# PROGRAMIRANJE C++

GIMNAZIJA „MEŠA SELIMOVIĆ“ TUZLA

# ŠTA JE PROGRAMIRANJE, A ŠTA PROGRAM?

**Programiranje** je termin pod kojim se najčešće podrazumjeva kreiranje računarskih programa, što uključuje teorijsku detaljnu razradu problema, iznalaženje konceptualnog rješenje i implementaciju korištenjem nekog od programskih jezika.

Pojam programiranje obuhvata i sve ostale procese čiji je cilj automatizovanje, rješavanje tipa problema, pa tako postoji programiranje za televizijske uređaje, veš mašine, mobilne telefne, kao i matematičko programiranje i dr.

- Programiranje predstavlja definisanje niza koraka koji se obavljaju jedan za drugim u cilju izvršenja određenog zadataka.
- Programiranje predstavlja davanje uputa računaru šta tačno da uradi.
- Upute dajemo računaru u nama razumljivom obliku koji računar ne razumije. Da bi računar razumio naše upute, potrebno je naš program pretvoriti u računaru razumljiv oblik (1 i 0 – binarna logika) tj. **kompajlirati ga.**
- **Program predstavlja skup naredbi koje prilikom izvršavanja obavljaju neki posao.**

# OD IZVORNOG KODA DO IZVRŠNOG PROGRAMA

- Izvorni kod (source code) je program napisan u bilo kojem programskom jeziku visokog nivoa.
- On se može pisati u bilo kem programu za uređivanje teksta (text editor), ali većina današnjih prevoditelja i povezivača (linker) se isporučuje kao cjelina zajedno sa ugrađenim programom za upis i ispravljanje izvornog koda. Ovakve programske cjeline poznate su pod imenom ***IDE-Integrated Development Environment . Integrisana razvojna okruženja.***

- Nakon ispisanog izvornog koda, on se pohranjuje u datoteku izvornog koda na disku dobijaju ekstenziju zavisno od programskog jezika u našem slučaju nastavak je .cpp, .cp ili .c.
- Poslije toga slijedi **prevođenje (kompajliranje)** izvornog koda (to se postiže klikom na izborniku naredbi s nekog menija ili kombinacijom određenih tipki sa tastature.
- Prevoditelj (kompajler) prilikom prevođenja provjerava sintaksu napisanog izvornog koda i u slučaju grešaka ispisuje odgovarajuće poruke. Greške koje se pojave u ovoj fazi nazivaju se Compile Time Errors. Kada prevođenje koda prođe bez grešaka onda se može pristupiti pokretanju koda/programa.
- Osim C++ programskog jezika popularni su Java, C,C#, Python i mnogi drugi programski jezici.
- Važno je napomenuti da HTML, CSS, JavaScript i slični ne spadaju u programske jezike već u opisne odnosno skriptne jezike.

# OSNOVNI POJMOVI I PRAVILA U PROGRAMIRANJU

- PREVODITELJ (COMPILER) – nam omogućava prevođenje napisanog koda (source code) u izvedbeni kod
- PROGRAM ZA UREĐIVANJE TEKSTA(EDITOR) – služi za pisanje teksta, odnosno koda. Danas mnogi editori dolaze sa predinstaliranom podrškom za isticanje ključnih riječi. Npr. Upisom riječi #INCLUDE, tekst dobija plavu boju što sugeriše da smo upisali jednu od ključnih riječi.
- PROGRAM ZA OTKRIVANJE GREŠAKA (DEBUGGER) – dolazi integrisan unutar VS C++. Postoje dvije vrste grešaka: sintaksičke i logičke. DEBUGGER otkriva sintaksičke, ali ne i logičke.
- SINTAKSA – je skup pravila kojih se moramo pridržavati prilikom korištenja određenog programskog jezika.

- SINTAKSIČKA GREŠKA – je greška na koju debugger upozorava i neće dopustiti pokretanje aplikacije dok se ista ne ukloni.
- LOGIČKA GREŠKA – je greška koju je teže otkriti jer debugger nije u mogućnosti otkriti ove greške.
- REDOSLJED IZVRŠAVANJA PROGRAMA – prevoditelj uvijek izvršava naredbe od prve linije do zadnje, gledano u vertikalnom smjeru.
- NUMERACIJA KODNIH LINIJA – kako bi se lakše snalazili u tekst editoru preporučuje se upaliti numerički prikaz linija. Opciju uključujemo tako što odaberemo TOOLS-OPTION-TEXT EDITOR-ALL LANGUAGES i upalimo opciju LINE NUMBER.
- INTELLI SENSE – je pomoć pri kucanju koda, kada u programu Visual Basic prilikom kucanja koda se na osnovu kucanja slova automatski nudi predviđena riječ (naredba / komanda).

- BIBLIOTEKE – su datoteke u kjima se nalaze već prevedene gotove funkcije i podaci. Biblioteke se obično isporučuju zajedno sa prevoditeljem, ali ih programer može i sam kreirati po potrebi. Upotrebom biblioteka se izbjegava ponavljanje pisanje često korištenih operacija. Tipičan primjer za to je biblioteka matematička funkcija u kojoj su definisane funkcije poput onih matematičkih (trigonometrijske ili eksponencijalne).

- NAREDBA CIN i COUT (UČITAVANJE SA KONZOLE I ISPIS U KONZOLU) – ove dvije naredbe se jako često koriste. Naredba CIN predstavlja učitavanje iz konzole (CONSOLE  IN), a naredba COUT predstavlja pisanje u konzolu (CONSOLE OUT). Primjer: za upis koristiomo cin>>upis; , a za ispis cout<<ispis;.

- ISPIS TEKSTA I ULOGA DVOSTRUKIH NAVODNIKA – ukoliko je potrebno neki tekst ispisati na ekran, koriste se dvostruki navodnici („“). Sve što se nalazi unutar ovih navodnika se tretira kao TEKST.

- SIMBOL TAČKA – ZAREZ (;) – u programskom jeziku C++ za završavanje naredbe koristi se simbol tačka – zarez (;). U jednoj komandnoj liniji moguće je imati više naredbi, jer je svaka odvojena sa simbolom tačka – zarez (;).

# DATOTEKE ZAGLAVLJA (HEADER FILE)

- Datoteke zaglavlja mogu da sadrže funkcije, konstante, klase, objekte i predloške/šablone (templates). Gotovo svi programi u C++ zahtjevaju uključivanje neke od datoteka zaglavlja. Neke od datoteka zaglavlja su standardne, ali korisnici mogu kreirati i vlastite datoteke zaglavlja.
- Standardne datoteke zaglavlja su:

Standardne C++ datoteka zaglavlja su:

<algorithm>, <bitset>, <cassert>, <cctype>, <cerrno>, <cfloat>, <ciso646>, <climits>, <clocale>, <cmath>), <complex> , <csetjmp>, <csignal>, <cstdarg>, <cstddef>, <cstdio> , <cstdlib> , <cstring>, <ctime> , <cwchar>, <cwctype>, <deque> , <exception>, <fstream> , <functional> , <iomanip> , <ios> , <iosfwd> , <iostream> , <istream>, <iterator>, <limits>, <list> , <locale> , <map> , <memory>, <new> , <numeric> , <ostream> , <queue> , <set>, <sstream>, <stack> ,<stdexcept> , <streambuf>, <string> , <strstream> , <typeinfo>, <unordered_map>, <unordered_set>, <utility>, <valarray>, <vector>

Slika 10: Standardne datoteke zaglavlja

# DATOTEKA ZAGLAVLJA **<iostream>**

- U ovoj datoteci su deklarisani objekti koji kontrolišu upis i ispis podataka pomoću apstrakcije (stream) koja predstavlja uređaje na kojima se ulazne/izlazne operacije izvode.
- Ime ove datoteke izvedeno je iz engleskih naziva za upis i ispis – Input/Output Stream.
- Stream je zapravo izvor ili odredište niza karaktera nedefinisane dužine.
- Datoteka <iostream> je često jedina datoteka koju moramo uključiti u naš program.
- Datoteka /<iostream> koristi objekte cin>> i cout<<.

# DATOTEKA ZAGLAVLJA **<cmath>**

- Ova datoteka sadrži deklaracije i definicije skupa funkcija za računanje matematičkih operacija i transformacija. Ova datoteka sadrži trigonometrijske funkcije, hiperboličke fumkcije, eksponencijalne i logaritamske funkcije, funkciju za izračunavanje kvadratnog korijena, funkciju za izračunavanje potencije nekog broja, apsolutne vrijednosti nekog broja, te za zaokruživanje brojeva.
- Trigonometrijske funkcije:cos, sin, tan, acos, asin, atan, atan2;
- Hiperboličke funkcije:cosh, sinh, tanh;
- Kvadriranje i vadratni korijen:pow,sqrt;
- Zaokruživanje, apsolutna vrijednost: ceil, abs, floor, fmod.

# NAJČEŠĆE GREŠKE

- Prilikom kucanja može se pojaviti greška prilikom kucanja programa. Greške mogu da budu logičke i sintaksičke. Kompajler će nam otkriti sintaksičke, ali logičke greške kompajler neće pronaći. Neke od najčešći grešaka sa kojim se možemo sresti su:
- Dijeljenje sa (0). Ova greška uzrokuje trenutno završavanje programa bez da je program obavio svoj zadatak i spada u logičku grešku.
- Izostavljanje biblioteke iostream onemogućuje unos podataka s tastature kao i njihov ispis na ekranu...Ovo je sintaksička greška.
- Izostavljanje (;) na kraju iskaza i/ili izraza onemogućuje kompajleru da prepozna iskaz/izraz. Ovo je sintaksičkoj (jezičkoj) greški.

- Pokušaj korištenja modula (%) na necjelobrojnim vrijednostima.
- Upotreba operatora jednako (==), različito (!=), veće ili jednako(>=) i manje ili jednako (<=) sa razmacima između simbola (= =, ! =, > =, < =).
- Zabuna oko operatora pridruživanja (=) i operatora usporedbe (==) neče prouzročiti sintaksičku grešku koju će kompajler prepoznati, ali će prouzročiti logičku pogrešku i samim tim pogrešan krajnji rezultat.
- Zaboravljanje vitičastih zagarada ({}) na početku i na kraju bloka mogu prouzročiti i sintaksičke i logičke greške.
- Izostavljanje biblioteke *cmath* u programu koji koristi matematičke funkcije poput kvadriranja, kvadratnog korijena, apsolutne vrijednosti, funkcije tangensa i kontagensa..)
- Zaboravljen iskaz *return* u funkciji koja vraća vrijednost.
- Navođenje iskaza *return* u funkciji bez povratne vrijednosti (funkciji tipa void).

# IDENTIFIKATORI

- Identifikatori su imena kojima se imenuju varijable, promjenjive i druge sintaksne kategorije (funkcije, klase).
- Oni služe za zapis imena varijabli, funkcija i korisničkih tipova. Svaki indentifikator može sadržavati velika i mala slova engleske abecede, cifre i simbol podcrte (_) no uz uslov da:
  - Prvi znak imena odnosno identifikatora mora biti ili slovo ili podcrta;
  - Indentifikator ne smije biti jednak nekoj od ključnih riječi jezika ali je može sadržavati.
  - Npr.: broj Pi kojem dodjeljujemo vrijednost 3.14...nakon par mjeseci ćemo znati na šta se odnosi ta varijabla, ali ako je nazovemo x vjerovatno ćemo za par mjeseci zaboraviti....

