"x = "<<x<<endl;
    cout<<"a = "<<a<<endl;
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 5
Po izvrsuvanje na operacijata a++:
x = 5
a = 6
Press any key to continue . . .
```

Пр. 4. 31. Демонстрација на операцијата ++a:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int x,a;
    cout<<"Vnesi cel broj: ";
    cin>>a;
    x=++a;
    cout<<"Po izvrsuvanje na operacijata ++a:"<<endl;
    cout<<"x = "<<x<<endl;
    cout<<"a = "<<a<<endl;
    system("PAUSE");
    return 0;
}
```

```
Vnesi cel broj: 5
Po izvrsuvanje na operacijata ++a:
x = 6
a = 6
Press any key to continue . . .
```

Резиме

За целобројните константи и променливи се дефинирани операциите целобројно делење (/) и остаток при целобројното делење (%).

C++ подржува пет оператори (по еден оператор за секоја аритметичка операција: +=, -=, *=, /=, %=) кои овозможуваат скратено пишување на изрази со кои се врши промена на вредност на некоја променлива.

Операциите зголемување за 1 и намалување за 1 можат да се напишат уште пократко со користење на операторите ++ и -- соодветно. Овие два оператори (++ и --) можат да се користат на два начини: како префикс или како постфикс (наставка), зависно од тоа дали операторот се наоѓа пред променливата (++a) или по променливата (a++).

Вештини што треба да ги усовршиш:

Правилно да ја користиш операцијата делење со цели и со реални броеви.

Да користиш операција модул (%) со цели броеви.

Да користиш скратено запишување на изрази со операции.

Да напишеш програма со до сега изучените техники.

Прашања:

1. Пресметај!

   а) `385/100=___` б) `385%10=___`

   в) `385/10%10=____` г) `385%100/10=__`

2. Што ќе се испише на екранот по следната секвенца од наредби?

   а)
   ```
   int a = 4;
   int b = 5;
   cout<<(a+b)/2;
   ```

   б)
   ```
   int a = 4;
   int b = 5;
   cout<<(a+b)/2.;
   ```

3. Следните аритметички изрази напиши ги на програмскиот јазик C++!

   $2(a+b)$ $\frac{1}{3}x$ $\frac{a+b}{2}$

4. Колкава е вредноста на изразот `a*b/c` за целобројните променливи `a,b,c`:

   а) `a=9, b=4, c= 6?` б) `a=10, b=5, c= 4?`

5. За целобројните променливи `a,b,c`, каде `a=8, b=12, c=4, d=6` пресметај

   а) `a*b/c*d=____` б) `a*b/(c*d)=____` в) `(a*b)/(c*d)=_____`

6. Следниве изрази претстави ги во скратена форма!

   а) `a = a-9;` б) `x = x/(y+1);` в) `a=a*((b-c)+5));`

7. Колкава ќе биде вредноста на променливите a и b по извршување на следниве секвенци од искази?

   а)
   ```
   a = 1;
   b = ++a + 5;
   ```

   б)
   ```
   a = 1;
   b = a++ + 5;
   ```

Задачи:

1. Напиши програма со која ќе пресметаш средна вредност на два цели броја!

Пример 1:	Пример 2:
Vnesi dva celi broja: 6 8 Srednata vrednost na 6 i 8 e 7	Vnesi dva celi broja: 7 8 Srednata vrednost na 7 i 8 e 7.5

2. Напиши програма со која ќе ги најдеш цифрите на даден трицифрен број! Помош: погледни го прашањето 1!

Пример 1:	Пример 2:
Vnesi tricifren broj: 485 Na brojot 485 prvata cifra e 4, vtorata cifra e 8, tretata cifra e 5	Vnesi tricifren broj: 326 Na brojot 326 prvata cifra e 3, vtorata cifra e 2, tretata cifra e 6

3. Напиши програма со која време дадено во секунди ќе се прикаже во часови, минути и секунди!

Пример 1:	Пример 2:
Vnesi vreme vo sekundi: 1000 1000 sekundi se 0 casovi, 16 minuti i 40 sekundi	Vnesi vreme vo sekundi: 20000 20000 sekundi se 5 casovi, 33 minuti i 20 sekundi

4.5.4 Дополнителни спецификации на јазикот C++ (втор дел)

Константи и променливи од типот char

Знаковните константи од типот char се пишуваат помеѓу апострофи, на пр. `'a'`, `'A'`, `'+'`, `'4'` итн. Внимавај, '4' и 4 не се исти податоци! Зошто?

За чување на знаците во меморијата е предвиден 1 бајт, што значи дека може да се зачуваат 2^8=256 различни знаци. Сите знаци се претставени во ASCII табела по точно определен редослед (од 0 до 255) и секој знак има свој реден број. Така буквата `'A'` има реден број 65, буквата `'a'` има реден број 97 итн.

Препорака:

ASCII табелата погледни ја на веб страната http://www.asciitable.com/.

Во знаковна променлива може да се зачува:

- еден знак запишан помеѓу апострофи или
- ASCII вредност на тој знак (декадна вредност).

Примери на доделување вредност на знаковна променлива:

```
char bukva1='A';
char bukva2=65;
```

Во двете променливи ќе се зачува иста вредност, знакот `'A'`, и тоа во форма на број кој претставува ASCII вредност на знакот `'A'`.

Знаковните константи и променливи можат да се споредуваат:

`cout<<'a'<'b'`	Со овој исказ на екранот ќе се прикаже 1 (точно).
`cout<<'a'<'B'`	Со овој исказ на екранот ќе се прикаже 0 (неточно).

Исто така, на знаковните константи и променливи можат да се применат и математичките операции што ќе го видиме во следниот пример. Всушност, математичките операции се извршуваат врз декадните вредности на знакот во ASCII табелата.

Пр. 4. 32. Програмата ги печати претходниот и следниот знак на дадениот знак:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      char znak;
7      cout<<"Vnesi eden znak: ";
8      cin >>znak;
9      cout<<"Pred znakot "<<znak;
10     cout<<" e znakot "<<(char)(znak-1)<<endl;
11     cout<<"Po znakot "<<znak;
12     cout<<" e znakot "<<(char)(znak+1)<<endl;
13     system ("PAUSE");
14     return 0;
15 }
```

```
Vnesi eden znak: d
Pred znakot d e znakot c
Po znakot d e znakot e
Press any key to continue . . .
```

Забелешка:

Обрни внимание на употреба на операторот за доделување на тип (char) пред изразот znak - 1! Зошто мора да се користи овој оператор?

Константи и променливи од типот string

Знаковните константи најчесто се користат во низи од знаци или стрингови. Стринговите се пишуваат помеѓу наводници и до сега ги користевме за испишување на пораки на екран. Стринг со исказот `cin` може да се внесе преку тастатурата и да се зачува во некоја променлива (кога се внесува стринг преку тастатурата, тогаш тој не се запишува помеѓу наводници).

Пр. 4. 33. Програма со која се чита стринг и се зачувува во стрингова променлива:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    string ime;
    cout<<"Kako se vikas?"<<endl;
    cin>>ime;
    cout<<endl<<ime<<", dobar den!"<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Kako se vikas?
Petar

Petar, dobar den!
Press any key to continue . . .
```

Исказот `cin` не е погоден за прифаќање на стрингови кои се состојат од повеќе зборови затоа што тој чита податок до првиот знак бланко (празно место), па од стрингот може да вчита само еден збор. Зошто? Во таквите ситуации се користи функција getline.

Пр. 4. 34. Внесување на стрингови од повеќе зборови:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    string ImePrezime;
    cout<<"Kako se vikas?"<<endl;
    getline (cin,ImePrezime);
    cout<<"Zdravo, "<<ImePrezime<<"!"<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Kako se vikas?
Petar Markovski
Zdravo, Petar Markovski!
Press any key to continue . . .
```

Форматирано печатење

Веќе се запознавме со изразот endl со кој курсорот поминува во нов ред. Сега ќе се запознаеме со уште некои техники за форматирано печатење.

За поминување во нов ред, во исказот cout се вклучува специјален знак \n.

Пр. 4. 35. Со исказот:

```
cout<<"Zdravo!\n"<<"Kako se vikas?";
```

на екранот ќе се прикаже:

```
Zdravo!
Kako se vikas?
```

За податоците да се прикажат табеларно, се користи специјален знак \t.

Пр. 4. 36. Со следниот код:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    cout<<"Red.br."<<"\tZadaca1"<<"\tZadaca2"<<"\tZadaca3"<<"\n";
    cout<<1<<"\t20"<<"\t15"<<"\t12"<<"\n";
    cout<<2<<"\t17"<<"\t25"<<"\t10"<<"\n";
    system ("PAUSE");
    return 0;
}
```

на екранот ќе се прикаже:

```
Red.br. Zadaca1 Zadaca2 Zadaca3
1       20      15      12
2       17      25      10
Press any key to continue . . .
```

Резиме

Знаковните константи од типот char се пишуваат помеѓу апострофи, на пр. `'a'`, `'A'`, `'+'` итн. Знаковните константи најчесто се користат во низи од знаци или стрингови. Исказот `cin` не е погоден за прифаќање на стрингови кои се состојат од повеќе зборови затоа што тој чита податок до првиот знак бланко, па се користи функцијата getline.

За форматирано печатење се користат специјалните знаци: \n за преминување во нов ред и \t за печатење во колони.

Вештини што треба да ги усовршиш:

Да користиш знаковни и стрингови променливи и константи.

Да користиш форматирано печатење.

Да напишеш програма со до сега изучените техники.

Прашања:

1. Ако буквата `'A'` во ASCII код има реден број 65, кој реден број го има буквата `'D'`?
2. Со кој тип мора да се декларира променливата x, ако сакаме да ѝ доделиме вредност '2'?
3. Објасни за што се користат специјалните знаци \n и \t!
4. Дали има разлика помеѓу следниве искази?

```
cout<<"Dobar den<<endl;   и   cout<<"Dobar den\n";
```

Задачи:

1. Напиши програма со која ќе се внесе име и презиме, а ќе се отпечати презиме па име! Помош: користи две стрингови променливи и исказот cout.

Пример 1:	Пример 2:
Kako se vikas? Mite Pavlov	Kako se vikas? Jana Kostovska
Zdravo, Pavlov Mite	Zdravo, Kostovska Jana

2. Напиши програма со која табеларно ќе ги прикажеш оценките на неколку ученици по предметите информатика, математика и физика!

4.6 СТРУКТУРА ЗА ИЗБОР ОД ДВЕ МОЖНОСТИ

Како во животот, така и во програмирањето, многу често треба да се донесе некоја одлука. Токму „способност за одлучување“ програмите ги прави корисни. На пример, за да работиш со компјутер треба да внесеш лозинка, посебна програма ја проверува лозинката и ако таа е точна дозволува работа со компјутерот, во спротивно пристапот до компјутерот е оневозможен. Во општ случај, програмата испитува некој услов и ако тој услов е задоволен се извршуваат едни инструкции. Ако условот не е задоволен се извршуваат други инструкции.