# KLJUČNE RIJEČI

- Svaki programski jezik koristi vlastiti, ograničeni skup riječi koje imaju posebna značenja. Takve čemo riječi nazvati ključne riječi.
- One su definisane pravilima jezika, ne mogu se mijenjati niti koristiti kao identifikator.
- Ključne riječi za C++ jezik su: ***bool***, ***break***, ***catch***, ***char***, ***class***, itd... ne smiju se koristiti kao identifikatori i zapisuju se malim slovima.

# NEKE OD KLJUČNIH RIJEČI...

U nastavku su prikazane neke od ključnih riječi.

```
auto     const      double   float    int        short    struct   unsigned
break    continue   else     for      long       signed   switch   void
case     default    enum     goto     register   sizeof   typedef  volatile
char     do         extern   if       return     static   union    while

asm          dynamic_cast   namespace   reinterpret_cast   try
bool         explicit       new         static_cast        typeid
catch        false          operator    template           typename
class        friend         private     this               using
const_cast   inline         public      throw              virtual
delete       mutable        protected   true               wchar_t
```

Slika 11: Neke od ključnih riječi C++ programskog jezika

# VARIJABLE

- Varijable su imena memorijskih lokacija koje sdrže pripadajuće vrijednosti. To su veličine čije se vrijednosti mijenjaju u toku izvršavanja programa.

# DEKLARACIJA VARIJABLI

- Deklarisati varijablu znači govoriti računaru da ćemo koristiti varijablu nekog imena.
- Npr.: imamo varijablu x  tipa integer (int x;) to znači da stvaramo varijablu koja je tipa integer i koja će se zvati x.

- int x;
- float var;
- bool rezultat;

# INICIJALIZACIJA VARIJABLE

- Kada napravimo varijablu potrebno joj je dodijeliti neku vrijednost tj. pridružiti joj neku vrijednost. Postupak dodjeljivanja vrijednosti varijabli zove se inicijalizacija. Primjeri:
  - int x=5;
  - float var=3.25;
  - bool rezultat=false;

# KONSTANTE

- Konstanta je veličina koja u toku izvršavanja programa ima samo jednu vrijednost, koja se ne može mijenjati. Definiše se ključnom riječju...***const***:

  - const int x=5;
  - const float var=3.25;

# TIPOVI PODATAKA

- Cijeli brojevi
- Realni brojevi
- Boolean vrijednosti
- Znaci
- Stringovi (niz znakova)
- Nad svim ovim tipovima podataka su omogućene izvjesne operacije, a upravo s činjenicom šta je raspoloživo ili omogućeno na mašini vezana je i ideja tipa podatka.

- Matematički gledano tip podatka je konačan skup elemenata sa kolekcijom operacija na tom skupu. Te se funkcije često nazivaju i standardne funkcije.
- Osnovni tipovi definisani sa skupom i funkcijama na skupu.

## Cjelobrojni tip podatka – integer – ključna riječ int

```
int x=5;
```

- Podtipovi koji se mogu pridodati su: short, long i unsigned.
- Tip short int omogućava manji raspon brojeva , a long int omogućava veći raspon brojeva, samim time i zauzimanje memorije je manje i veće.
- Za korištenje striktno pozitivnih brojeva koristi se unsigned int.
- Standardne operacije zajedno sa skupom skupa Z (skup cijelih brojeva):

**+Z, *Z, -Z, /Z, =Z, Z>Z, Z<Z**

# Realni tip podataka – real – ključna riječ float

```
float x=5.5;
```

- Radi se o cijelim brojevima s pomičnim zarezom (tzv.decimalni brojevi) i ovaj tip nam omogućava veći raspon brojeva. Ovakav tip nam daje veću preciznost. Decimalni zarez se ispisuje tačkom (.), a ne zarezom (,).
- Standardne operacije zajedno sa skupom skupa R (skup realnih brojeva):

**+R, *R, -R, /R, =R, R>R, R<R**

# Realni tip podataka – double – ključna riječ double

```
double x=5.5555968;
```

- Double tip podataka daje okvirno duplo veću preciznost od float tipa, ali zauzima i duplo više memorije.

# Logički tip podatka – boolean – ključna riječ bool

- Ovaj tip podataka ima samo dvije vrijednosti. To su vrijednost true (istina) i false (laž). Sinonim za te vrijednosti su ujedno cifre 1 i 0 gdje cifra 0 (nula) predstavlja laž (false) a istina se predstavlja cifrom 1 (jedan). Varijabli koja je tip bool možemo dodijeliti i vrijednost 2,3,5 itd.ali će ta vrijednost biti zapisana kao 1 (jedan) odnosno kao istina.
- Skup B (true, false) zajedno sa standardnim operacijama

NOT (!), AND () i OR()

bool X=true; ili bool X=1; ili bool X=false<, ili bool X=0;

- Znakovni tip podataka – character - ključna riječ char

char x='a';  ili  char x='1';  ili  char x='#';

- Znakovni tip podataka predstavalja znakove (karaktere), a ne brojeve i ne cifre. Može sadržavati isključivo jedan znak (cifru, slovo ili simbol). Prilikom inicijalizacije ovog tipa podataka koriste se jednostruki navodnici.

- String tip podataka – string – ključna riječ string

string ime =„program“;  string broj=„3“;  string broj=„3.14“;

- String tip podataka je potrebno posebno uključiti u našu aplikaciju uključivanjem biblioteke.

#include <string>

- Ovaj tip podatka ne spada u primitivne tipove i njih ćemo posebno obraditi naknadno. Ovaj tip podatka predstavlja skup znakova (simboli, slova i brojeva). Sve što se nalazi unutar dvostrukih navodnika predstavlja tekst.

- U smislu računarskog programiranja može se reći da tip podataka određuju:
  - skup vrijednosti koje podatak može imati;
  - memorijski prostor potreban za smještanje podataka;
  - operacije koje mogu da se vrše nad tim podatkom;
- Tip podatka vidljiv je iz deklaracije.
- Svi podaci (aritmetički izrazi, rezultat izračunavanja izraza, argumenti procedura i funkcija...) moraju imati tip

| Tip podatka | Ključna riječ | Potrebno memorije (bajt) | Opseg vrijednosti |
|---|---|---|---|
| Character | Char | 1 | - 128 do 127 |
| Integer | Int | 4 | -2,147,483,648 do 2,147,483,647 |
| Short integer | Short | 2 | -32768 do 32767 |
| Long integer | Long | 4 | -2,147,483,648 do 2,147,483,647 |
| Unsigned long integer | Unsigned long | 4 | 0 do 4,296,967,295 |
| Single precision floating point | Float | 4 | 1.2E-38 DO 3,4E38 |
| Double precision floating point | double | 8 | 2.2E08 DO 1.8E308 |

# OPERATORI I IZRAZI

- Osnovni operatori kji se primjenjuju na varijable su sabiranje, oduzimanje, množenje i dijeljenje. Oni se mogu primjeniti samo na jednu vrstu i ovu se unarni ili se mogu primjeniti na dvije vrijednosti u tom slučaju ih nazivamo binarnim.
- Operatore dijelimo na:
  - aritmetički
  - relacijski
  - logički

# ARITMETIČKI OPERATORI

- Aritmetički operatori se primjenjuju nad numeričkim operandima tj. cjelobrojnim, realnim i znakovnim operandima (znakovna konstanta se svodi na cjelobrojni tip).

| Tip operatora | | Značenje |
|---|---|---|
| UNARNI | ++ | Inkrement (povećaj za jedan) |
| | -- | Dekrement (umanji za jedan) |
| BINARNI | + | Sabiranje |
| | - | Oduzimanje |
| | * | Množenje |
| | / | Dijeljenje |
| | % | Modulo (ostatak cjelobrojnog dijeljenja) |

- ***Cjelobrojno dijeljenje*** – odsjecanje decimalnih cifara kako bi se dobio cjelobrojni rezultat. Ako je rezultat dijeljenja negativan, onda način zaokrućivanja rezultata ovisi o implementaciji. Najčešće je to ipak odsjecanje, tj. zaokruživanje prema nuli.

3/2=1

3.0/2=1.5

-3/2=-1

- ***Ostatak cjelobrojnog dijeljenja***
- Operacija modulo (%) djeluje na cjelobrojnim operandima i kao rezultat daje ostatak pri cjelobrojnom dijeljenju operanada.

X=10, Y=3

X/Y=3

X%Y=1

- **Operatori Inkrement i Dekrement**
- Operator inkrementa ++ povečava vrijednost varijable za jedan.

  Izraz x++;  ekvivalentan je izrazu  x=x+1;

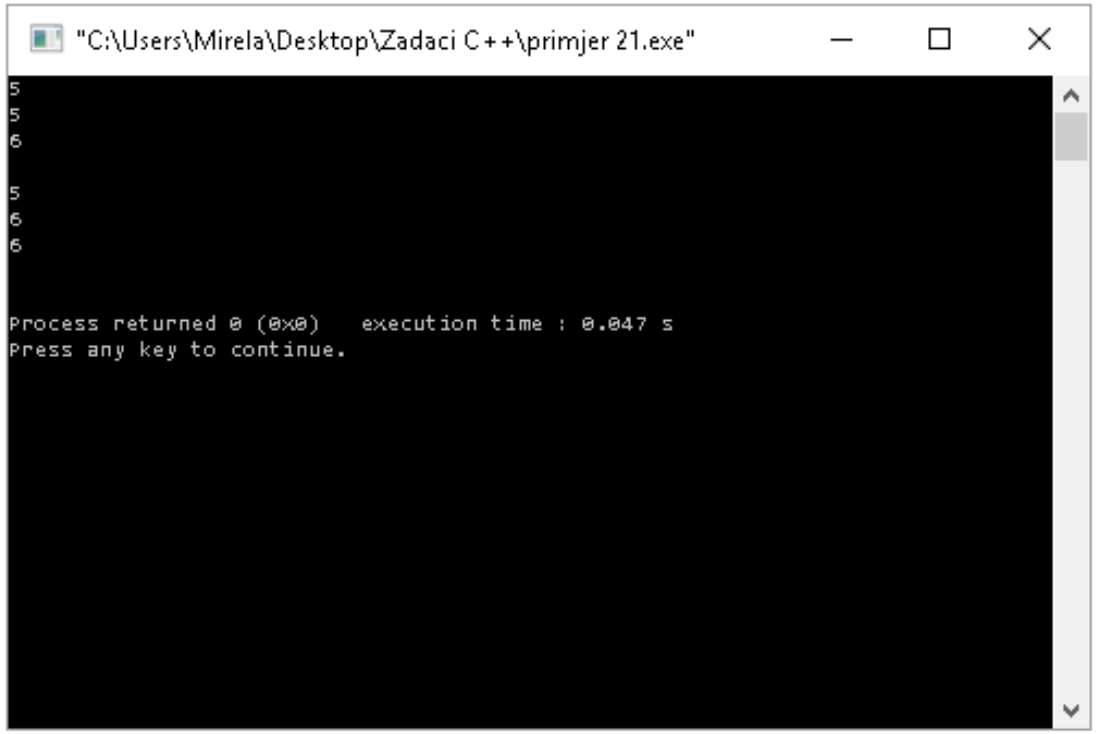
- Operator dekrementa – smanjuje vrijednost varijable za jedan.
  - Izraz x--;  ekvivalentan je izrazu  x=x-1;
- **Prefiks i sufiks forma**
- Operatore inkrementa/dekrementa moguće je pisati ispred i iza varijable:

  **++x;  x++;  –x;  x–;**

- U prefiks notaciji (++x, --x) varijabla će biti promijenjena prije no što će njena vrijednost biti iskorištena u složenom izrazu.
- U sufiks notaciji (x++,  x--) varijabla će biti promijenjena nakon što će njena vrijednost biti iskorištena u složenom izrazu.

Start here | *primjer 21.cpp | primjer 20.cpp

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

int main()
{

int c;
c=5;

cout<< c << endl;
cout<< c++ << endl;
cout<< c << endl<<endl;

c=5;


cout<< c << endl;
cout<< ++c << endl;
cout<< c << endl<<endl;

return 0;
}
```




Logs & others

- **Operatori pridruživanja**
- Operator pridruživanja nazivamo „jednako“ a njegov simbol je „=„. Uz ovaj operator postoje i izvedenice koje omogućuju manje pisanja koda. Složeni operatori biće predstavljeni u tabeli.
- Naredba pridruživanja je oblika:

**varijabla=izraz;**

```
int i=2;

float a=3.17;

char c= 'm';
```

Osim osnovnog operatora pridruživanja postoje i složeni operatori pridruživanja:

**+= -= *= /= %=**

# PRIMJERI SLOŽENIH OPERATORA PRIDRUŽIVANJA

| Izraz | Ekvivalentan izraz |
|---|---|
| i +=5 | i=i+5 |
| i -=j | i=i-j |
| i *=j+1 | i=i*(j+1) |
| i /=4 | i=i/4 |
| i %=2 | i=i%2 |

# RELACIJSKI (POREDBENI) OPERATORI

- Osim aritmetičkih operacija, jezik C++ omogućava i usporedbu dva broja. Kao rezultat usporedbe dobiva se tip podatka bool: ako je uvijet usporedbe zadovoljen, rezultat je true (istina), a ako nije rezultat je false (laž).

| Operator | Značenje |
|---|---|
| < | Manje od |
| <= | Manje ili jednako od |
| > | Veće od |
| >= | Veće ili jednako od |
| == | Jednako |
| != | Različito |

# LOGIČKI OPERATORI

- Složeniji logički izrazi formiraju se pomoću logičkih operatora. Za logičke podatke definirana su tri operatora:

| Operator | Značenje |
|---|---|
| && | AND (I) |
| I I | OR (ILI) |
| ! | NOT (NE) |

- Logička negacija (NOT) je unarni operator koji mijenja logičku vrijednost varijable: istinu pretvara u laž i obrnuto. Logičko I (AND) daje kao rezultat istinu samo ako su oba operanda istinita, a logičko ILI (OR) daje istinu ako je bilo koji od operanada istinit.

| Prvi operand a | Drugi operand b | Rezultat a && b | Rezultat a I I b |
|---|---|---|---|
| True | True | True | True |
| True | False | False | True |
| False | True | False | True |
| False | False | False | False |

# KOMENTARI

- U datoteci izvornog koda korisno je opisati šta se kojim od dijelova koda želi postići, šta su argumenti, objasniti deklaraciju varijabli i sl. Takvi se pomoćni opisi nazivaju komentari. Komentari pomažu programerima kasnije da razumiju određeni dio koda koji možda žele mijenjati. Komentar teksta počinje sa ***//KOMENTAR***, a završava se krajem reda. Može biti napisan u istom redu s naredbom ili u zasebnom redu.
- Jezik C++ podržava i komentare unutar para znakova ***/* KOMENTAR */***. Takvi komentari započinju sa /*, a završavaju */. Kraj reda ne znači podrazumjevani završetak komentara, pa se ovakvi komentarimogu protezati na nekoliko redova izvornog koda, a da se pritom znak za komentar ne mora ponavljati u svakom redu.
- UBACITI SLIKU 14.....

# FUNKCIJA MAIN

- Ovo je osnovna funkcija svakog C++ programa koja označava dio programa koji se treba izvršiti. Ona sadrži blok naredbi – jednu ili više naredbi koje su omeđene parom otvorena – zatvorena vitičasta zagrada{ }.
- Naredbe određuju radnje koje je potrebno izvršiti da bi se ulazni podaci transformisali u izlazne, i na taj način riješio problem za koji je program napisan.
- C++ raspolaže određenom skupinom naredbi koje ćemo u daljem predavanju upoznati.
- Svaka naredba bloka završava sa tačkom-zarez ;.

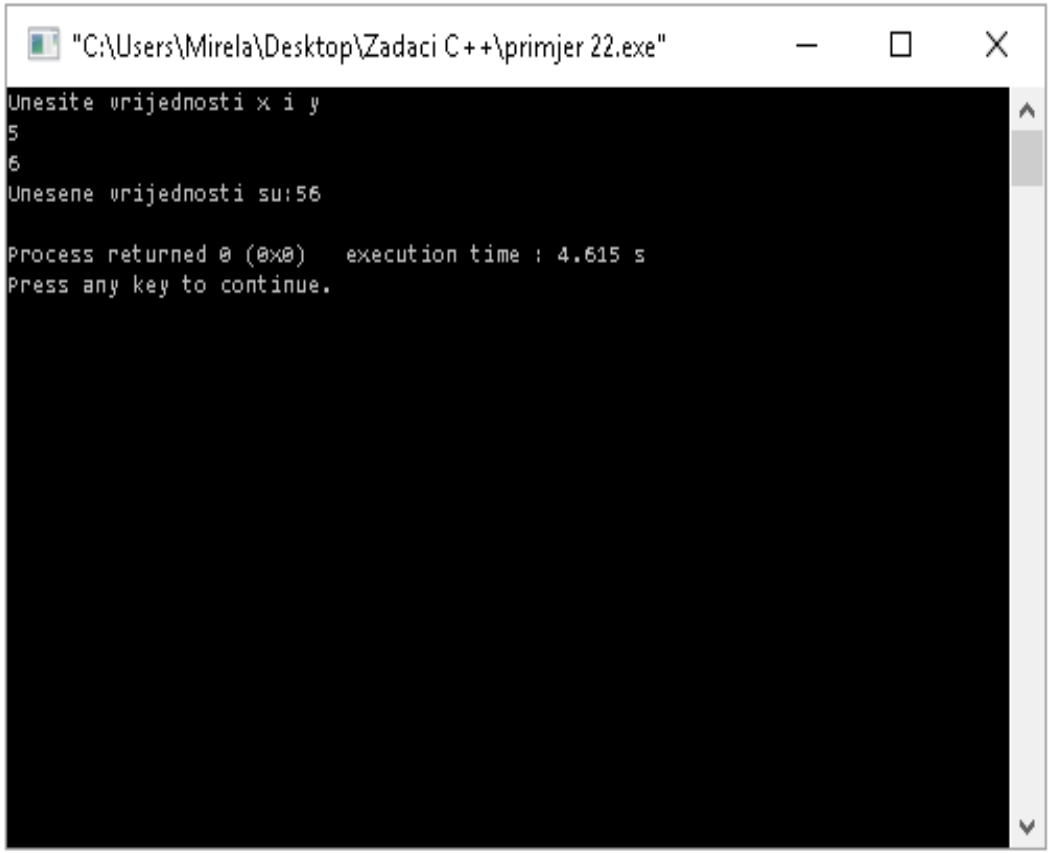
# NAREDBA *CIN* I *COUT*

- ***cout*** – naredba za prikazivanje informacije na ekranu. Ova naredba šalje vrijednost na ekran.
- ***cin*** – naredba za učitavanje podataka. Ova naredba prihvata vrijednosti koje korisnici ukucavaju pomoću tastature.

```
cout<< „Moj prvi program“;
cin>>ime;
```

Start here | primjer 22.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6
7  int x,y;
8
9
10 cout<< "Unesite vrijednosti x i y"<< endl;
11 cin>> x;
12 cin>> y;
13
14 cout<< "Unesene vrijednosti su:"<< x<<y<<endl;
15
16 return 0;
17 }
18
```



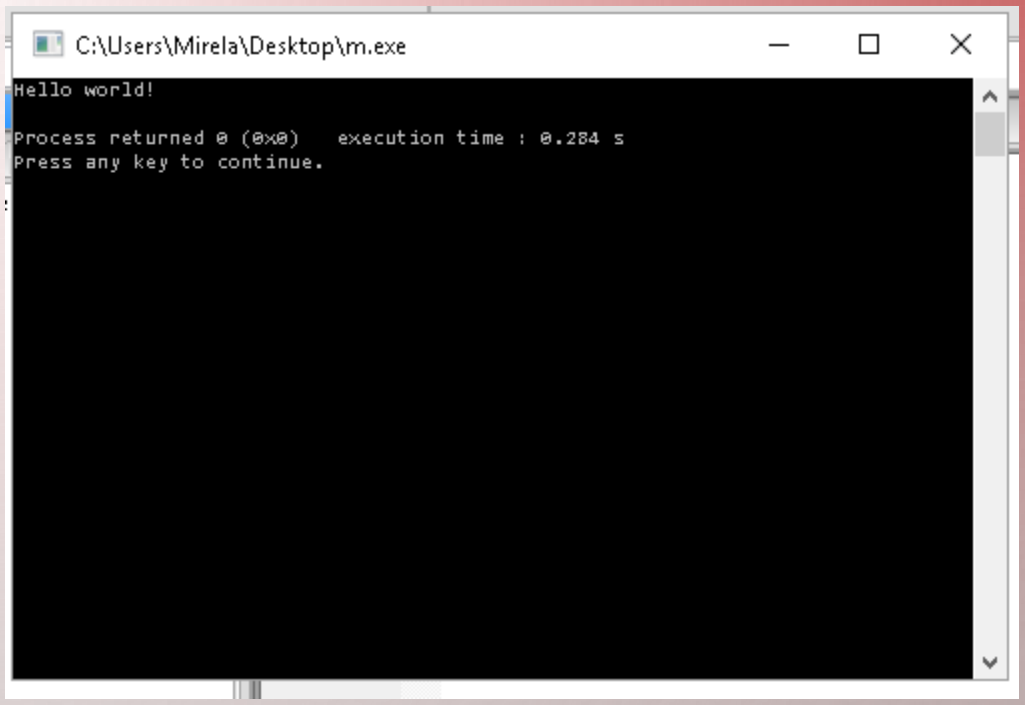
# UVODNI ZADACI - OSNOVE (TIPOVI PODATAKA, UNOS I ISPIS NA KONZOLI)

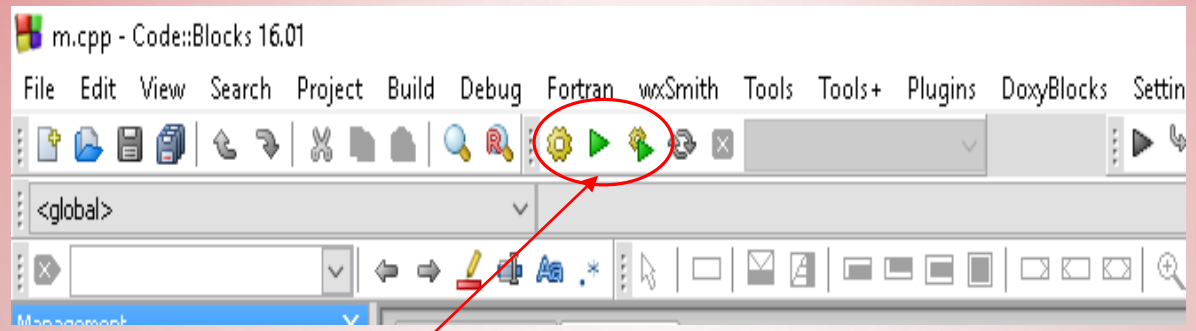
- Primjer 1

„Hello world“ program, je računarski program koji ispisuje „Hello World“ na konzoli.