4.6.1 Споредбени изрази

Одлуките се контролираат со *логичките изрази*. Наједноставните логички изрази се изрази во кои се споредуваат две вредности. Таквите изрази се нарекуваат *споредбени изрази*. Споредување се врши само помеѓу две вредности од ист тип, на пр. можат да се споредат два броја, два знака или две низи од знаци. За градење на споредбени изрази се користат оператори за споредување:

Оператор	Математички симбол	Опис
<	<	помало
<=	≤	помало или еднакво
>	>	поголемо
>=	≥	поголемо или еднакво
==	=	еднакво
!=	≠	не е еднакво

Внимавај!

Операторите кои се состојат од два симбола, немаат празно место помеѓу нив!

За секој споредбен израз секогаш може да се утврди дали тој е точен или е неточен, односно дали има вредност 1 (точно) или 0 (неточно). Резултатот на споредување е податок од тип bool.

Примери на споредбени изрази и нивните вредности:

Израз	Вредност
`7 >= 5`	1 (точно)
`3 + 8 <= 3`	0 (неточно)
`5 == 4 - 1`	0 (неточно)
`3 != 0`	1 (точно)

Секако, освен константи, може да се користат и изрази со променливи на кои им е доделена некоја вредност. На пример, за a=2, b=3 и c=6:

Израз	Вредност
`a == 5`	0, бидејќи 2==5 не е точно
`a*b >= c`	1, бидејќи 2*3>=6 е точно
`b+4 > a*c`	0, бидејќи 3+4>2*6 не е точно

И логичките податоци, како и сите други, можат да се зачуваат во некоја променлива. Од кој тип ќе биде таа променлива?

Пр. 4. 37. Користење на променлива од тип bool:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    bool c;
    c = 5>3;
    cout<<"Dali 5>3?"<<endl<<c<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Dali 5>3?
1
Press any key to continue . .
```

Сложени логички изрази

Логичките изрази можат да бидат и посложени, а тие се конструираат од споредбените изрази со помош на логичките оператори *И*, *ИЛИ* и *НЕ*.

Логички оператори во C++:

Оператор	Ознака за оператор	Математички симбол	Опис
AND (И)	&&	$\wedge$	Конјукција
OR (ИЛИ)	||	$\vee$	Дисјункција
NOT (НЕ)	!	$\neg$	Негација

Таблиците на вистинитост се познати од математиката:

p	q	p&&q
0	0	0
0	1	0
1	0	0
1	1	1

p	q	p||q
0	0	0
0	1	1
1	0	1
1	1	1

p	!p
0	1
1	0

При одредување на вистинитост на некој сложен израз редослед на операции е: негација, конјукција па дисјунција.

Примери на логички изрази:

Логички израз	Вредност
(1 && 0)||0	0
!((3==3) || 2>7)	0
!((5>3 || 2==5) && 2>5)	1

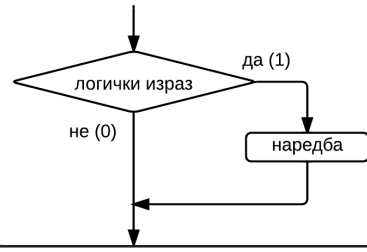
4.6.2 Структура (исказ) избор од две можности

Структурата за избор од две можности (разгранета структура) овозможува различен тек на програмата зависно од резултатот на поставениот услов. Ако изразот во условот е точен тогаш ќе се изврши некоја наредба, а ако изразот не е точен таа наредба нема да се изврши, а може но не мора да се изврши друга наредба.

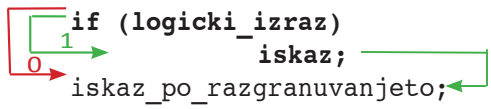
Разгранувањето може да биде еднократно и двократно.

Еднократно разгранување

Ова е наједноставна форма на разгранета структура: ако некој услов е исполнет ќе се изврши некоја наредба, во спротивно таа наредба нема да се изврши.

Псевдо алгоритам за еднократно разгранување:	*Блок дијаграм за еднократно разгранување:*
ако услов *тогаш* наредба;	

Во програмскиот јазик C++ за еднократно разгранување се користи исказот `if` кој има општ облик:



Ако вредноста на логичкиот израз (`logicki_izraz`) е вистинита (`true`, `1`), се изведува исказот по изразот `if (logicki_izraz)`. Ако вредноста на логичкиот израз (`logicki_izraz`) е невистинита (`false`, `0`), исказот по изразот `if (logicki_izraz)` се прескокнува и се изведува следниот исказ по разгранувањето.

Со следниов исказ се проверува дали дадениот број x е негативен:

```
if (x < 0)
      cout<<"Brojot "<<x<<" e negativen";
```

Ако x има негативна вредност на пр. -10, логичкиот израз `x < 0` ќе биде точен, односно ќе има вредност 1, па ќе се изврши исказот

`cout<<"Brojot "<<x<<" e negativen";`.

На екранот ќе се прикаже: „`Brojot -10 e negativen`“.

Ако x нема негативна вредност, на пр. 10, тогаш логичкиот израз `x < 0` ќе биде неточен, односно ќе има вредност 0, па исказот

`cout<<"Brojot "<<x<<" e negativen";`

нема да се изврши. На екранот ништо нема да се прикаже.

Исказот може да се запише и како

```
if (x < 0) cout<<"Brojot "<<x<<" e negativen";
```

но се запишува на претходен начин заради прегледност.

Важно!

По изразот if (logicki_izraz) не се става точка и запирка (;). Ако ова го направиш преведувачот ќе мисли дека знакот ; припаѓа на наредба која треба да се изврши ако условот е исполнет, па ќе имаш несакан ефект. Помеѓу условот и знакот ; нема ништо, па преведувачот гледа празна наредба и програмата нема ништо да изврши.

Пр. 4. 38. Со следнава програма се проверува дали променливата `delitel` е еднаква на 0:

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      int delitel;
7      cout<<"Vnesi broj: ";
8      cin>>delitel;
9      if (delitel == 0)
10         cout<<"Ne e mozno delenje so 0!";
11     system("PAUSE");
12     return 0;
13 }
```

```
Vnesi cel broj! 0
Ne e mozno delenje so 0!
Press any key to continue . .
```

Ако променливата `delitel` има вредност 0 програмата ќе не извести дека со 0 не може да се дели.

Внимавај!

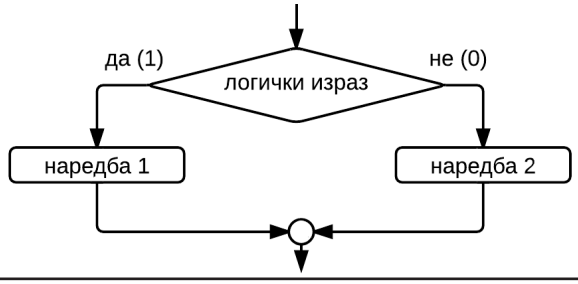
Често се случува грешка и наместо операторот == се користи операторот =. Ова може да доведе до неисправно работење на програма. Внимавај овие два оператори да не ги мешаш!

Двократно разгранување

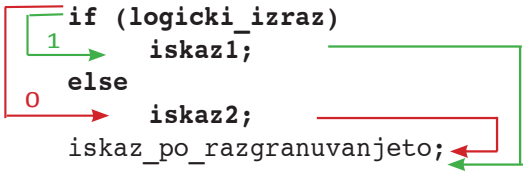
Двократно разгранување значи: ако некој услов е исполнет ќе се изврши некоја наредба, во спротивно таа наредба нема да се изврши туку ќе се изврши друга наредба.

Псевдо алгоритам за двократно разгранување:

ако услов
    тогаш наредба1
    инаку наредба2;



Во програмскиот јазик C++ за двократно разгранување се користи наредба `if-else` која има општ облик:



Ако вредноста на логичкиот израз (`logicki_izraz`) е вистина (true, 1), тогаш се извршува наредбата `iskaz1`. Откако таа ќе се изврши се извршуваат наредбите по исказот `if else`.

Ако вредноста на логичкиот израз е невистина (`false, 0`), наредбата `iskaz1` се прескокнува и се извршува наредбата по `else` – `iskaz2`. Откако таа ќе се изврши се извршуваат наредбите по исказот `if else`.

Со следниов исказ се проверува дали даден цел број е позитивен и се прикажува соодветна порака:

```
if (x < 0)
      cout<<"Brojot "<<x<<" e negativen"<<endl;
else
      cout<<"Brojot "<<x<<" ne e negativen"<<endl;
```

За `x=-8` ќе се изврши наредбата по `if` и ќе се прикаже пораката „Brojot -8 e negativen“. За `x=8` ќе се изврши наредбата по `else` и ќе се прикаже пораката „Brojot 8 ne e negativen“.

Важно!

По изразите if и else не се става знакот точка и запирка (;).

Пр. 4. 39. Програмата за два дадени различни цели броја го прокажува поголемиот број:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"Vnesi dva razlicni celi broja! ";
    cin>>a>>b;
    if (a>b)
        cout<<"Brojot "<<a<<" e pogolem."<<endl;
    else
        cout<<"Brojot "<<b<<" e pogolem."<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dva razlicni celi broja! 5 8
Brojot 8 e pogolem.
Press any key to continue . . .
```

```
Vnesi dva razlicni celi broja! 7 2
Brojot 7 e pogolem.
Press any key to continue . . .
```

Пр. 4. 40. Програмата проверува дали дадениот број е едноцифрен:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int x;
    cout<<"Vnesi cel broj! ";
    cin>>x;
    if (0 <= x && x <= 9)
        cout<<"Brojot "<<x<<" e ednocifren."<<endl;
    else
        cout<<"Brojot "<<x<<" ne e ednocifren."<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Vnesi cel broj! 6
Brojot 6 e ednocifren.
Press any key to continue . . .
```

```
Vnesi cel broj! 25
Brojot 25 ne e ednocifren.
Press any key to continue . . .
```

Важно!

Проверката дали бројот x е едноцифрен во математиката се запишува како $0 \le x \le 9$. Програмските јазици не дозволуваат ваков вид на запишување. Ова се два изрази кои мора да се поврзат со операторот && (И).

Блок од искази

Важно е да се знае дека исказот `if` контролира извршување само на првиот исказ кој се наоѓа веднаш по него. Веќе следниот исказ не е дел од исказот `if` па тој исказ ќе се изврши независно дали условот е точен или не. Да го погледнеме следниот пример:

```
if (x < 0)
        cout<<"Brojot "<<x<<" e negativen."<<endl;
        cout<<"Negovata apsolutna vrednost e "<<-x;
```

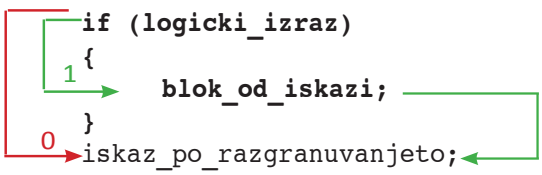
Втората нареба по исказот `if` ќе се изврши и за позитивните броеви па за нив ќе се испише погрешна апсолутна вредност.

Овој проблем се решава со градење на *блок од искази*. ***Кога исказот if контролира два или повеќе искази, сите тие се стават меѓу големи загради.***

Претходниот пример исправно се запишува на следниот начин:

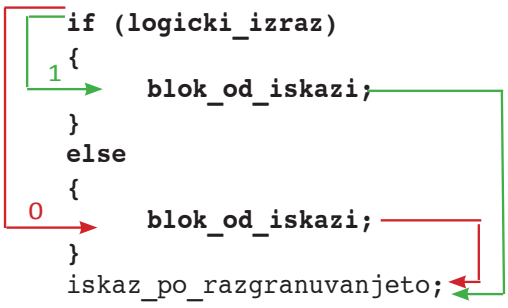
```
if (x <0)
{
        cout<<"Brojot "<<x<<" e negativen."<<endl;
        cout<<"Negovata apsolutna vrednost e "<<-x;
}
```

Во општ случај исказот за еднократно разгранување со користење на блок искази се запишува:



Ист случај е и кога има блок од искази по `else`.

Во општ случај исказот за двократно разгранување со користење на блок од искази се запишува:



Така, исказот if со кој се прикажува апсолутна вредност на бројот x се запишува:

```
if (x <0)
{
      cout<<"Brojot "<<x<<" e negativen."<<endl;
      cout<<"Negovata apsolutna vrednost e "<<-x;
}
else
{
      cout<<"Brojot "<<x<<" ne e negativen."<<endl;
      cout<<"Negovata apsolutna vrednost e "<<x;
}
```

Совет:

Нема да згрешиш ако само еден исказ ставиш помеѓу големи загради. Дури во почетокот би било добро тоа да го правиш.

Пр. 4. 41. Следнава програма од корисникот бара да внесе корисничко име и лозинка. Доколку тие се исправни ќе даде соодветна порака, во спротивно ќе даде друга порака.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    string korisnicko_ime, lozinka;
    cout<<"Vnesi korisnicko ime! "<<endl;
    cin>>korisnicko_ime;
    cout<<"Vnesi lozinka! "<<endl;
    cin>>lozinka;
    if (korisnicko_ime == "Darko" && lozinka == "C++")
    {
        cout<<"Vnesovte ispravni korisnicko ime i lozinka!";
        cout<<"Dobredojdovte, "<< korisnicko_ime <<"!"<<endl;
    }
    else
    {
        cout<<"Korisnicko ime ili lozinka ne se tocni!";
        cout<<"Pristapot ne e dozvolen!"<<endl;
    }
    system ("PAUSE");
    return 0;
}
```

```
Vnesi korisnicko ime!
Darko
Vnesi lozinka!
C++
Vnesovte ispravni korisnicko ime i lozinka!Dobredojdovte, Darko!
Press any key to continue . . .
```

```
Vnesi korisnicko ime!
Darko
Vnesi lozinka!
C+
Korisnicko ime ili lozinka ne se tocni!Pristapot ne e dozvolen!
Press any key to continue . . .
```

Што ќе се случи ако намето && ставиш ||?

Резиме

Изразите во кои ссе поредуваат две вредности се нарекуваат споредбени изрази. Операторите за споредување се: <, >, <=, >=, ==, !=. Сложените логички изрази се добиваат со поврзување на споредбените изрази со помош на логичките оператори && (И), || (ИЛИ) и ! (НЕ).

Структурата за избор од две можности (разгранета структура) овозможува различен тек на програма зависно од резултатот на поставениот услов. Ако условот е точен ќе се изврши некоја наредба, а ако не е исполнет таа наредба нема да се изврши, а може но не мора да се изврши друга наредба.

За еднократно разгранување се користи исказот if, а за двократно разгранување се користи исказот if-else. Кога исказот if контролира два или повеќе искази, тие се стават помеѓу големите загради.

Вештини што треба да ги усовршиш:

Да градиш сложени логички изрази со помош на логички оператори и да знаеш да ја провериш нивната вистинитост.

Да ги користиш структурите за еднократно и за двократно разгранување.

Да напишеш програма со користење на структурите за еднократно и за двократно разгранување.

Прашања:

1. Како се нарекуваат изразите во кои се споредуваат две вредности?
2. Кои оператори се користат за споредување?
3. Дали можат да се споредуваат вредности на променливи кои не се од ист тип?
4. Со помош на кои оператори се градат сложени логички изрази?
5. Одреди вредност на следниве споредбени изрази:
   а) 9 <= 17 б) 9 >= 25 в) 9 == 13 г) 9 != 13 д) 9 !< 25
6. Одреди вредност на следниве логички изрази:
   а) 14 > 7 && 5 <= 5 б) 4 > 3 || 17 <= 7
   в) ! (13 != 7) г) 9 != 7 && ! 0
7. Како се нарекува структура која овозможува разгранување во програма?
8. Кој исказ се користи за разгранување во програма?
9. Напиши ги синтаксите на исказите `if` и `if-else`!
10. Како се означува блок од искази кои се користат во исказите `if` и `if-else`?
11. Што ќе се отпечати со следната наредба?

```
if (x>10)
      cout<<"Brojot <<x<<" e pogolem od 10";
else
      cout<<"Brojot <<x<<" e pomal od 10";
```

    а) за x = 5 б) за x = 15.

12. Колкава ќе биде вредност на целобројната променлива `x` по извршување на следнава наредба?

```
if (x+2>10) x = x + 1;
```

    а) за x = 9 б) за x = 5.

13. Која вредност ќе има променливата **s** по извршување на следнава секвенца од искази:

```
s=0;
if (a>0) s=s+a;
if (b>0) s=s+b;
```

а) за a=5, b=3 б) за a=-5, b= 3 в) за a=-5, b= -3

Задачи:

1. Напиши програма која за дадениот цел број x ќе провери дали тој е парен или непарен и ќе прикаже соодветна порака!

Пример 1: Vnesi cel broj: 12 Brojot 12 e paren.	Пример 2: Vnesi cel broj: 35 Brojot 35 e neparen.

2. Внеси два различни броја, потоа опечати го помалиот, па поголемиот број!

Пример 1: Vnesi dva broja: -3 7 Pomaliot broj e -3, a pogolemiot broj e 7	Пример 2: Vnesi dva broja: 3 -7 Pomaliot broj e -7, a pogolemiot broj e 3

3. Напиши програма која ќе пресметува периметар на квадрат само ако внесената вредност за страната на квадратот е позитивна! Во спротивно ќе даде соодветна порака.

Пример 1: Strana na kvadratot: 2.5 Perimetar na kvadratot e 10!	Пример 2: Strana na kvadratot: -2.5 Stranata mora da e pozitivna!

4. Напиши програма која ќе одзема во множество на природни броеви!

Пример 1: Vnesi go prviot broj: 8 Vnesi go vtoriot broj: 5 8 - 5 = 3	Пример 2: Vnesi go prviot broj: 5 Vnesi go vtoriot broj: 8 Razlikata 5 - 8 ne postoi vo N

5. Напиши програма со која ќе се пресмета апсолутна вредност на бројот x!

Пример 1: Vnesi eden broj: 8.2 Apsolutnata vrednost na brojot 8.2 e 8.2	Пример 2: Vnesi eden broj: -8.2 Apsolutnata vrednost na brojot -8.2 e 8.2

6. Напиши програма со која се собираат само позитивните вредности на броевите a, b и c!

Пример 1: Vnesi tri broja: -3 7 5 Zbirot na pozitivnite broevi e S=12	Пример 2: Vnesi tri broja: -3 -7 -5 Zbirot na pozitivnite broevi e S=0

7. Напиши програма со која ќе се најде најголем од дадените три броја!
8. Напиши програма со која ќе се подредат три броја по големина!
9. Напиши програма со која се одредува дали од дадените отсечки со должините a, b и c може да се формира триаголник!
10. Даден е трицифрен природен број. Напиши програма со која се одредува дали бројот е палиндром (ако исто се чита од десно и од лево)!

4.6.3 Техника на вгнездување на искази

Во блокот од искази може да се користат било кои искази, па така и исказот за избор од две можности. Впрочем, и во секојдневниот живот често се случува еден условен настан да зависи од друг условен настан и е потребно да се испитаат повеќе услови. Тогаш еден услов е надворешен, а друг е внатрешен или вгнезден.

Во програмските јазици дозволено е да се користат повеќе `if` искази вгнездени еден во друг било во блокот на искази по `if` било во блокот од искази по `else`. Ова се нарекува *техника на вгнездени искази*.

Пр. 4. 42. Во примерот 4.41 може од корисникот да се побара корисничко име и ако корисничкото име е исправно тогаш се бара лозинка. Ако корисничкото име не е исправно нема потреба да се бара лозинка.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    string korisnicko_ime, lozinka;
    cout<<"Vnesi korisnicko ime! ";
    cin>>korisnicko_ime;
    if (korisnicko_ime == "Darko")
    {
        cout<<"Vnesi lozinka! ";
        cin>>lozinka;
        if (lozinka == "C++")
        {
            cout<<"Ispravni korisnicko ime i lozinka!"<<endl;
            cout<<"Dobredojdovte, ";
            cout<< korisnicko_ime <<"!"<<endl;
        }
        else
        {
            cout<<"Pogresna lozinka!"<<endl;
            cout<<"Pristapot ne e dozvolen!"<<endl;
        }
    }
    else
        {
            cout<<"Pogresno korisnicko ime!"<<endl;
            cout<<"Pristapot ne e dozvolen!"<<endl;
        }
    system("PAUSE");
    return 0;
}
```

```
Vnesi korisnicko ime! Darko
Vnesi lozinka! C++
Ispravni korisnicko ime i lozinka!
Dobredojdovte, Darko!
Press any key to continue . . .
```

```
Vnesi korisnicko ime! Darko
Vnesi lozinka! C+
Pogresna lozinka!
Pristapot ne e dozvolen!
Press any key to continue . . .
```

Совет:

За посложените програми како што е оваа добро е да се користи блок-дијаграм.

Резиме:

Во програмските јазици дозволено е да се користат повеќе if искази вгнездени еден во друг, било во блокот на искази по if, било во блокот од искази по else. Ова се нарекува техника на вгнездени искази.

Вештини што треба да ги усовршиш:

Да препознаваш и да користиш техника на вгнездени искази.
Да напишеш програма со користење на до сега изучените техники.

Прашања:

1. Која вредност ќе ја добие променливата a по извршување на следните наредби:

```
float a = 1.56;
if (a < 1)
        a = a+1;
else
{
        a = a+4;
        a = -a+7;
}
```

2. Што ќе се прикаже по извршување на следниве искази?

```
int broj = -6;
char bukva = 'K';
if (broj < 0)
{
        if (bukva < 'S')
                cout<<1;
        else
                cout<<2;
}
else
        cout<<3;
```

Задачи:

1. Напиши програма која за даден цел број проверува дали е позитивен или негативен, ако е позитивен да се провери дали е парен или непарен и да се прикаже соодветна порака!

Пример 1:	Пример 2:	Пример 3:
Vnesi cel broj: - 6	Vnesi cel broj: 7	Vnesi cel broj: 6
Brojot -6 e negativen!	Brojot 7 e pozitiven i e neparen!	Brojot 7 e pozitiven i e paren!

2. Напиши програма со која ќе се најде решение на линеарната равенка ax + b = 0! Да се разгледаат и специјалните слачаи за a=0 и b=0!