Start here | m.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       cout << "Hello world!" << endl;
8       return 0;
9   }
10
```






Pritiskom na tipku F5 ili na zeleno dugme koji je prikazan na slici pokreće se postupak ispravnosti napisanog koda.

Proces provjere ispravnosti koda (Debugging) je proces pronalaženja i smanjenja broja grešaka ili nedostataka u programu.

Pokretanje postupak provjere koda, aplikacija će se izvršiti i zatvoriti. To se rješava na dva načina. Prvi način je da dodavanje još jedne linije koda koja glasi:system(„pause“); prije linije return 0; , a poslije cout<<„Hello World“;. Drugi način je da umjesto pritiska na tipku F5 pritisnemo kombinaciju „Ctrl+F5“. Na ovaj način dobijamo aplikaciju nazad a nismo pokrenuli proces provjere koda.

- Primjer 2 - Potrebno je izračunati zbir, razliku, proizvod i količnik dva realna broja. Na početku treba deklarisati dvije realne varijable. Pomoću naredbe ***cin*** korisniku omogućiti unos brojeva, a koristeći naredbu ***cout*** ispisati rezultat operacije.

Start here | primjer 2.cpp

```
1   #include <iostream>
2
3
4   using namespace std;
5
6   int main()
7
8   {
9       float a,b;
10      cout << "Unesi prvi broj:" << endl;
11      cin>> a;
12      cout << "Unesi drugi broj:" << endl;
13      cin>> b;
14      cout << "Rezultat sabiranja dva broja je: "<< a+b<< endl;
15      cout << "Rezultat oduzimanja dva broja je:" << a-b<< endl;
16      cout << "Rezultat množenja dva broja je:" << a*b<< endl;
17      cout << "Rezultat dijeljenja dva broja je:" << a/b<< endl;
18
19      return 0;
20  }
```

"C:\Users\Mirela\Desktop\primjer 2.exe"

```
Unesi prvi broj:
9
Unesi drugi broj:
6
Rezultat sabiranja dva broja je: 15
Rezultat oduzimanja dva broja je:3
Rezultat mnoRenja dva broja je:54
Rezultat dijeljenja dva broja je:1.5

Process returned 0 (0x0)   execution time : 6.646 s
Press any key to continue.
```

- Primjer 3 – Deklarisati varijable a1,a2,a3,a4,a5 i a6 tipa char. Dodijeliti u svaku varijablu po jedno slovo iz riječi „Mostar“. Zatim ispisati tu riječ na ekran.

Start here | primjer 3.cpp | Untitled1.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6
7   char a1='M';
8   char a2='O';
9   char a3='S';
10  char a4='T';
11  char a5='A';
12  char a6='R';
13  cout <<a1<<a2<<a3<<a4<<a5<<a6<<endl;
14
15  }
16
```

"C:\Users\Mirela\Desktop\primjer 3.exe"

```
MOSTAR

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

- PRIMJER 4– Potrebno je uključiti biblioteku string, deklarisati varijablu „mojelme“ te inicijalizirati vrijednost te varijable na Vaše ime. Ispisati vrijednost varijable „mojelme“.

Start here | Primjer 4.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7
8  string mojeIme="Mirela";
9  cout<<"Moje ime je "<<mojeIme<<endl;
10 return 0;
11
12 }
13
```

"C:\Users\Mirela\Desktop\Primjer 4.exe"

```
Moje ime je Mirela

Process returned 0 (0x0)   execution time : 0.039 s
Press any key to continue.
```

# ZADATAK:

Napraviti program, slijedeći navedene zahtjeve:

a) Deklarisati varijablu a1 tipa int i dodjeliti joj vrijednost 10,

b) Deklaristai varijablu a2 tipa int i učitati joj vrijednost sa tastature,

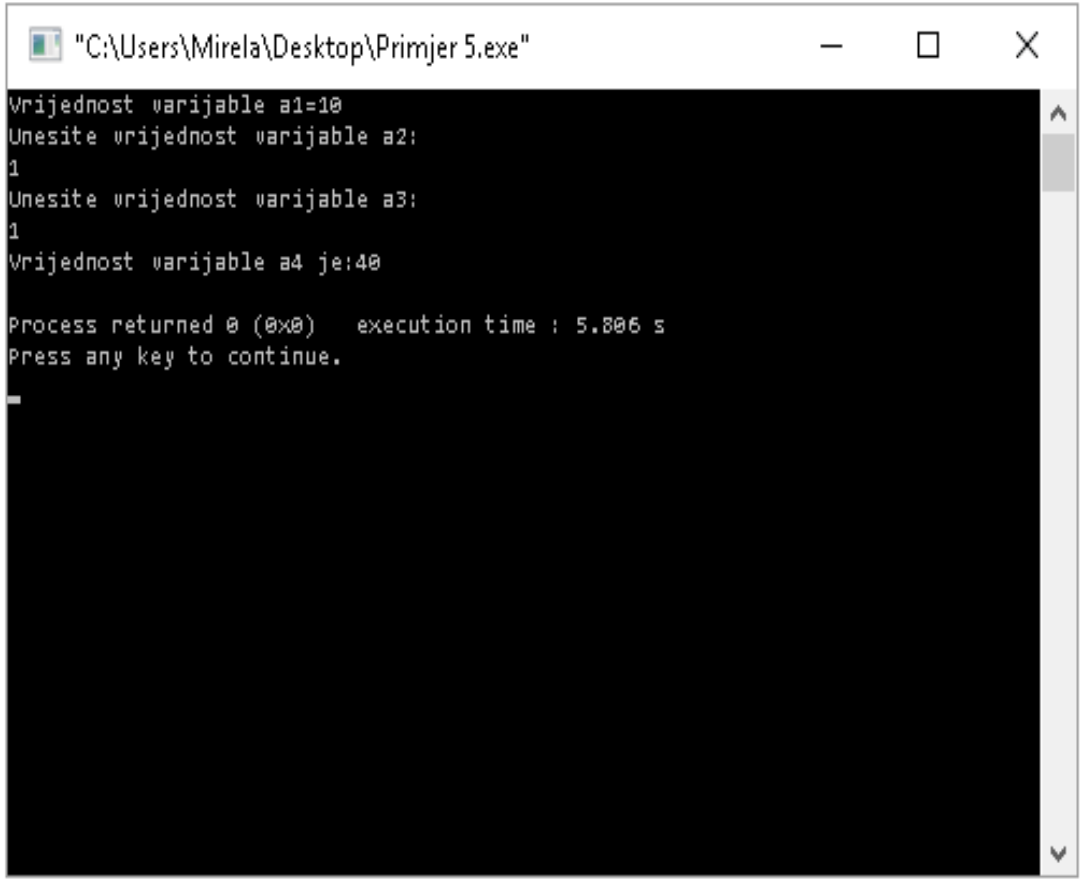
c) Deklarisati varijablu a3 tipa int i učitati joj vrijednost sa tastature,

d) Deklarisati varijablu a4 tipa int i dodijeliti joj vrijednost (a1+a2+a3)*3+4,

e) Ispisati vrijednost varijable a4.

Start here | Primjer 5.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int a1,a2,a3,a4;

a1=10;
cout<<"Vrijednost varijable a1=10"<<endl;

cout<< "Unesite vrijednost varijable a2:"<<endl;
cin>> a2;

cout<< "Unesite vrijednost varijable a3:"<<endl;
cin>> a3;

a4=(a1+a2+a3)*3+4;

cout<<"Vrijednost varijable a4 je:"<<a4<<endl;
return 0;

}
```



# PROGRAMSKA STRUKTURA SELEKCIJA

- Selekcija se u C++ može predstaviti sljedećim iskazima:

- ***if iskazom***
- ***if – else iskazom***
- ***switch iskazom***

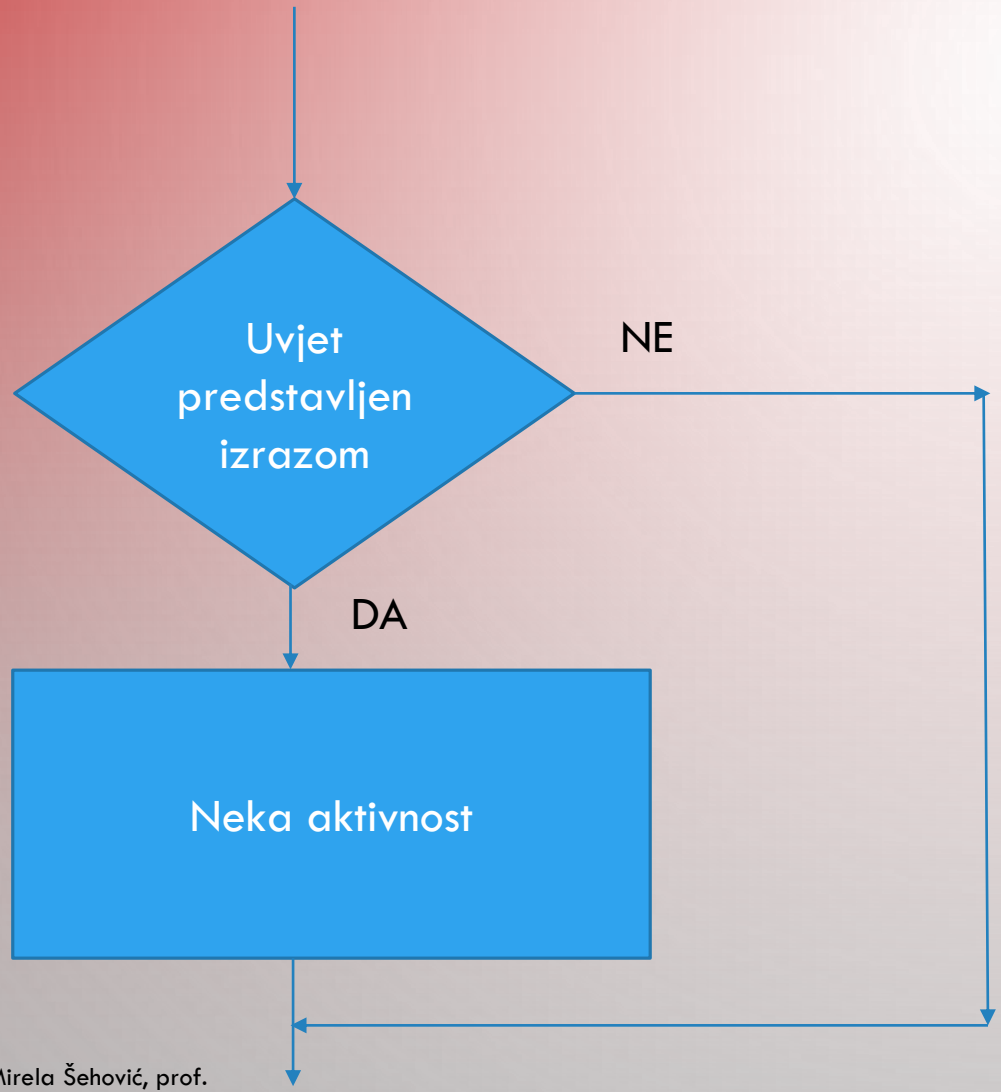
# *if ISKAZ*

- Sintaksu **if** iskaza možemo predstaviti:

```
if (uvjet)
{blok_naredbi;}
```

- gdje **if** ključna riječ se prevodi „ako“;
- uvjet je izraz čija je vrijednost tipa bool (istina/laž);
- blok naredbi je niz naredbi koje će se izvršiti u slučaju da je vrijednost izračunatog izraza istina;

- **if** iskaz koristimo da predstavimo sljedeći tok aktivnosti:




**Primjer: Izvršavanje samo jedne naredbe unutar if iskaza:**

```
if (rezultat_testa <60)
cout<<„Pali ste na testu!“;
```

- Primjer: Izvršavanje više naredbi unutar **if** iskaza, u tom slučaju govorimo o bloku naredbi i koristimo vitičaste zagrade:

```
if (radni_sati >40)
{
prekovremeni=radni_sati-40,
honorar=prekovremeni*satnica;
}
```

- Primjer 6: Napisati program za jednostavnu igru pogađanja brojeva (u intervalu od 1 do 10).

Start here | Primjer 6.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      int pogodak=7;
8      int pokusaj;
9
10 cout<<"Zamislio sam broj izmedju 1 i 10"<<endl;
11 cout<< "Pokusajte pogoditi o kojem je broju rijec"<<endl;
12 cin>> pokusaj;
13
14 if (pokusaj==pogodak)
15
16 cout<<"Nevjeovatno pogodili ste"<<endl;
17 return 0;
18
19 }
20
```

"C:\Users\Mirela\Desktop\Primjer 6.exe"

```
Zamislio sam broj izmedju 1 i 10
Pokusajte pogoditi o kojem je broju rijec
7
Nevjeovatno pogodili ste

Process returned 0 (0x0)   execution time : 3.219 s
Press any key to continue.
```

- Ukoliko se u program unese broj 7 ispisati će se poruka „Nevjerovatno pogodili ste“, ako se unese bilo koji drugi broj koji nije 7, nema poruke i ništa se ne dešava.