Пример 1:	Пример 2:	Пример 3:
a = 3	a = 0	a = 0
b = -21	b = -21	b = 0
Resenieto na ravenkata e x=7	Ravenkata nema resenie	Ravenkata ima beskonecno resenija

3. Да се внесат должини на три отсечки a, b и c и да се провери дали тие можат да градат триаголник. Ако можат, да се провери дали триаголникот е рамностран, рамнокрак, или разностран. За сите случаи да се прикаже соодветна порака!

4.7 Структура за избор од повеќе можности

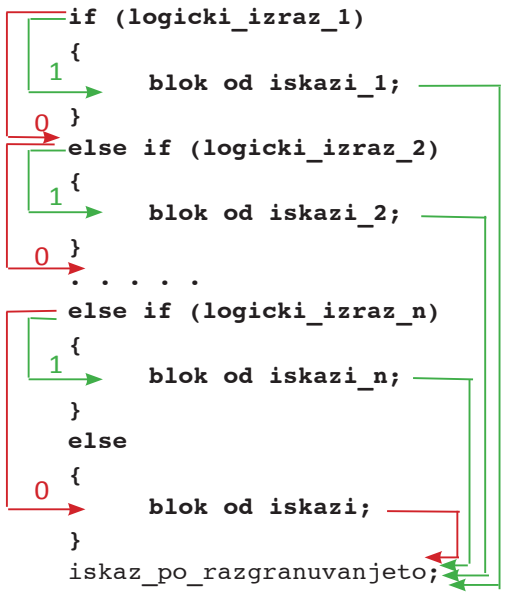
4.7.1 Повеќекратно разгранување

Да го погледнеме примерот на програма која прикажува поголем од два дадени броја:

```
if (a>b)
      cout<<"Brojot "<<a<<" e pogolem."<<endl;
else
      cout<<"Brojot "<<b<<" e pogolem."<<endl;
```

Што ќе се случи ако корисник (иако е предупреден) внесе два броја кои се еднакви? Програмата ќе провери дали a>b, и бидејќи овој услов не е исполнет, ќе го изврши исказот по else, што нема да биде исправно. За програмата да биде сосема коректна, треба да се провери и точноста на условот a==b.

Со исказот `if-else` може да се избира помеѓу две алтернативи и да се изврши само една од нив, зависно од вистинитоста на логичкиот израз. Но често се случува да треба да се бира помеѓу три или повеќе опции како во наведениот случај. Тогаш се применува *техника на вгнездување на искази*, односно се гради *повеќекратна разгранета структура*.



Пр. 4. 43. Примерот за поголемиот од два броја ќе го проширите со тоа што ќе се даде можност корисникот да внесе и два еднакви броја:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"Vnesi dva celi broja! "<<endl;
    cin>>a>>b;
    if (a == b)
        cout<<"Broevite se ednakvi"<<endl; //(1)
    else if (a > b)
            cout<<"Brojot "<<a<<" e pogolem"<<endl; // (2)
        else
            cout<<"Brojot "<<b<<" e pogolem"<<endl; //(3)
    system ("PAUSE");
    return 0;
}
```

Програмата проверува дали е точен изразот `a == b`, ако е точен ќе се исполни наредбата по делот `if` (наредбата коментирана со (1)). Ако изразот не е точен програмата поминува на делот по `else` и проверува дали е точен изразот `a > b`. Ако овој израз е точен програмата ќе го изврши исказот по делот `if` (2), инаку поминува на исказ по делот `else` (3).

Зад. 4. 13. Напиши програма која за даден цел број проверува дали тој број е позитивен, негативен или еднаков на 0 и печати соодветна порака!

Пр. 4. 44. Да погледнеме уште еден пример. Програмата за бројната оценка (1 - 5) печати соодветен успех (недоволен - одличен):

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int ocenka;
    cout<<"Vnesi ocenka od 1 do 5! "<<endl;
    cin>>ocenka;
    if (ocenka == 1)
        cout<<"Vasiot uspeh e nedovolen!"<<endl;    //(1)
    else if (ocenka == 2)
        cout<<"Vasiot uspeh e dovolen!"<<endl;    // (2)
    else if (ocenka == 3)
        cout<<"Vasiot uspeh e dobar!"<<endl;    // (3)
    else if (ocenka == 4)
        cout<<"Vasiot uspeh e mnogu dobar!"<<endl;    // (4)
    else if (ocenka == 5)
        cout<<"Vasiot uspeh e odlicen!"<<endl;    // (5)
    else
        cout<<"Vnesenata ocenka ne e regularna!"<<endl; //(greska)
    system ("PAUSE");
    return 0;
}
```

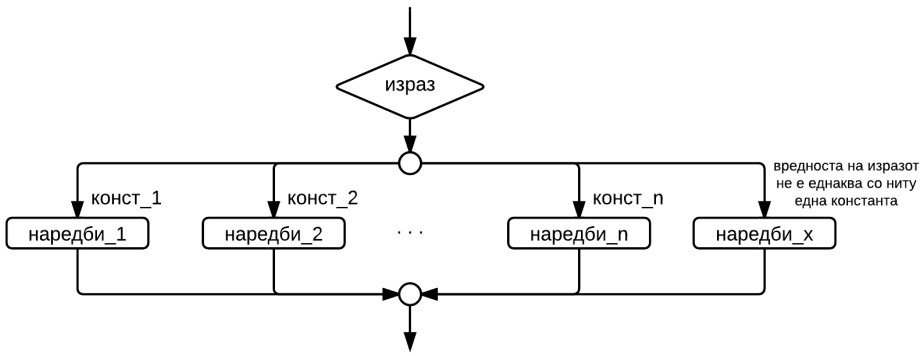
```
Vnesi ocenka od 1 do 5!
4
Vasiot uspeh e mnogu dobar!
Press any key to continue . . .
```

```
Vnesi ocenka od 1 do 5!
7
Vnesenata ocenka ne e regularna!
Press any key to continue . . .
```

4.7.2 Структура за избор од повеќе можности

Повеќекратното разгранување е добро решение кога се работи за избор од повеќе можности, но може да биде заморно и кодот да стане предолг ако се работи за поголем број на можни избори. Во програмските јазици во такви прилики се користи структурата за избор од повеќе можности.

Блок дијаграм за структурата за избор од повеќе можности:



За разлика од повеќекратното разгранување каде условите се логички изрази, кај структурата од повеќе можности условот е израз чија вредност мора да биде некоја целобројна или знаковна константа.

```
switch (izraz)
{
      case konst_1:
            blok_na_iskazi_1;
      break;
      case konst_2:
            blok_na_iskazi_2;
      break;
      . . . .
      case konst_n:
            blok_na_iskazi_n;
      break;
      default:
            posledniot_blok_na_iskazi;
}
```

Вредност на изразот се споредува со целобројните константи: `konst_1`, `konst_2`, `konst_3`, итн. Ако вредноста на изразот е еднаква на некоја од дадените константи, ќе се изврши блокот на искази придружен на таа константа. По извршувањето на тој блок, исказот `break` го прекинува извршувањето на `switch-case` исказот и програмата продолжува со првата наредба по овој исказ. Ако вредноста на изразот не е еднаква на ниту една од дадените константи, се извршува блокот на искази по `default`.

Пр. 4. 45. Програмата за успех на ученикот запишана со исказот `switch-case`:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int ocenka;
7      cout<<"Vnesi ocenka od 1 do 5! "<<endl;
8      cin>>ocenka;
9      switch (ocenka)
10     {
11         case 1:
12             cout<<"Vasiot uspeh e nedovolen!"<<endl;
13         break;
14         case 2:
15             cout<<"Vasiot uspeh e dovolen!"<<endl;
16         break;
17         case 3:
18             cout<<"Vasiot uspeh e dobar!"<<endl;
19         break;
20         case 4:
21             cout<<"Vasiot uspeh e mnogu dobar!"<<endl;
22         break;
23         case 5:
24             cout<<"Vasiot uspeh e odlicen!"<<endl;
25         break;
26         default:
27             cout<<"Vnesenata ocenka ne e regularna!"<<endl;
28     }
29     system ("PAUSE");
30     return 0;
31 }
```

```
Vnesi ocenka od 1 do 5!
4
Vasiot uspeh e mnogu dobar!
Press any key to continue . . .
```

```
Vnesi ocenka od 1 do 5!
7
Vnesenata ocenka ne e regularna!
Press any key to continue . . .
```

Важно!

Не заборавај на исказот break по блоковите на искази! Доколку тоа се случи, swich-case исказот нема да се прекине туку ќе продолжи со извршување на блокот од искази придружен на следната константа. Така, со следниот код ќе се отпечати DvaTri наместо Dva.

```
int x=2;
switch(x)
{
        case 1: cout<<"Eden";
        case 2: cout<<"Dva";
        case 3: cout<<"Tri";
}
```

Во исказот `switch-case` може да се користат и знаковни константи:

Пр. 4. 46. Корисникот внесува два броја и знак за аритметичка операција која сака да ја изврши. Програмата извршува соодветна операција и прикажува резултат.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    float a,b; char znak;
    cout<<"Vnesi dva broja: ";
    cin>>a>>b;
    cout<<"Vnesi znak za operacija: ";
    cin>>znak;
    switch (znak)
    {
        case '+':
            cout<<a<<znak<<b<<"="<<a+b<<endl;
        break;
        case '-':
            cout<<a<<znak<<b<<"="<<a-b<<endl;
        break;
        case '*':
            cout<<a<<znak<<b<<"="<<a*b<<endl;
        break;
        case '/':
            if (b == 0)
            {
                cout<<"Ne e dozvoleno delenje so 0!"<<endl;
            break;
            }
            cout<<a<<znak<<b<<"="<<a/b<<endl;
        break;
        default:
            cout<<"Pogresno vnesen znak!"<<endl;
    }
    system ("PAUSE");
    return 0;
}
```

```
Vnesi dva broja: 7 12
Vnesi znak za operacija: +
7+12=19
Press any key to continue . . .
```

```
Vnesi dva broja: 15 0
Vnesi znak za operacija: /
Ne e dozvoleno delenje so 0!
Press any key to continue . . .
```

Понекогаш е потребно за различни вредности на изразот да се извршат исти наредби. На пример, со следниот код целобројната променлива broj добива вредност 1 па ќе се изврши исказот cout << “Less than 3!”; со кој ќе се прикаже: „**Pomal od** 3!“:

```
int broj = 1;
switch (broj)
{
        case 0:
        case 1:
        case 2:
                cout << “Pomal od 3!”;
        break;
        case 3:
                cout << “Ednakov na 3!”;
        break;
        default:
                cout << “Pogolem od 3!”;
}
```

Што ќе се прикаже ако на променливата **broj** наместо 1 ѝ се додели вредност 5? За кои уште вредности на променливата **broj** ќе се прикаже „**Pomal od** 3!“?

Зад. 4. 14. Напиши програма со која корисникот ќе внесе реден број на месец во година и согласно изборот на екранот ќе се отпечати на кое годишно време тој месец му припаѓа! Во случај на несоодветен влезен податок да се прикаже соодветна порака.

Резиме

Посебен случај на вгнездени услови е кога условите се вгнездуваат само во блокот на искази по делот else. Овој случај се нарекува повеќекратно разгранување. За избор од повеќе можности се користи структурата switch-case каде условот е израз чија вредност мора да биде некоја целобројна константа.