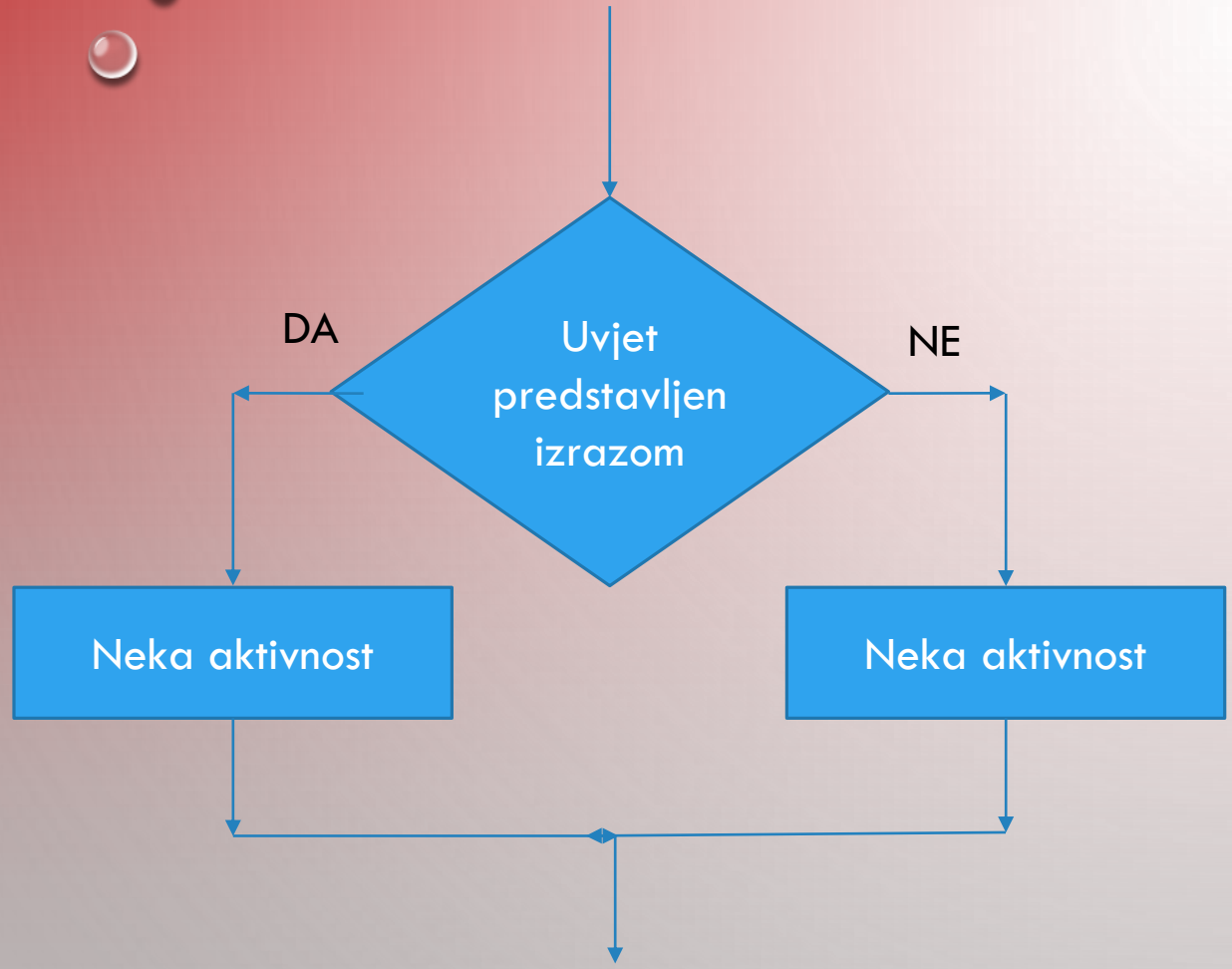
# IF – ELSE ISKAZ

- Sintaksu if – else iskaza možemo predstaviti:

```
if (uvjet)
{blok_naredbi_1;}
else
{blok_naredbi_2;}
```

- gdje if ključna riječ se prevodi „ako“;
- Else je također ključna riječ prevodi se inače;
- uvjet je izraz čija je vrijednost tipa bool (istina/laž);
- blok naredbi_1 je niz naredbi koje će se izvršiti u slučaju da je vrijednost izračunatog izraza istina;
- blok naredbi_2 je niz naredbi koje će se izvršiti u slučaju da je vrijednost izračunatog izraza nije istina;

- If – else  iskaz koristimo da predstavimo sljedeći tok aktivnosti:



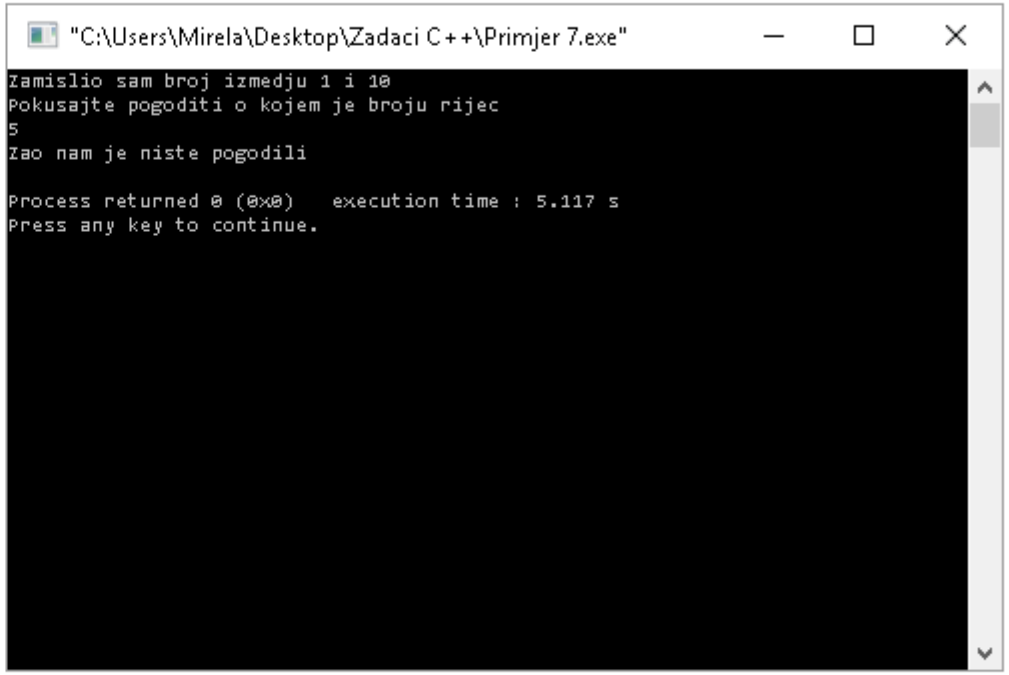

**Primjer: Izvršavanje naredbi unutar if - else iskaza:**

```
if (rezultat_testa <60)
cout<<„Pali ste na testu!“;
else
cout<<„Položili ste test!“;
```

- Primjer 7: Napisati program za jednostavnu igru pogađanja brojeva (u intervalu od 1 do 10).

Start here | *Primjer 6.cpp | **Primjer 7.cpp**

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int pogodak=7;
    int pokusaj;

cout<<"Zamislio sam broj izmedju 1 i 10"<<endl;
cout<< "Pokusajte pogoditi o kojem je broju rijec"<<endl;
cin>> pokusaj;

if (pokusaj==pogodak)

cout<<"Nevjeovatno pogodili ste"<<endl;
else
cout<<"Zao nam je niste pogodili"<<endl;
return 0;

}
```




- Ukoliko se u program unese broj 7 ispisati će se poruka „Nevjerovatno pogodili ste“, ako se unese bilo koji drugi broj koji nije 7, ispisati će se poruka „Zao nam je niste pogodili“..

Start here | *Primjer 8.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int pogodak=7;
    int pokusaj;

cout<<"Zamislio sam broj izmedju 1 i 10"<<endl;
cout<< "Pokusajte pogoditi o kojem je broju rijec"<<endl;
cin>> pokusaj;

if (pokusaj==pogodak)
cout<<"Nevjeovatno pogodili ste"<<endl;
else
{
    if (pokusaj>pogodak)
cout<<"Zao nam je niste pogodili unijeli ste preveliki broj"<<endl;
else
cout<<"Zao nam je niste pogodili unijeli ste premali broj"<<endl;

}
return 0;
}
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 8.exe"

```
Zamislio sam broj izmedju 1 i 10
Pokusajte pogoditi o kojem je broju rijec
5
Zao nam je niste pogodili unijeli ste premali broj

Process returned 0 (0x0)   execution time : 2.683 s
Press any key to continue.
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 8.exe"

```
Zamislio sam broj izmedju 1 i 10
Pokusajte pogoditi o kojem je broju rijec
9
Zao nam je niste pogodili unijeli ste preveliki broj

Process returned 0 (0x0)   execution time : 5.391 s
Press any key to continue.
```