Вештини што треба да ги усовршиш:

Да познаваш синтакса на наредбата за повеќекратно разгранување.

Да напишеш програма со користење на до сега изучените техники.

Прашања:

1. Во кои случаи се применува повеќеразгранета програмска структура?
2. Што ќе се прикаже на екранот по извршување на следниве искази за x=3?

```
int x;
cin>>x;
if (x == 1)
        cout<<”Slabo!”<<endl;
else if (x == 2)
        cout<<”Ne e loso!”<<endl;
else if (x == 3)
        cout<<”Bravo!”<<endl;
else
        cout<<”Pogresen vnes!”<<endl;
```

3. Дали постои разлика помеѓу следните кодови?

а)
```
if (znak == 1)
      broj = 10;
  else if (znak == 2)
      broj = 20;
  else
      broj = 30;
```

б)
```
switch (znak)
   {
      case 1:
           broj = 10;
       break;
       case 2:
            broj = 20;
       break;
       default:
            broj = 30;
   }
```

Задачи:

Следните задачи реши ги со двете структури!

1. Напиши програма со која корисникот ќе внесе реден број на денот во седмицата и на екранот ќе се отпечати името на соодветниот ден! Во случај на несоодветен влезен податок да се прикаже соодветна порака.

Пример 1:	Пример 2:
Vnesi reden broj na den vo sedmica: 5 Denes e petok!	Vnesi reden broj na den vo sedmica: 9 Pogresen podatok!

2. Напиши програма со која оценките A, B, C, D, E се преведуваат во бројните оценки 5, 4, 3, 2, 1! Во случај на несоодветен влезен податок да се прикаже соодветна порака!
3. Напиши програма со која корисникот ќе внесе реден број на месец во година и согласно изборот на екранот ќе се отпечати колку денови има соодветниот месец! Да се предвиди ситуација за престапна година! Во случај на несоодветен влезен податок да се прикаже соодветна порака!
4. Напиши програма со која корисникот ќе избере број пред хороскопскиот знак и согласно изборот ќе добие краток хороскоп! Во случај на несоодветен влезен податок да се прикаже соодветна порака!

4.8 Основна структура за повторување

Често пати се јавува потреба една или повеќе инструкции да се извршат повеќе пати. На пример: Додека има хартија печати покани, Напиши 100 пати „Никогаш повеќе нема да зборувам на час“ и слично. И во програмирањето многу често се јавува потреба од повторување на еден ист исказ (или на блок од искази) повеќе пати. За да не се повторува пишувањето на исказот, во програмскиот код се користи структура за повторување која се вика циклус, петелка или јамка (loop).

Структурите за повторување овозможуваат некој блок од искази да се изврши одреден број пати. При тоа, бројот на повторувањата на циклусот е дефиниран со однапред зададен природен број или зависи од некој услов кој определува кога повторувањето ќе се прекине, при што условот може да се испита пред почетокот на циклусот или по неговото завршување.

Секое повторување на циклусот се нарекува *итерација*.

4.8.1 Структура за повторување на циклус до исполнување на услов

Наједноставниот од сите циклуси е `while` циклусот. Во овој циклус прво се проверува дали некој услов е исполнет. Ако условот е исполнет, циклусот се повторува, ако условот не е исполнет тогаш не се влегува во циклусот.

Пр. 4. 47. Со кодот:

```
int i = 1;
while (i <= 10)
{
        cout<<i<<"\t";
        i++;
}
```

ќе се прикаже:

1 2 3 4 5 6 7 8 9 10

Пр. 4. 48. Со кодот:

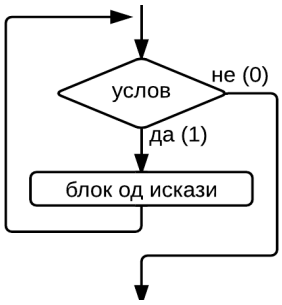
```
int i = 1;
while (i <= 10)
{
        cout<<'A'<<"\t";
        i++;
}
```

ќе се прикаже:

A A A A A A A A A A

Основна форма на while циклусот е:

```
while (uslov)
{
   blok_od_iskazi;
}
iskaz_po_ciklusot;
```



- Условот во циклусот може да биде споредбен или логички израз кој може да добие вредност точно (`true, 1`) или неточно (`false, 0`).
- Блокот од искази во циклусот се повторува сè додека условот е исполнет. Бидејќи условот се проверува на почетокот може да се случи телото на циклусот да не се изврши ниту еднаш.
- Кога условот нема да биде исполнет, блокот од искази во циклусот се прескокнува и програмата ќе продолжи со извршување на првата наредба по телото на циклусот.
- Структурата `while` мора да биде напишана така што во конечен број на повторувања логичкиот израз ќе добие вредност неточно. Така се обезбедува да се излезе од циклусот по конечен број на повторувања. Во спротивно циклусот би се повторувал бесконечно.

Важно!

По изразот while (uslov) не се става точка и запирка (;).

Пр. 4. 49. Програмата испишува броеви наназад. Корисникот внесува број од кој ќе започне испишувањето.

```
1   #include <cstdlib>
2   #include <iostream>
3   using namespace std;
4   int main()
5   {
6       int n;
7       cout<<"Ke brojam nanazad!"<<endl;
8       cout<<"Od koj broj da zapocnam? ";
9       cin >>n;
10      while (n > 0)
11      {
12          cout<<n<<endl;
13          n--;
14      }
15      system ("PAUSE");
16      return 0;
17  }
```

```
Ke brojam nanazad!
Od koj broj da zapocnam? 5
5
4
3
2
1
Press any key to continue . . .
```

Блокот од искази кои го сочинуваат циклусот се ставени во големите загради. Кои се тие искази?

На почетокот на програмата, пред циклусот, од корисникот се бара да внесе број од кој ќе започне броењето со што се определува дали циклусот ќе се повторува или не. Циклусот ќе се изврши само ако корисникот внесе број поголем од 0 (тоа е определено со условот (`broj>0`)).

Ако корисникот внесе број 5 циклусот ќе се повтори точно 5 пати. Во секое повторување програмата го испишува бројот на екранот, потоа тој број го намалува за еден и повторно го проверува условот (`broj>0`). Циклусот ќе се повторува сè додека бројот е поголем од 0. Кога бројот ќе добие вредност 0 изразот `broj>0` ќе има вредност невистина (`false, 0`) и програмата ќе продолжи со извршување на првата наредба по циклусот. Која е тоа наредба?

Да видиме што се случува при извршување на програмата:

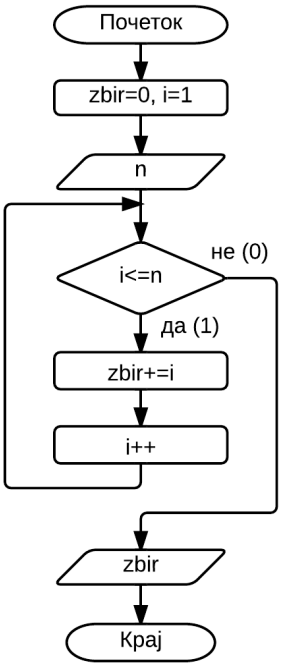
`n --;`	`(n>0)`	`cout<<n;`	Број на повторувања
5	1	5	1
4	1	4	2
3	1	3	3
2	1	2	4
1	1	1	5
0	0		

Што ќе се случи ако корисникот внесе 0 или негативен број? Циклусот нема да се изврши ниту еднаш, затоа што прво се проверува условот и во телото на циклусот се влегува само ако условот е исполнет, односно ако логичкиот израз даден во условот има вредност вистина (`true, 1`).

Ако бројот во секое повторување не се намалува за 1 тогаш логичкиот израз даден во условот (`n>0`) секогаш ќе биде точен (`true, 1`) и циклусот ќе се повторува бесконечен број пати, односно додека програмата не ја истроши работната меморија со која располага. Коментирај го исказот `n--;` и провери!

Пр. 4. 50. Програмата пресметува збир на првите n природни броеви. Бројот n го внесува корисникот.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n;
    int i = 1;
    int zbir = 0;
    cout<<"Do koj broj da sobiram? ";
    cin>>n;
    while (i<= n)
    {
        zbir = zbir + i;
        i++;
    }
    cout<<"Zbirot na broevite od "<<1;
    cout<<" do "<<n<<" iznesuva "<<zbir<<endl;
    system("PAUSE");
    return 0;
}
```




```
Do koj broj da sobiram? 6
Zbirot na broevite od 1 do 6 iznesuva 21
Press any key to continue . . .
```

Пр. 4. 51. Корисник внесува n цели броеви. Да се најде нивниот збир.

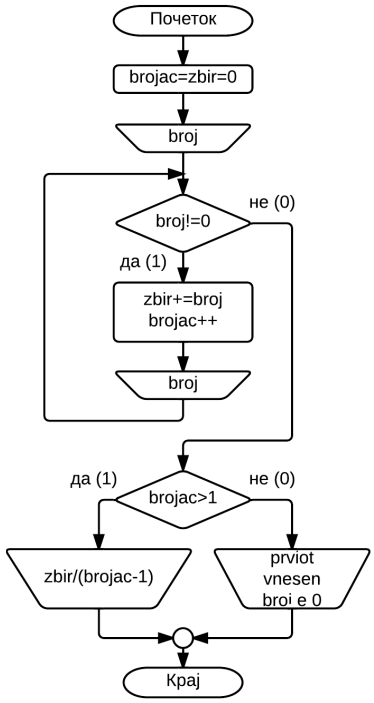
```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n, broj;
    int i = 1;
    int zbir = 0;
    cout<<"Kolku broevi ke vnesete? ";
    cin>>n;
    while (i<= n)
    {
        cout<<"Vnesi broj: "<<"\t";
        cin>>broj;
        zbir = zbir + broj;
        i++;
    }
    cout<<"Zbirot e "<<zbir<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Kolku broevi ke vnesete? 3
Vnesi broj:     6
Vnesi broj:     -12
Vnesi broj:     15
Zbirot e 9
Press any key to continue .
```

Зад. 4. 15. Измени ја претходната програма така да се собираат само позитивни броеви! Ако корисникот внесе негативен број тој нема да влезе во збирот.

Пр. 4. 52. Корисник внесува броеви, внесувањето се прекинува кога ќе внесе бројот 0. Да се најде аритметичката средина на внесените броеви!

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      int broj, brojac, zbir;
7      brojac = zbir = 0;
8      cout<<"Vnesi broj: ";
9      cin>>broj;
10     while (broj!=0)
11     {
12         zbir = zbir + broj;
13         brojac++;
14         cout<<"Vnesi broj: ";
15         cin>>broj;
16     }
17     if (brojac!=0)
18     {
19         cout<<"Aritmeticka sredina na";
20         cout<<" vnesenite broevi e ";
21         cout<<float(zbir)/brojac<<endl;
22     }
23     else
24         cout<<"Prviot vnesen broj e 0!"<<endl;
25     system("PAUSE");
26     return 0;
27 }
```




```
Vnesi broj (0 za kraj): 3
Vnesi broj: -2
Vnesi broj: 5
Vnesi broj: 6
Vnesi broj: 0
Aritmeticka sredina na vnesenite broevi e 3
Press any key to continue . . .
```

```
Vnesi broj (0 za kraj): 0
Prviot vnesen broj e 0
Press any key to continue .
```

На почетокот на програмата, пред циклусот, од корисникот се бара да го внесе првиот број (`cin>>broj`) со што се определува дали циклусот воопшто ќе се изврши. Ако првиот внесен број е 0 ќе се прикаже соодветна порака.