# ZADACI ZA VJEŽBANJE

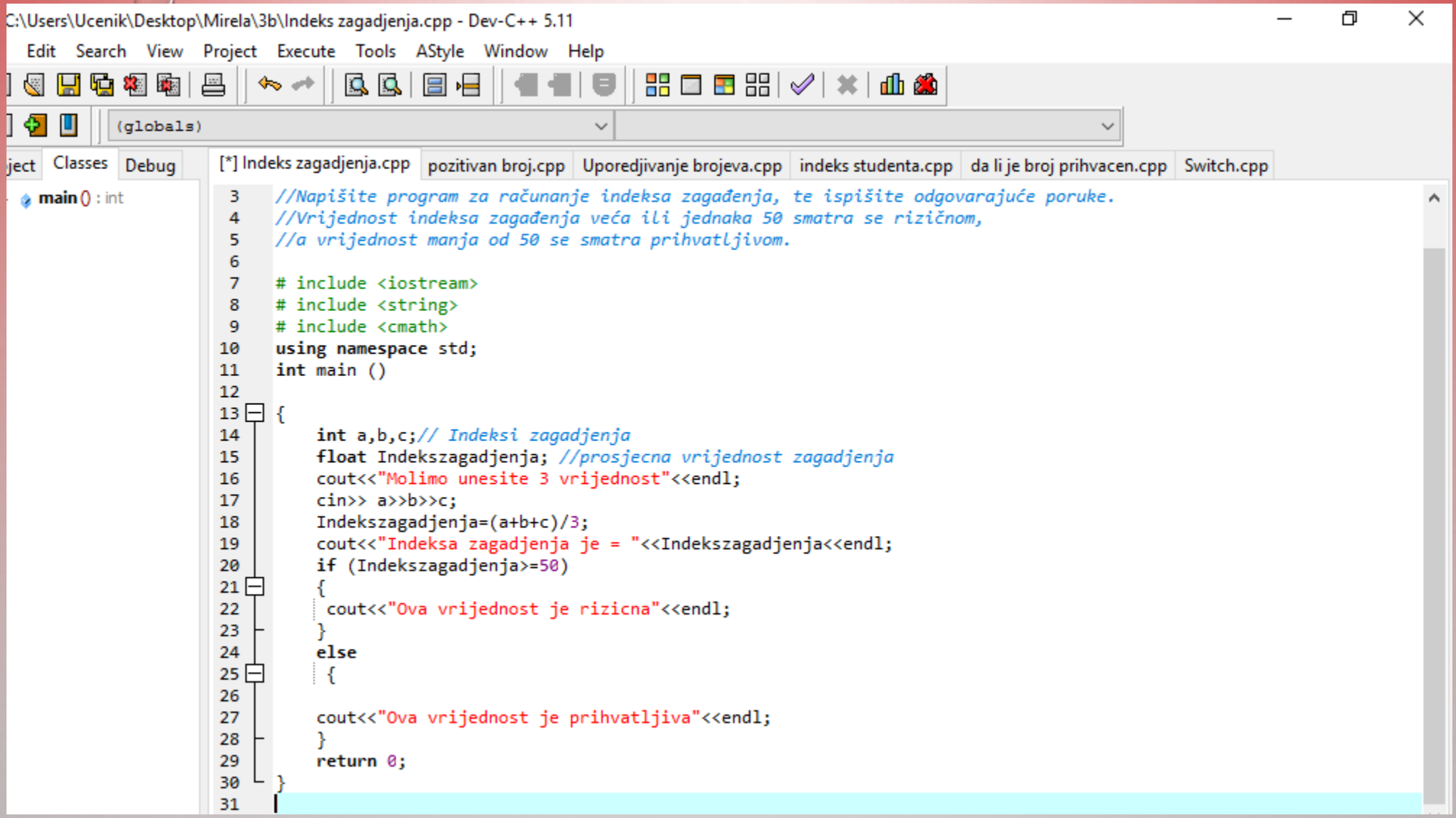
1. Veličina velikih svjetskih gradova vrši mjerenja zagađenog zraka. Očitavanje se vrše u 12 :00 sati na tri različita mjesta u gradu. Prosječna vrijednost ova tri očitanja se naziva indeks zagađenja. Napišite program za računanje indeksa zagađenja, te ispišite odgovarajuće poruke. Vrijednost indeksa zagađenja veća ili jednaka 50 smatra se rizičnom, a vrijednost manja od 50 se smatra prihvatljivom.
2. Napisati program u kojem ćete omogućiti korisniku da unese cijeli broj različit od 0. Nakon toga program treba provjeriti je li uneseni broj negativan ili pozitivan. U oba slučaja ispisati odgovarajuću poruku.
3. Napravite program u kome ćete od korisnika zahtjevati unos dva broja. Za unesene vrijednosti koristite varijable a i b tipa integer. Program treba, ako je prvi broj veći od drugog, ispisati poruku „prvi broj je veći od drugog“, inače treba ispisati poruku“prvi broj nije veći od drugog“.

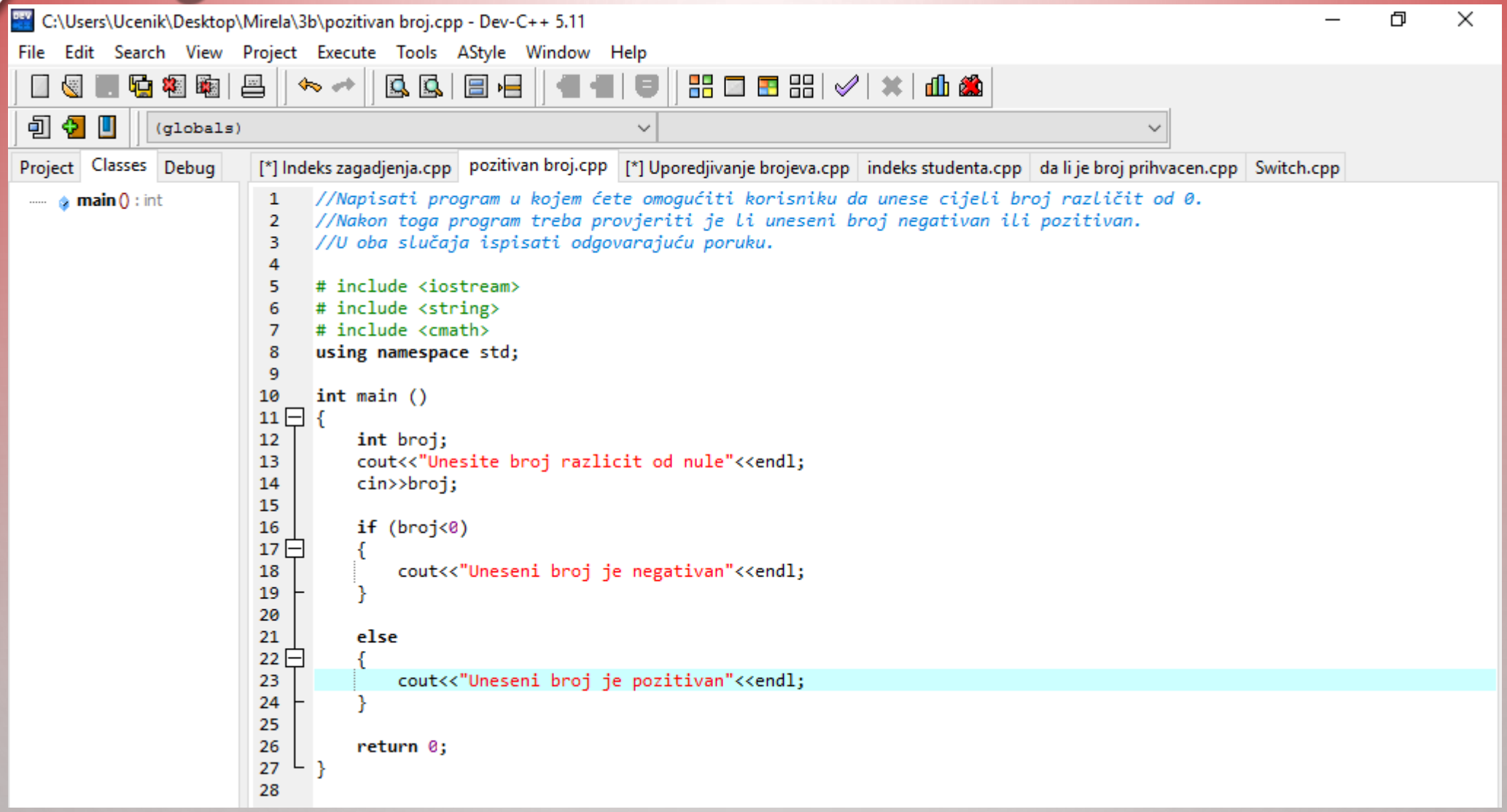
4. Napravite program u kome ćete od korisnika zahtjevati da unese broj indeksa nekog studenta. Program treba ispisati da li je to validan broj indeksa (validni brojevi su brojevi između 1 i 1600, uključujući i njih).

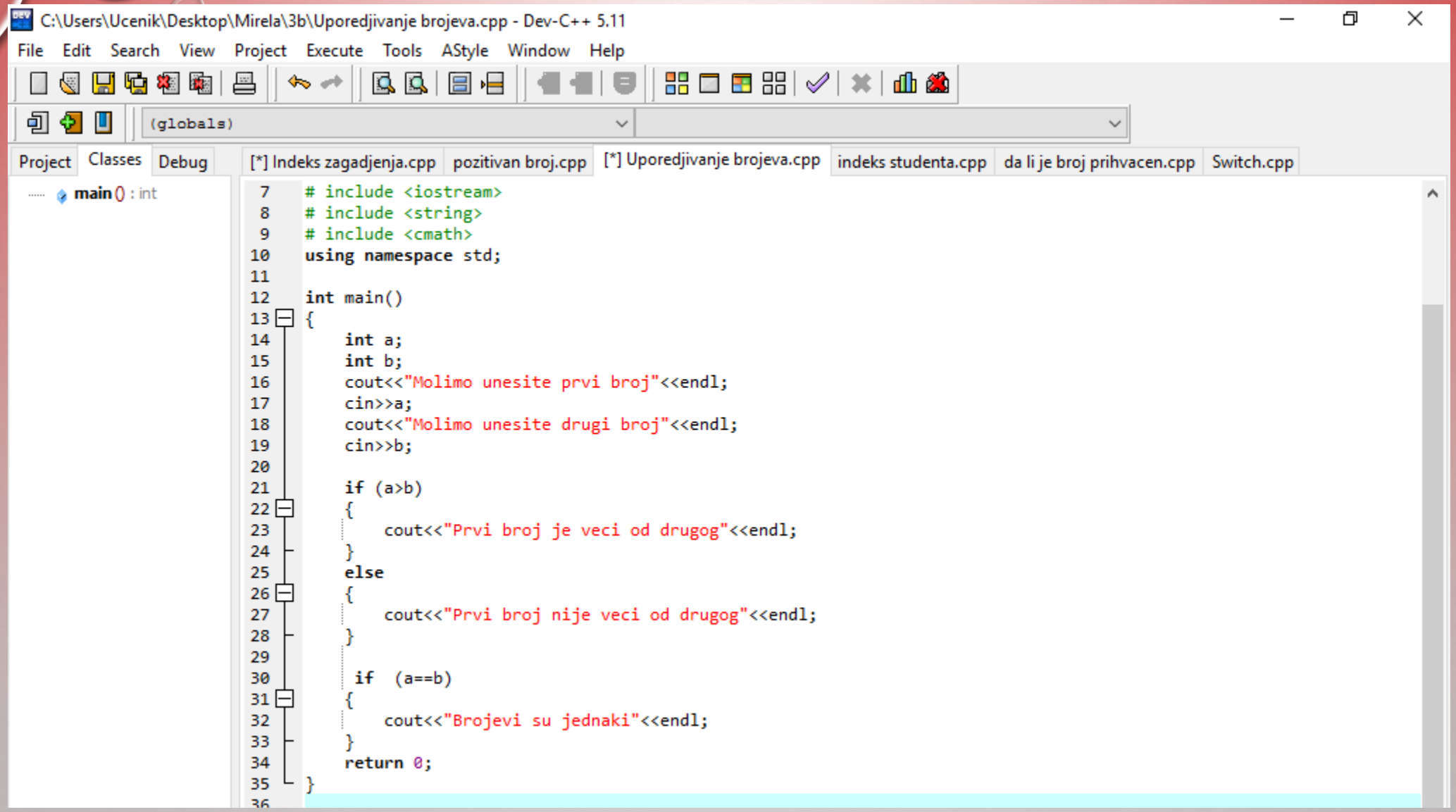
5. Napravite program koji će ispisati da li je uneseni broj prihvaćen. Broj je prihvaćen ako je zadovoljio sljedeće nabrojane uslove:

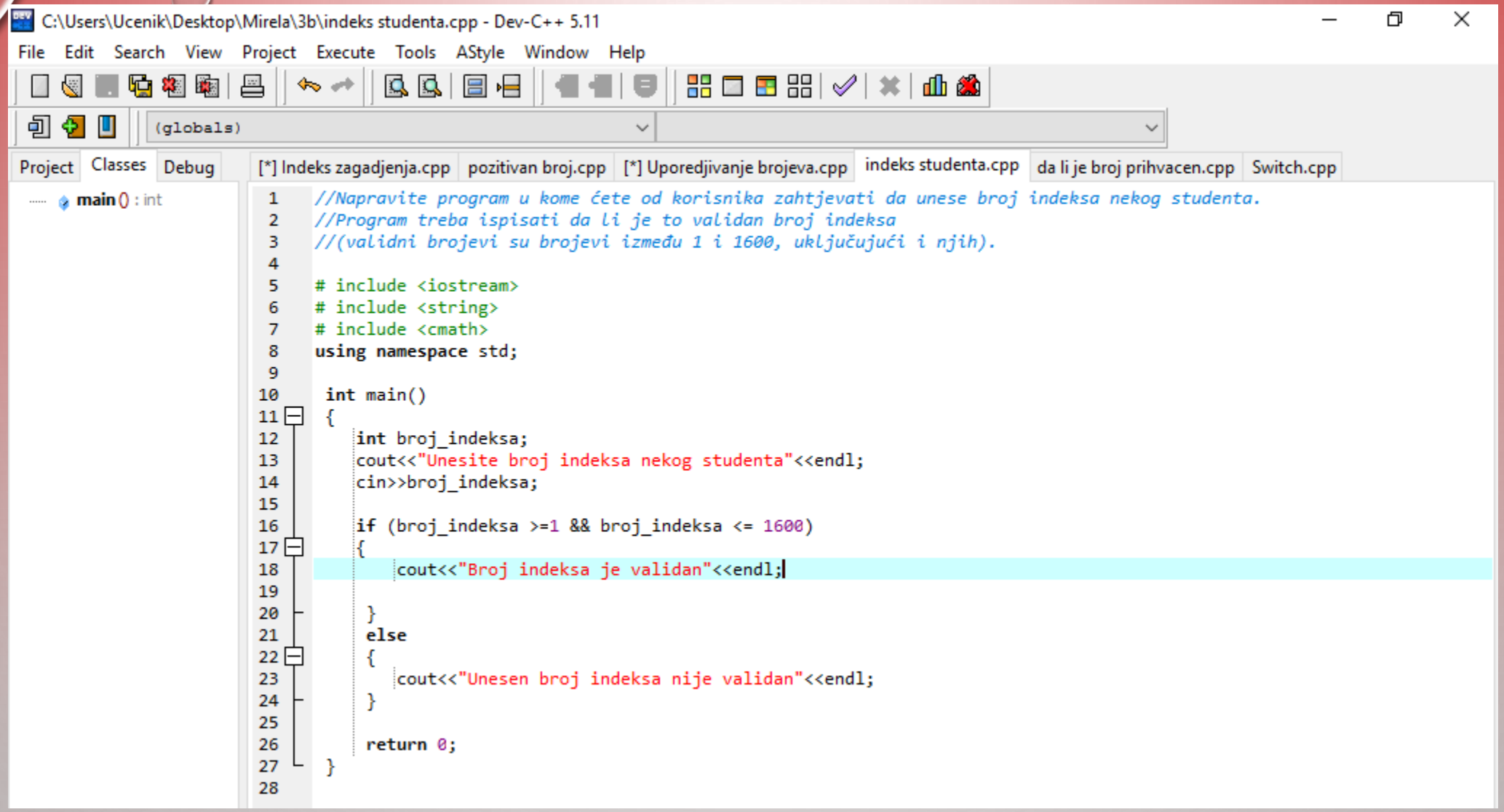
   - pozitivan

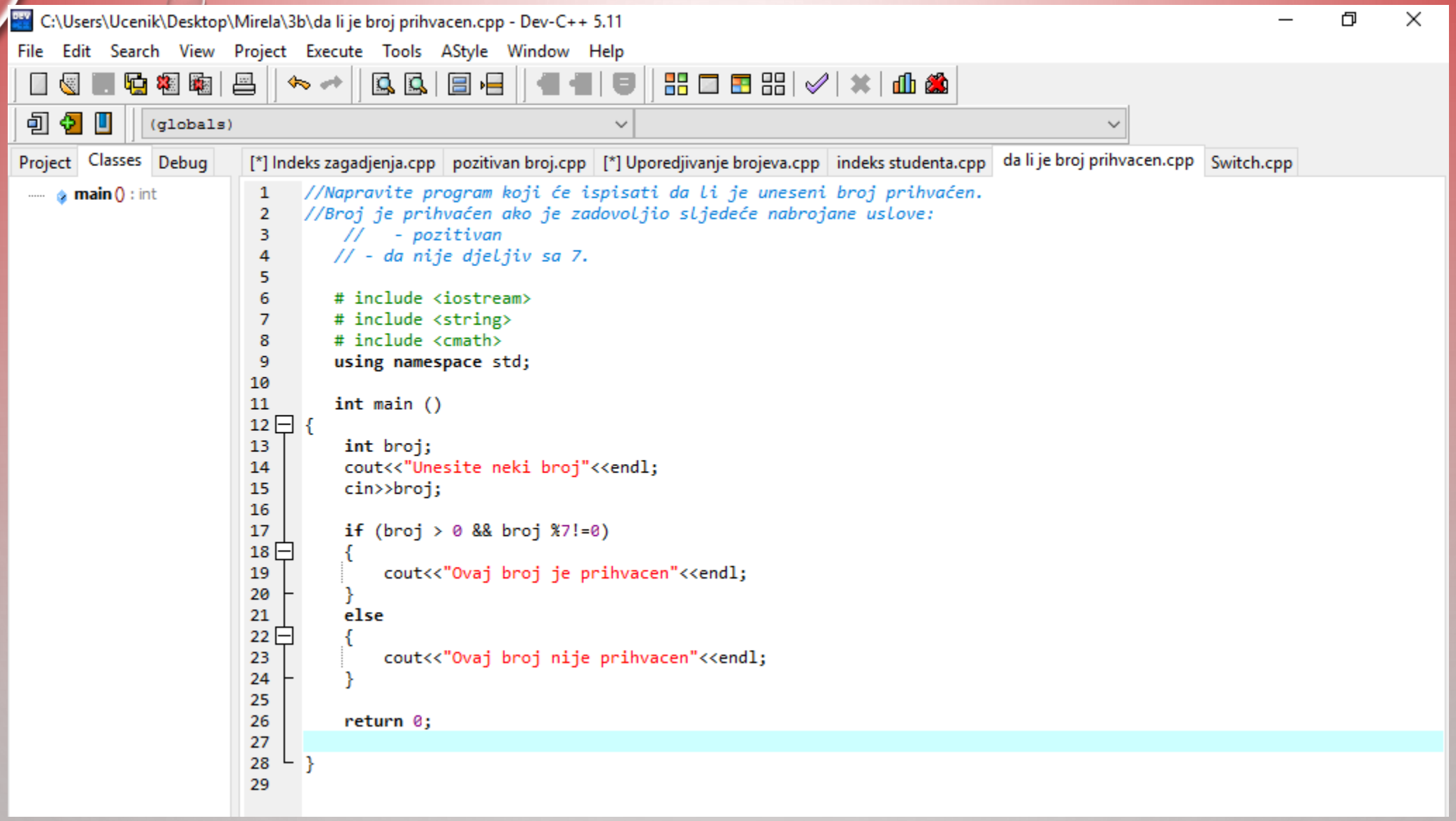
   - da nije djeljiv sa 7.


C:\Users\Ucenik\Desktop\Mirela\3b\Indeks zagadjenja.cpp - Dev-C++ 5.11
Edit Search View Project Execute Tools AStyle Window Help
(globals)
Classes Debug
main () : int
[*] Indeks zagadjenja.cpp
pozitivan broj.cpp
Uporedjivanje brojeva.cpp
indeks studenta.cpp
da li je broj prihvacen.cpp
Switch.cpp
3 //Napišite program za računanje indeksa zagađenja, te ispišite odgovarajuće poruke.
4 //Vrijednost indeksa zagađenja veća ili jednaka 50 smatra se rizičnom,
5 //a vrijednost manja od 50 se smatra prihvatljivom.
6
7 # include <iostream>
8 # include <string>
9 # include <cmath>
10 using namespace std;
11 int main ()
12
13 {
14 int a,b,c;// Indeksi zagadjenja
15 float Indekszagadjenja; //prosjecna vrijednost zagadjenja
16 cout<<"Molimo unesite 3 vrijednost"<<endl;
17 cin>> a>>b>>c;
18 Indekszagadjenja=(a+b+c)/3;
19 cout<<"Indeksa zagadjenja je = "<<Indekszagadjenja<<endl;
20 if (Indekszagadjenja>=50)
21 {
22 cout<<"Ova vrijednost je rizicna"<<endl;
23 }
24 else
25 {
26
27 cout<<"Ova vrijednost je prihvatljiva"<<endl;
28 }
29 return 0;
30 }
31


C:\Users\Ucenik\Desktop\Mirela\3b\pozitivan broj.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
[*] Indeks zagadjenja.cpp
pozitivan broj.cpp
[*] Uporedjivanje brojeva.cpp
indeks studenta.cpp
da li je broj prihvacen.cpp
Switch.cpp
main () : int
1 //Napisati program u kojem ćete omogućiti korisniku da unese cijeli broj različit od 0.
2 //Nakon toga program treba provjeriti je li uneseni broj negativan ili pozitivan.
3 //U oba slučaja ispisati odgovarajuću poruku.
4
5 # include <iostream>
6 # include <string>
7 # include <cmath>
8 using namespace std;
9
10 int main ()
11 {
12 int broj;
13 cout<<"Unesite broj razlicit od nule"<<endl;
14 cin>>broj;
15
16 if (broj<0)
17 {
18 cout<<"Uneseni broj je negativan"<<endl;
19 }
20
21 else
22 {
23 cout<<"Uneseni broj je pozitivan"<<endl;
24 }
25
26 return 0;
27 }
28


C:\Users\Ucenik\Desktop\Mirela\3b\Uporedjivanje brojeva.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
main() : int
[*] Indeks zagadjenja.cpp
pozitivan broj.cpp
[*] Uporedjivanje brojeva.cpp
indeks studenta.cpp
da li je broj prihvacen.cpp
Switch.cpp
7 # include <iostream>
8 # include <string>
9 # include <cmath>
10 using namespace std;
11
12 int main()
13 {
14 int a;
15 int b;
16 cout<<"Molimo unesite prvi broj"<<endl;
17 cin>>a;
18 cout<<"Molimo unesite drugi broj"<<endl;
19 cin>>b;
20
21 if (a>b)
22 {
23 cout<<"Prvi broj je veci od drugog"<<endl;
24 }
25 else
26 {
27 cout<<"Prvi broj nije veci od drugog"<<endl;
28 }
29
30 if (a==b)
31 {
32 cout<<"Brojevi su jednaki"<<endl;
33 }
34 return 0;
35 }