4.8.2 Структура за повторување do-while

Во структурите за повторување кај кои бројот на повторувања на исказите од циклусот зависи од некој услов, условот може да се испита и на крајот на циклусот. Во овој случај, во програмскиот јазик C++ се користи `do-while` структура. Оваа структура поретко се користи од структурата `while`.

Пр. 4. 53. Со кодот:

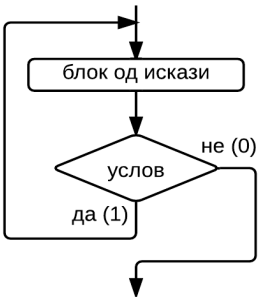
```
int i = 1;
do
{
        cout<<i<<"\t";
        i++;
}
while (i <= 10);
```

ќе се прикаже:

1 2 3 4 5 6 7 8 9 10

Основна форма на do-while циклусот е:

```
do
{
   blok_od_iskazi;
}
while (uslov);
```



Со структурата do-while условот се проверува на крајот од циклусот. Ако условот е исполнет, блокот на искази во циклусот ќе се повтори, а ако условот не е исполнет програмата ќе продолжи со следната наредба по циклусот.

Структурата do-while мора да биде напишана така што во конечен број на повторувања логичкиот израз ќе добие вредност неточно. Така обезбедува да се излезе од циклусот по конечен број на повторувања. Во спротивно циклусот би се повторувал бесконечно.

Внимавај:

Не заборавај по циклусот do-while да ставиш точка и запирка по условот.

Пр. 4. 54. Програмата испишува броеви наназад. Корисникот внесува број од кој ќе започне испишувањето.

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int n;
7      cout<<"Ke brojam nanazad!"<<endl;
8      cout<<"Od koj broj da zapocnam so odbrojuvanje? ";
9      cin >>n;
10     do
11     {
12         cout<<endl<<n;
13         n--;
14     }
15     while (n>0);
16     cout<<endl<<"Odbrojuvanjeto zavrseno!"<<endl;
17     system ("PAUSE");
18     return 0;
19 }
```

```
Ke brojam nanazad!
Od koj broj da zapocnam so odbrojuvanje? 5

5
4
3
2
1
Odbrojuvanjeto zavrseno!
Press any key to continue . . .
```

Истата задача ја решивме и со циклусот while. Да ги споредиме!

Кај `do-while` циклусот, блокот на искази од циклусот ќе се изврши барем еднаш независно дали условот е исполнет или не. Зошто?

Блок од искази кои го сочинуваат циклусот, исто така, се ставени во големи загради. Тоа се исти искази како во `while` циклусот.

На почетокот на програмата, пред циклусот, од корисникот се бара да внесе број од кој ќе започне броењето.

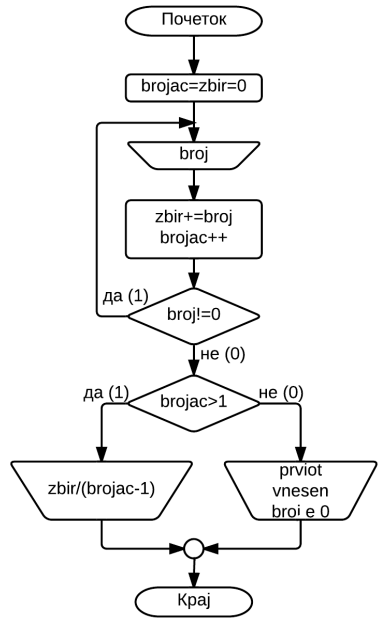
Ако корисникот внесе број 5 циклусот ќе се повтори точно 5 пати. Во секое повторување програмата го испишува бројот на екранот, потоа тој број го намалува за еден и повторно го проверува условот (`n>0`). Циклусот ќе се повторува сè додека бројот е поголем од 0. Кога бројот ќе добие вредност 0 изразот `broj>0` ќе има вредност невистина (`false, 0`) и програмата ќе продолжи со извршување на прва наредба по телото на циклусот. Која е тоа наредба?

Што ќе се случи ако корисникот внесе `0` или негативен број? Независно од тоа кој број е внесен, блокот од искази во циклусот ќе се изврши, затоа што прво се влегува во циклусот, а условот се проверува на крајот од циклусот. Ова е основната разлика помеѓу `while` и `do-while` структурата.

Ако бројот во секое повторување не се намалува за 1 тогаш логичкиот израз даден во условот (`n>0`) секогаш ќе биде точен (`true, 1`) и циклусот ќе се повторува бесконечен број пати, односно додека програмата не ја истроши работната меморија со која располага.

Пр. 4. 55. Корисник внесува броеви, внесувањето се прекинува кога ќе внесе бројот 0. Да се најде аритметичката средина на внесените броеви! Ако првиот внесен број е 0 да се прикаже соодветна порака.

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4  int main()
5  {
6      int broj, brojac, zbir;
7      brojac = zbir = 0;
8      do
9      {
10         cout<<"Vnesi broj: "<<endl;
11         cin>>broj;
12         zbir+=broj;
13         brojac++;
14     }
15     while (broj!=0);
16     if (brojac>1)
17     {
18         cout<<"Aritmeticka sredina na";
19         cout<<" vnesenite broevi e ";
20         cout<<float(zbir)/(brojac-1)<<endl;
21     }
22     else
23         cout<<"Prviot vnesen broj e 0!"<<endl;
24     system("PAUSE");
25     return 0;
26 }
```



```
Vnesi broj (0 za kraj):
-4
Vnesi broj (0 za kraj):
6
Vnesi broj (0 za kraj):
12
Vnesi broj (0 za kraj):
-3
Vnesi broj (0 za kraj):
0
Aritmeticka sredina na vnesenite broevi e 2.75
Press any key to continue . . .
```

```
Vnesi broj (0 za kraj):
0
Prviot vnesen broj e 0
Press any key to continue . . .
```

Резиме

Структурите за повторување овозможуваат некој блок од искази да се изврши одреден број пати. Секое повторување на циклусот се нарекува *итерација*.

Во while циклусот прво се проверува дали некој услов е исполнет. Ако условот е исполнет циклусот се повторува, ако условот не е исполнет тогаш не се влегува во циклусот. Исказите од циклусот може да не се извршат ниту еднаш.

Во do-while циклусот условот се проверува на крајот од циклусот. Ако условот е исполнет циклусот се повторува, ако условот не е исполнет се излегува во циклусот. Исказите од циклусот мора барем еднаш да се извршат.

Прашања:

1. Што овозможуваат структурите за повторување?
2. Како уште се нарекува структурата за повторување?
3. Што е итерација?
4. Со што е дефиниран бројот на повторувањата на циклусите?
5. Што ќе се прикаже на екранот со следните кодови?

а)
```
int i = 2;
while (i <= 10)
{
    cout<<i<<" ";
    i+=2;
}
```

б)
```
int i = 20;
while (i <= 10)
{
     cout<<i<<" ";
     i+=2;
}
```

6. Што ќе се прикаже на екранот со следните кодови?

а)
```
int i = 2;
do
{
    cout<<i<<" ";
    i+=2;
}
 while (i <= 10);
```

б)
```
int i = 20;
 do
 {
     cout<<i<<" ";
     i+=2;
 }
 while (i <= 10);
```

7. Колку пати ќе се изврши следниот циклус?

а)
```
int x= 13;
 while (x<= 13)
 {
    x++;
    cout<<x;
 }
```

б)
```
int x= 0;
 while (x<= 10)
 {
    x+=2;
    cout<<x;
 }
```

8. Програмата 10 пати го отпечати знакот *. Пополни ги празните места:

```
int x=__;
while (x<=10)
{
      _____<<"*";
      x_____;
}
```

9. Постави ги изразите во исправен редослед (напиши редни броеви на цртичките):

```
___   do                          ___   (proizvod <100000);
___   proizvod = proizvod *x;     ___   {
___   cin<<x;                     ___   proizvod=1;
___   while                       ___   }
```

Задачи:

1. Напиши програма со која се пресметува збир на реципрочните вредности на првите n природни броеви (1 + 1/2 + 1/3 + 1/4 + 1/5 + ... + 1/n)! Бројот n го внесува корисникот.
2. Напиши програма со која се пребројуваат и собираат парните броеви во опсегот од m до n!
3. Корисникот внесува n цели броеви. Напиши програма со која ќе се преброи колку од внесените броеви се позитивни, а колку негативни!
4. Корисникот внесува броеви сè додека нивниот збир е помал од 100. Напиши програма со која ќе се изброи колку броеви се внесени!
5. Корисникот внесува природен број. Напиши програма со која се пребројува колку цифри има бројот!
6. Корисникот внесува природен број. Напиши програма со која ќе се отпечатат цифрите на внесениот број почнувајќи од цифрата на единици!
7. Корисникот внесува броеви, внесувањето се прекинува кога ќе внесе 0. Напиши програма со која ќе се прикаже најголемиот број од внесените броеви!

4.9 Останати структури за повторување

4.9.1 Структура за повторување на циклус со броење на циклусите

Кога однапред е познат бројот на повторување на циклусот најчесто се користи `for` циклусот.

Пр. 4. 56. Кодот со кој се печатат првите 10 природни броеви изгледа вака:

```
for (i = 1; i <= 10; i++)
      cout<<i<<" ";
```

По клучниот збор `for` следуваат три изрази во заграда кои го контролираат циклусот

```
(i = 1; i <= 10; i++).
```

Со првиот израз се определува почетната вредност на променливата `i(i=1;)`. Вториот израз е логички израз кој кажува до кога циклусот ќе се извршува (додека `i<=10;`). Третиот израз кажува за колку вредност на променливата `i` ќе се промени.

Исказот кој се извршува во циклусот (`cout<<i<<" ";`) следува по изразот `for`. Доколку има блок од искази кои се повторуваат во циклусот, тие се стават помеѓу големи загради.

Општа форма на for циклусот е:

```
for (pocetna_vrednost; uslov; promena)
{
        blok_na_iskazi;
}
```

Секој `for` циклус има своја контролна променлива на која вредноста ѝ се менува со секое извршување на циклусот што е одредено со вредноста на прирастот (оваа вредност може да биде и негативна). Условот мора да биде логички израз кој може да добие вредност точно (`true, 1`) или неточно (`false, 0`). Блокот од искази кој се наоѓа во циклусот се извршува додека вредноста на условот е вистина. Кога условот ќе добие вредност невистина, циклусот се прекинува.

Внимавај!

По изразот for нема точка и запирка (;), како ниту по третиот израз внатре во заградите.

C++ дозволува контролната променлива да се декларира во самиот `for` исказ. На пример,

```
for (int i = 1; i <= n; i++)
      cout<<i;
```

е исто што и:

```
int i;
for (i = 1; i <= n; i++)
      cout<<i;
```

Зад. 4. 16. Што ќе се отпечати со следниов код?

```
for (i = 50; i > 20; i = i - 2)
      cout<<i<<" ";
```

Колкава е почетната вредност на контролната променлива? Колкава е промената? До кога циклусот ќе се извршува?

Ако наместо `i=i-2` се напише `i=i+2`, вредноста на променливата `i` во секоја итерација ќе се зголемува за 2 и секогаш ќе биде поголема од 20, во овој случај се добива бесконечен циклус.

Ако почетната вредност на променливата `i` е 5, циклусот нема да се изврши бидејќи условот при првата проверка е невистинит.

Пр. 4. 57. Програма со која се пресметува производ на првите n природни броеви. Бројот n го внесува корисникот.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int i, n;
    int proizvod = 1;
    cout<<"Do koj broj da mnozam? "<<endl;
    cout<<"Vnesi priroden broj: ";
    cin>>n;
    for (i = 1; i<=n; i++)
        proizvod = proizvod * i;
    cout<<"Proizvodot na broevite od "<<1;
    cout<<" do "<<n<<" e "<<proizvod<<endl;
    system ("PAUSE");
    return 0;
}
```

```
Do koj broj da mnozam?
Vnesi priroden broj: 5
Proizvodot na broevite od 1 do 5 e 120
Press any key to continue . . .
```

На променливата `proizvod` ѝ се доделува почетна вредност 1. Зошто?

Резиме

Кога однапред е познат бројот на повторување на циклусот најчесто се користи `for` циклусот.

Секој `for` циклус има своја контролна променлива на која вредноста и се менува со секое извршување на циклусот што е одредено со вредноста на прирастот. Блокот од искази кој се наоѓа во циклусот се извршува додека вредноста на условот е вистина.

Прашања:

1. Кога се користи `for` циклусот?
2. Колку пати ќе се повтори циклусот?

а)
```
for(int i=0;i<10;i++)
    cout<<i;
```

б)
```
for(int i=0;i<10;i=i+2)
    cout<<i;
```

в)
```
for(int i=0;i<=10;i++)
    cout<<i;
```

г)
```
for(int i=0;i<=10;i=i+2)
    cout<<i;
```

3. Која вредност ќе се прикаже по извршување на следниот код?

а)
```
int br=0;
for(int a=1;a<=20;a++)
    if (a%2==0)
    br++;
cout<<br;
```

б)
```
br=0;
for(int a=1;a<=20;)
    if (a%2==0)
    br++;
cout<<br;
```

4. Што ќе се прикаже со следниот код?

а)
```
for(int i=10;i>8;i--)
    cout<<"Ana";
```

б)
```
for(int i=5;i>8;i--)
    cout<<"Ana";
```

5. Напиши го while циклусот како for циклус!

```
int i=1;
while(i<=10)
{
        if(i<5 && i!=2 )
        cout<<'x';
        i++;
}
```

6. Напиши го do-while циклусот како for циклус!

```
int n = 100;
do
{
        cout<<'x';
        n-=10;
}
while(n>0);
```

Задачи:

1. Напиши програма со која ќе се прикажат сите трицифрени броеви на кои последната цифра им е 0!
2. Напиши програма со која ќе се пресмета x^n, n е природен број!
3. Корисник внесува 10 броеви. Напиши програма со која ќе се соопшти дали повеќе се внесени позитивните или негативните броеви!
4. Напиши програма со која се пребројуваат парните броеви во опсегот од m до n! Ако m>n да се заменат вредностите на променливите m и n!
5. Напиши програма со која ќе се најдат и прикажат сите делители на природен број n!

4.9.2 Дополнителни структури за повторување

Вгнездени циклуси

Понекогаш е потребно некои активности да се извршат одреден број пати, а потоа истите тие активности да се повторат уште неколку пати. На пример, ако 5 играчи играат карти, на сите 5 играчи треба да им се поделат по 4 карти. Или професор треба да прегледа тестови во 3 паралелки, во секоја паралелка прегледува онолку тестови колку ученици има во таа паралелка. Во ваквите случаеви циклусите се вгнездуваат еден во друг.

На пример,

со кодот

```
int i,j;
for (i=1;i<=3;i++)
{
   for (j=1;j<=3;j++)
   {
      cout<<i<<" "<<j<<"\t";
   }
 cout<<"\n";
}
```

на екранот ќе се прикаже:

```
1 1   1 2   1 3
2 1   2 2   2 3
3 1   3 2   3 3
```

Пр. 4. 58. Пример за вгнездени for циклуси. Со програмата се прикажува таблица на множење од 1 до 10:

```
#include <iostream>
#include <cstdlib>
#include <iomanip>
using namespace std;
int main()
{
    int i,j;
    for(i=1;i<=10;i++)
    {
        for(j=1;j<=10;j++)
        {
            cout<<setw(5)<<i*j;
        }
        cout <<"\n";
    }
    system ("PAUSE");
    return 0;
}
```

```
    1    2    3    4    5    6    7    8    9   10
    2    4    6    8   10   12   14   16   18   20
    3    6    9   12   15   18   21   24   27   30
    4    8   12   16   20   24   28   32   36   40
    5   10   15   20   25   30   35   40   45   50
    6   12   18   24   30   36   42   48   54   60
    7   14   21   28   35   42   49   56   63   70
    8   16   24   32   40   48   56   64   72   80
    9   18   27   36   45   54   63   72   81   90
   10   20   30   40   50   60   70   80   90  100
Press any key to continue . . .
```

При влезот во надворешниот циклус, контролната променлива `i` добива вредност 1, се проверува условот на надворешниот циклус (`i<=10;`), па ако условот е исполнет се влегува во внатрешен циклус.

При влезот во внатрешниот циклус, контролната променлива `j` добива вредност 1, се проверува условот на надворешниот циклус (`j<=10;`), па ако условот е исполнет се извршуваат искази од внатрешниот циклус (`cout<<i*j<<"\t";`).

Исказите од внатрешниот циклус ќе се извршуваат се додека условот (j<=10;) е исполнет. Кога овој услов нема да биде исполнет, се излегува од внатрешниот циклус и повторно започнува да се извршува надворешниот циклус. Вредноста на контролната променлива за надворешниот циклус се менува (`i++`), се проверува условот на надворешниот циклус (`i<=10;`), па ако тој е исполнет се влегува во внатрешен циклус. Сега се повторува опишаната постапка на извршување на внатрешниот циклус.

Сè ова се повторува додека условот за надворешниот циклус е исполнет, кога овој услов нема да биде исполнет се излегува од надворешниот циклус. За секоја вредност на контролната променлива `i` на надворешниот циклус се извршува целиот внатрешен циклус.

Забелешка:

За испишувањето да биде во правилни колони, се користи операторот за форматирано печатење setw(5) со кој се одредува колкав простор ќе се предвиди за испишување на податоците. Овој оператор е вклучен во библиотеката iomanip, па таа библиотека е вклучена со наредбата include.

Исказите break и continue

Се запознавме со сите три структури за повторување во јазикот C++ и тоа е вообичаен начин на кој овие структури се користат. Сепак, начинот на нивното извршување може да се смени со помош на `break` и `continue` исказите.

Со исказот `break` се прекинува понатамошното извршување на циклусот во кој овој исказ се наоѓа. Со овој исказ веќе се запознавме во `switch-case` структурата.

Со исказот continue се прекинува извршување само на моментната итерација на циклусот во кој овој исказ се наоѓа (таа итерација се прескокнува).

Пр. 4. 59. Пример за користење на исказот break:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    for (n=10; n>0; n--)
    {
        cout << n << ", ";
        if (n==5)
        {
            cout <<"\nOdbrojavanjeto e prekinato!
            break;
        }
    }
    system ("PAUSE");
    return 0;
}
```

```
10, 9, 8, 7, 6, 5,
Odbrojavanjeto e prekinato!
Press any key to continue . . .
```

Пр. 4. 60. Пример за користење на исказот continue:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;
    for (n=10; n>0; n--)
    {
        if (n==5)
        continue;
        cout << n << ",";
    }
    cout <<"\nOdbrojuvanje bez brojot 5!\n";
    system ("PAUSE");
    return 0;
}
```

```
10,9,8,7,6,4,3,2,1,
Odbrojuvanje bez brojot 5!
Press any key to continue . . .
```

Исказот continue може да предизвика бесконечен циклус, па се препорачува наместо овој исказ да се користи исказот за разгранување.

Исказот goto

Исказот goto е исказ за безусловен скок. Овој исказ овозможува извршувањето на програмата да продолжи од избраниот исказ. Исказот со кој понатаму се продолжува може да биде каде било во кодот, но мора да биде означен со симболичко име по кое се става знакот две точки (:).

Во општ случај наредбата goto се користи во следната форма:

```
ime: iskazot_na_koj_se_pominuva;
...
goto (ime);
```

Пр. 4. 61. Пример за користење на исказот `goto`:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int a, P;
    vnes: cout<<"Vnesi strana na kvadratot: ";
    cin>>a;
    if (a<0)
    {
        cout<<"Strana ne moze da bide negativna!\n";
        goto vnes;
    }
    P=a*a;
    cout <<"Plostinata na kvadratot e "<<P<<"\n";
    system ("PAUSE");
    return 0;
}
```

```
Vnesi strana na kvadratot: -6
Strana ne moze da bide negativna!
Vnesi strana na kvadratot: 6
Plostinata na kvadratot e 36
Press any key to continue . . .
```

Бесконечен циклус

Може да се случи циклус да се извршува бесконечно пати ако услов во циклусот секогаш е исполнет. Програма во која е бесконечен циклус може да се прекине со затворање на прозорецот во кој програмата се извршува.

Пр. 4. 62. Пример на бесконечен циклус:

```
for(i=3;i<5;)
        cout<<”Beskonecen ciklus”;
```

За да одбегнеш бесконечен циклус:

- води сметка услов во циклусот некогаш да мора да добие вредност неточно,
- во исказот for, во заградите, наведи ги сите три изрази,
- одбегнувај во циклусот да менуваш вредност на контролната променлива.

Резиме

Дозволено е циклуси да се вгнездуваат еден во друг. При тоа постои внатрешен и надворешен циклус.

Со исказот `break` се прекинува понатамошното извршување на циклусот во кој овој исказ се наоѓа. Со исказот `continue` се прекинува извршување само на моментната итерација на циклусот во кој овој исказ се наоѓа. Исказот `goto` е исказ за безусловен скок. Овој исказ овозможува извршувањето на програмата да продолжи од избраниот исказ.

Вештини што треба да ги усовршиш:

Да познаваш и да користиш дополнителни структури за повторување.

Да препознаваш и да знаеш како да одбегнеш бесконечен циклус.

Да напишеш програма со користење на досега изучените техники.

Прашања:

1. Дали е дозволено вгнездување на циклуси во C++?
2. Кога се користи исказот break?
3. Кога се користи исказот continue?
4. Кога се користи исказот goto?
5. Што е бесконечен циклус?
6. Како може да се одбегне бесконечен циклус?
7. Спореди ги следниве примери за користење на исказот continue:

а)

```
int n=11;
while(n>1)
{
      n--;
      if (n==5)
            continue;
      cout << n << ", ";
}
cout << "\nOdbrojuvanje bez brojot 5!";
```

Екран:

```
10, 9, 8, 7, 6, 4, 3, 2, 1,
Odbrojuvanje bez brojot 5!
```

б)

```
int n=10;
while(n>=1)
{
      if (n==5)
            continue;
      cout << n << ",";
      n--;
}
cout << "\nOdbrojuvanje bez brojot 5!";
```

Екран:

```
10, 9, 8, 7, 6,
Odbrojuvanje bez brojot 5!
```

Зошто во вториот пример одбројувањето е прекинато? Каде е грешката?

Задачи:

1. Напиши програма која ќе ги прикаже делителите на броевите од 10 до 100!
2. Корисник внесува 10 броеви. Напиши програма со која ќе се пребројат и ќе се најде збир само на позитивните броеви! Во случај корисникот да внесе негативен број да се прикаже соодветна порака и да се прекине циклусот!

```
Vnesi broj: 5
Vnesi broj: -2
Vnesi broj: 10
Vnesen e negativen broj, toj ne se broi i ne se sobira!
Vneseni se 2 broevi, nivniot zbir e 15!
```

3. Напиши програма со која се печатат броеви од 1 до n! Бројот n го внесува корисникот. Доколку корисникот внесе број помал или еднаков на 1 да се отпечати соодветна порака и корисникот да се упати повторно да го внесе бројот n!

4.10 Примери за посложени алгоритми и програми

Пр. 4. 63. Да се провери дали внесениот број е прост!

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n;    // brojot sto se vnesuva
    int k;    // kandidat za delitel
    cout<<"Vnesi broj n(n>1): ";
    cin>>n;
    if(n%2==0 && n!=2) //(1)
    {
        cout<<"Brojot ne e prost!\n";
    }
    else     //(2)
    {
        k=3; //prviot kandidat za delitel e brojot 3
        while(k <= n/2) //dodeka kandidatot e pomal od n/2
        {
            if( n%k==0) //prasuvame dali toj broj e delitel na brojot n
            {
                cout<<"Brojot ne e prost!\n";
                //ako e, pecatime deka n ne e prost broj
                break; // i go prekinuvame ciklusot
            }
            k=k+2;
            //inaku sledniot kandidat za delitel e sledniot neparen broj
        }
        if(k>n/2)
            //ako ciklusot ne e prekinat so naredbata break, brojot n e prost
            cout<<"Brojot e prost!\n";
    }
    system ("PAUSE");
    return 0;
}
```

Објаснување:

`//(1)` Сите парни броеви се делат со 2 па тие не се прости броеви. Исклучок е бројот 2 кој е прост број. Ако корисникот внесе парен број кој е различен од 2, веднаш може да се отпечати дека тој број не е прост.

`//(2)` Треба да се провери дали внесениот број се дели со уште некој број освен со 1 и со n. Бидејќи се работи само со непарните броеви, кандидати за нивните делители можат да бидат само броевите 3, 5, 7 и т.н. заклучно со најголемиот број кој е помал од n/2 (ниту еден број нема делител поголем од n/2, а да тоа не е бројот n). Во некој циклус, на пр. во циклусот while, може да се провери дали кандидатите за делител навистина се делители на бројот n (if(n%k==0)). Ако има таков делител, веднаш штом тој ќе се најде може да се отпечати порака дека бројот n не е прост број и циклусот ќе се прекине (break;).

Зад. 4. 17. Да се најдат сите прости броеви од бројот 1 до бројот n кој го внесува корисникот!

Пр. 4. 64. Да се најде најголемиот заеднички делител на два цели броја x и y!

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int x,y;
    cout << " Vnesi go prviot broj: "<<endl;
    cin >> x ;
    cout << " Vnesi go vtoriot broj: "<<endl;
    cin >> y;
    while (x!=y)
    {
            if (x>y)
                x=x-y;
        else
                y=y-x;
    }
    cout<<"NZD e: "<<x<< endl;
    system ("PAUSE");
    return 0;
}
```

Објаснување: Алгоритамот на Евклид.

Овој алгоритам е заснован на својството дека НЗД на два броја е еднаков со НЗД на помалиот од двата броја и нивната разлика. На пример:

НЗД(18,12) = НЗД (12,6) = НЗД(6,6) = 6

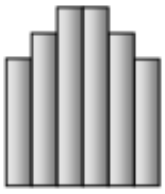
Постапката се повторува сè додека не се добијат два еднакви броја, и тој број претставува НЗД (`while (x!=y)`).

Поголемиот број се заменува со разликата на поголемиот со помалиот број:

Поминување 1:	x=18, y=12	18>12 → x=18-12=6, y=12
Поминување 2:	x=6, y=12	6<12 → x=6, y=12-6=6
Поминување 3:	x=6, y=6	6=6 → NZD=6

4.11 Задачи за талентираните ученици:

4.11.1 Линеарна структура

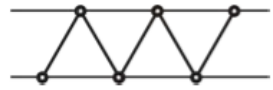


1. За училишна приредба Ана и Марио треба да направат замок од 6 штици поредени една до друга како на сликата. Првата штица е со должина k, втората е за 20 cm подолга од првата, третата е за 20 cm подолга од втората, четвртата има иста должина со третата, петтата има иста должина со втората и шестата има иста должина со првата штица. Напиши програма со која за дадена должина на првата штица се пресметува вкупната должина на сите штици.

   Пример:

   Влез: 50 Излез: 420

2. Учениците од I година на излет треба да направат висечки мост. Мостот се состои од две паралелни јажиња кои се врзани за крајбрежјата. Третото јаже треба со јазли да се прицврсти за првите две јажиња во цик-цак така да формира одреден број на рамнокраки триаголници. Напиши програма со која ќе се пресмета колку вкупно јазли треба да се направат ако треба да се формираат `t` триаголници!



Пример:

Влез: 4 Излез: 6

3. Во такси во исто време влегле тројца патници. Кога првиот патник излегол, таксиметарот покажувал C1 денари, кога излегол вториот патник, цената била C2 денари и кога излегол третиот патник цената била C3 денари. Колку треба да плати секој од патниците ако секој плаќа пропорционално, т.е. кога првиот патник ќе излезе, тој треба да плати само третина од C1 денари за првиот дел на патот?

Пример:

Влез: 30 50 100 Излез: 10 20 70

4. Банкомат издава банкноти од 1000, 500, 100, 50 и 10 денари. Напиши програма со која ќе се пресмета колку вкупно банкноти банкоматот ќе исплати за n денари, ако се исплатува по принцип на најмалку исплатени банкноти.

Пример:

Влез: 3780 Излез: 10

4.11.2 Разгранети структури

1. Горан сака во една соба да постави маса за билијард. Собата има форма на квадрат со страна со должина d, а масата мора да има правоаголна форма и мора да има одредени димензии c1 и c2. Од масата до ѕидовите мора да има одредено растојание. Напиши програма со која ќе се одреди дали масата може да ја собере во собата. Во програмата се внесуваат следните податоци: димензија на собата d, димензии на масата за билијард c1 и c2, најмало растојание на масата од ѕидовите m.

Пример: Влез: d = 4.5 c1 = 2.1 c2 = 2.6 m = 1 Излез: ne moze

Влез: d = 6.5 c1 = 2.1 c2 = 2.6 m = 0.6 Излез: moze

2. Марија учи деливост на броеви. Најлесно ѝ оди деливост со бројот 5, затоа што е доволно да ја види последната цифра на некој број за да дознае дали тој е делив со 5. Јана ѝ помага на Марија така што ѝ задава број x, а Марија мора да пронајде најмал број делив со 5 кој е помал од x и најголем број делив со 5 кој е поголем од x.

Напиши програма која ќе ѝ помогне ма Јана да ги провери одговорите на Марија!

Примери:

Влез: 12 Излез: 10 15

Влез: 27 Излез: 25 30

Влез: 15 Излез: 10 20

3. Матеј треба да оди во САД на натпревар по информатика. Тој дознал дека Американците поинаку го прикажуваат времето, како што се гледа во следнава табела:

македонско време	американско време	македонско време	американско време
00:00	12 AM	12:00	12 PM
01:00	1 AM	13:00	1 PM
02:00	2 AM	14:00	2 PM
03:00	3 AM	15:00	3 PM
04:00	4 AM	16:00	4 PM
05:00	5 AM	17:00	5 PM
06:00	6 AM	18:00	6 PM
07:00	7 AM	19:00	7 PM
08:00	8 AM	20:00	8 PM
09:00	9 AM	21:00	9 PM
10:00	10 AM	22:00	10 PM
11:00	11 AM	23:00	11 PM

Напиши програма која на Матеј ќе му помогне да претвора македонско време во американско.

Пример:

Влез	Влез
22	8
Излез	Излез
10 PM	8 AM

4.11.3 Структури со повторување

1. Маја треба да направи цветови од хартија. Секој цвет треба да биде со различна боја и ливчињата на секој цвет треба да бидат со иста боја. Од еден лист хартија можат да се исечат 4 ливчиња за цветови. Напиши програма со која за даден број на цветови и за дадени броеви на ливчињата на секој цвет се определува вкупен број на потребни листови хартија.

   Влезни податоци: Првата линија содржи ненегативен цел број n кој претставува број на цветови кои треба да се направат, во секоја од следниве n линии се наоѓа ненегативен цел број кој претставува број на ливчиња во секој од цветовите.

   Излези податоци: Вкупен број на листови потребни за правење на цветови.

   Пример.

   Влез:
   5
   5
   3
   9
   4
   6

   Излез:
   9

2. Во текот на учебната година Јана има работено n тестови по математика и ја интересира дали ќе има петка за крајот на годината. Јана ги знае поените по сите n тестови и знае колку вкупно поени се потребни за петка. Напиши програма со која Јана ќе пресмета дали ќе добие петка!

   Преку тастатурата се внесуваат следниве податоци:

   - број на тестови (n)
   - поени од сите тестови
   - број на потребните поени за петка

   Пример:

Влез	Влез
3	3
8	10
9	9
6	9
27	8
Излез	Излез
не	да

3. Напиши програма со која се одредува палиндром на даден природен број. Палиндром е број кај кој редоследот на цифри е обратен, на пр. на бројот 123 палиндром е бројот 321.

   Пример:

   Влез: 35472 Излез: 27453

4. Горјан ги послужил другарите со чоколадо во форма на правоаголник чии димензии се дадени со природни броеви a и b. Тој им кажал на другарите да кршат само квадрати. Другарите еден по еден откршувале квадрати, но секој земал најголем можен квадрат. Напиши програма со која се одредува колку другари се послужиле со чоколадото!

   Примери:

   Влез: a=12 b=7 Излез: 6

   (Вкупно 6 другари се послужиле со чоколадо, тие по ред земале квадрати со страните 7, 5, 2, 2, 1, 1.)

   Влез: a=7 b=7 Излез: 1

5. Во една гајба се наоѓаат m килограми јаболко и n килограми круши (m и n се природни броеви). Напиши програма која за дадени вредности на m и n ќе го прикаже најмалиот можен број на гајби потребни за препакување на јаболките и крушите така што гајбите ќе бидат со иста маса и јаболките и крушите треба да бидат одвоени.

   Пример:

   Влез: m=16 n=12 Излез: 7