C:\Users\Ucenik\Desktop\Mirela\3b\indeks studenta.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
main () : int
[*] Indeks zagadjenja.cpp
pozitivan broj.cpp
[*] Uporedjivanje brojeva.cpp
indeks studenta.cpp
da li je broj prihvacen.cpp
Switch.cpp
//Napravite program u kome ćete od korisnika zahtjevati da unese broj indeksa nekog studenta.
//Program treba ispisati da li je to validan broj indeksa
//(validni brojevi su brojevi između 1 i 1600, uključujući i njih).
# include <iostream>
# include <string>
# include <cmath>
using namespace std;
int main()
{
int broj_indeksa;
cout<<"Unesite broj indeksa nekog studenta"<<endl;
cin>>broj_indeksa;
if (broj_indeksa >=1 && broj_indeksa <= 1600)
{
cout<<"Broj indeksa je validan"<<endl;
}
else
{
cout<<"Unesen broj indeksa nije validan"<<endl;
}
return 0;
}


C:\Users\Ucenik\Desktop\Mirela\3b\da li je broj prihvacen.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
main() : int
[*] Indeks zagadjenja.cpp
pozitivan broj.cpp
[*] Uporedjivanje brojeva.cpp
indeks studenta.cpp
da li je broj prihvacen.cpp
Switch.cpp
1 //Napravite program koji će ispisati da li je uneseni broj prihvaćen.
2 //Broj je prihvaćen ako je zadovoljio sljedeće nabrojane uslove:
3 // - pozitivan
4 // - da nije djeljiv sa 7.
5
6 # include <iostream>
7 # include <string>
8 # include <cmath>
9 using namespace std;
10
11 int main ()
12 {
13 int broj;
14 cout<<"Unesite neki broj"<<endl;
15 cin>>broj;
16
17 if (broj > 0 && broj %7!=0)
18 {
19 cout<<"Ovaj broj je prihvacen"<<endl;
20 }
21 else
22 {
23 cout<<"Ovaj broj nije prihvacen"<<endl;
24 }
25
26 return 0;
27
28 }
29

# VIŠESTRUKI IZBOR: SWITCH ISKAZ

- Višestruki izbor korištenjem iskaza switch:

```
switch (cjelobrojni_izraz)
{
case konstanta1:prvi_blok_naredbi; break;
case konstanta2:prvi_blok_naredbi; break;
case konstanta3:prvi_blok_naredbi; break;
.
.
.
case konstantan:n-ti blok naredbi;break;
default:m-ti blok naredbi;
}
```

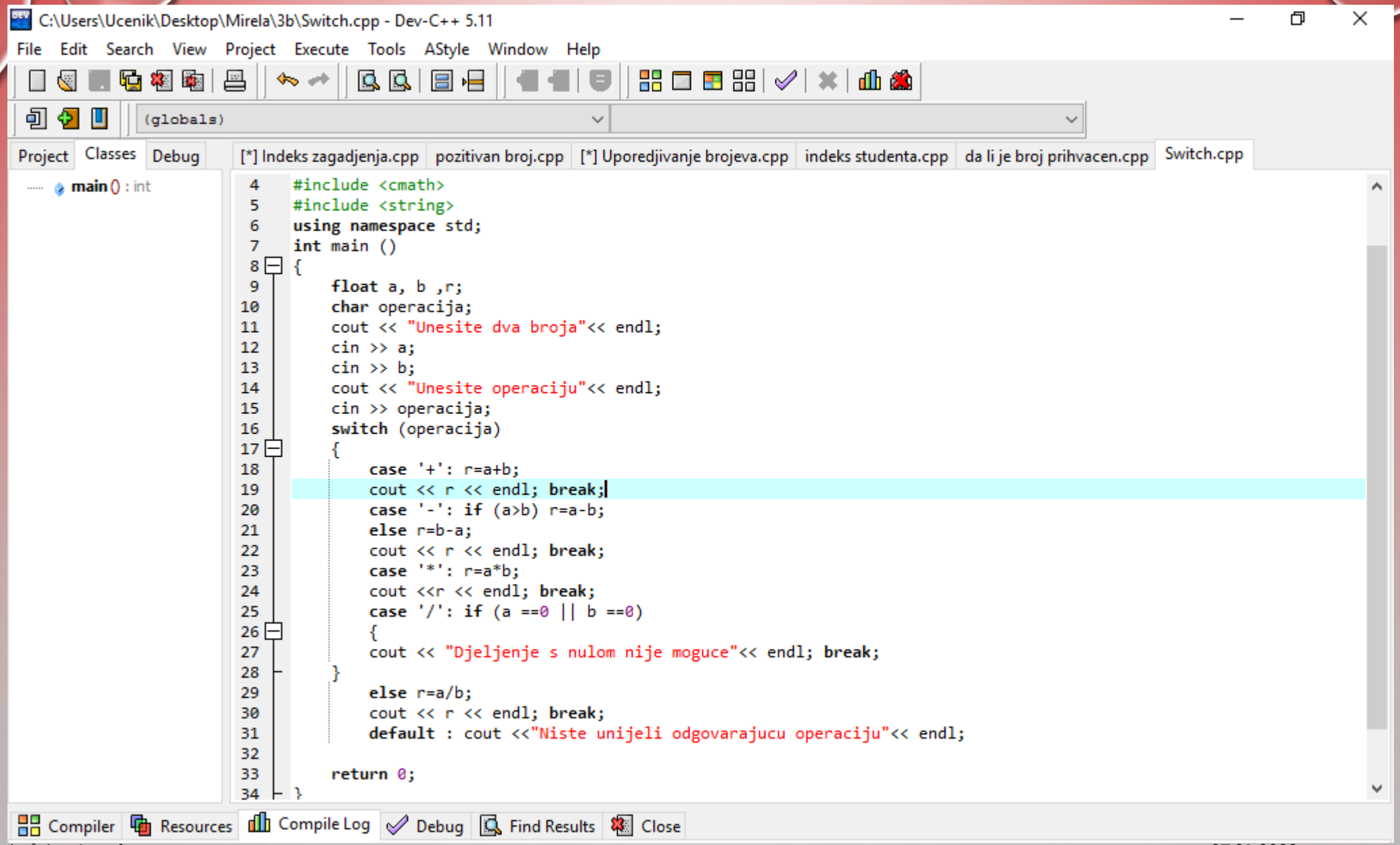
- switch i default su ključne (rezervisane) riječi
- cjelobrojni_izraz je izraz čiji je rezultat cjelobrojna vrijednost (ili vrijednost kompatibilna s cjelobrojnom vrijednošću)
- case konstanta_n je mogući slučaj

- Za izvršavanje switch iskaz najprije se evauira cjelobrojni izraz.
- Ukoliko se izračunata vrijednost nalazi na popisu mogućih slučajeva(case konstanta_n), izvršavanje programa se nastavlja na bloku naredbi iza nevedenog slučaja i prekida se kada dodje do iskaza break.
- Ukoliko se vrijednost cjelobrojnog izraza ne nalazi na popisu mogućih slučajeva izvršavaju se naredbe koje se nalaze iza ključne riječi default.
- Korištenje ključne riječi default je opcionalno (tj.nije nužno da je koristite da bi switch iskaz funkcionisao).
- Ukoliko ne koristite default, a vrijednost cjelobrojnog izraza se ne nalazi na popisu mogućih slučajeva izvršavanje programa se nastavlja na prvoj naredbi switch iskaza.

- Primjer: Napraviti program koji će na osnovu dva unesena broja i znaka za operaciju (+,-,* i /) izvršiti operaciju nad unesenim brojevima.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7     float broj1, broj2, rezultat=0;
8      char operacija;
9
10 cout<<"Unesite dva broja:"<<endl;
11
12 cin>> broj1;
13 cin>> broj2;
14 cout<<"Unesite operaciju:"<<endl;
15 cin>> operacija;
16 switch (operacija)
17
18 {
19     case '+': rezultat=broj1+broj2;cout<<rezultat<<endl;break;
20     case '-': rezultat=broj1-broj2;cout<<rezultat<<endl;break;
21     case '*': rezultat=broj1*broj2;cout<<rezultat<<endl;break;
22     case '/': if (broj2==0){cout<<"Dijeljenje sa 0 nije moguce";break;}else rezultat=broj1/broj2;cout<<rezultat<<endl;break;
23     default:cout<<"Niste unijeli odgovarajucu operaciju"<<endl;
24
25
26 }
27 return 0;
28 }
```

```
"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 9.exe"
Unesite dva broja:
1
1
Unesite operaciju:
+
2

Process returned 0 (0x0)   execution time : 4.845 s
Press any key to continue.
```


C:\Users\Ucenik\Desktop\Mirela\3b\Switch.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
main () : int
[*] Indeks zagadjenja.cpp
pozitivan broj.cpp
[*] Uporedjivanje brojeva.cpp
indeks studenta.cpp
da li je broj prihvacen.cpp
Switch.cpp
4 #include <cmath>
5 #include <string>
6 using namespace std;
7 int main ()
8 {
9 float a, b ,r;
10 char operacija;
11 cout << "Unesite dva broja"<< endl;
12 cin >> a;
13 cin >> b;
14 cout << "Unesite operaciju"<< endl;
15 cin >> operacija;
16 switch (operacija)
17 {
18 case '+': r=a+b;
19 cout << r << endl; break;
20 case '-': if (a>b) r=a-b;
21 else r=b-a;
22 cout << r << endl; break;
23 case '*': r=a*b;
24 cout <<r << endl; break;
25 case '/': if (a ==0 || b ==0)
26 {
27 cout << "Djeljenje s nulom nije moguce"<< endl; break;
28 }
29 else r=a/b;
30 cout << r << endl; break;
31 default : cout <<"Niste unijeli odgovarajucu operaciju"<< endl;
32
33 return 0;
34 }
Compiler Resources Compile Log Debug Find Results Close

# ZADATAK ZA SAMOSTALAN RAD

- Napišite program koji će za slovo uneseno s tastature odrediti je li riječ o samoglasniku ili suglasniku te ispisati odgovarajuću poruku.
- NAPOMENA:C++ je jezik koji razlikuje velika i mala slova!!!

Start here | *Primjer 10.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      char slovo;
8
9  cout<<"Unesite slovo:"<<endl;
10 cin>> slovo;
11
12 switch(slovo)
13     {
14         case 'a': cout<<"samoglasnik"<<endl; cout<<endl; break;
15         case 'e': cout<<"samoglasnik"<<endl; cout<<endl; break;
16         case 'i': cout<<"samoglasnik"<<endl; cout<<endl; break;
17         case 'o': cout<<"samoglasnik"<<endl; cout<<endl; break;
18         case 'u': cout<<"samoglasnik"<<endl; cout<<endl; break;
19         default:cout<<"Suglasnik"<<endl;
20     }
21 return 0;
22 }
23
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 10.exe"

```
Unesite slovo:
a
samoglasnik


Process returned 0 (0x0)   execution time : 2.489 s
Press any key to continue.
```

# PONAVLJANJE (ITERACIJE) ISKAZ „WHILE“

- U C++ ponavljanje (iteracija) se može predstaviti sa:
- while iskazom (petljom),
- do – while iskazom (petljom),
- for iskazom (petljom).
- Sintaksi while iskaza možemo predstaviti:

```
while (uvjet)
        {
naredba_1;
naredba_n;
        }
```

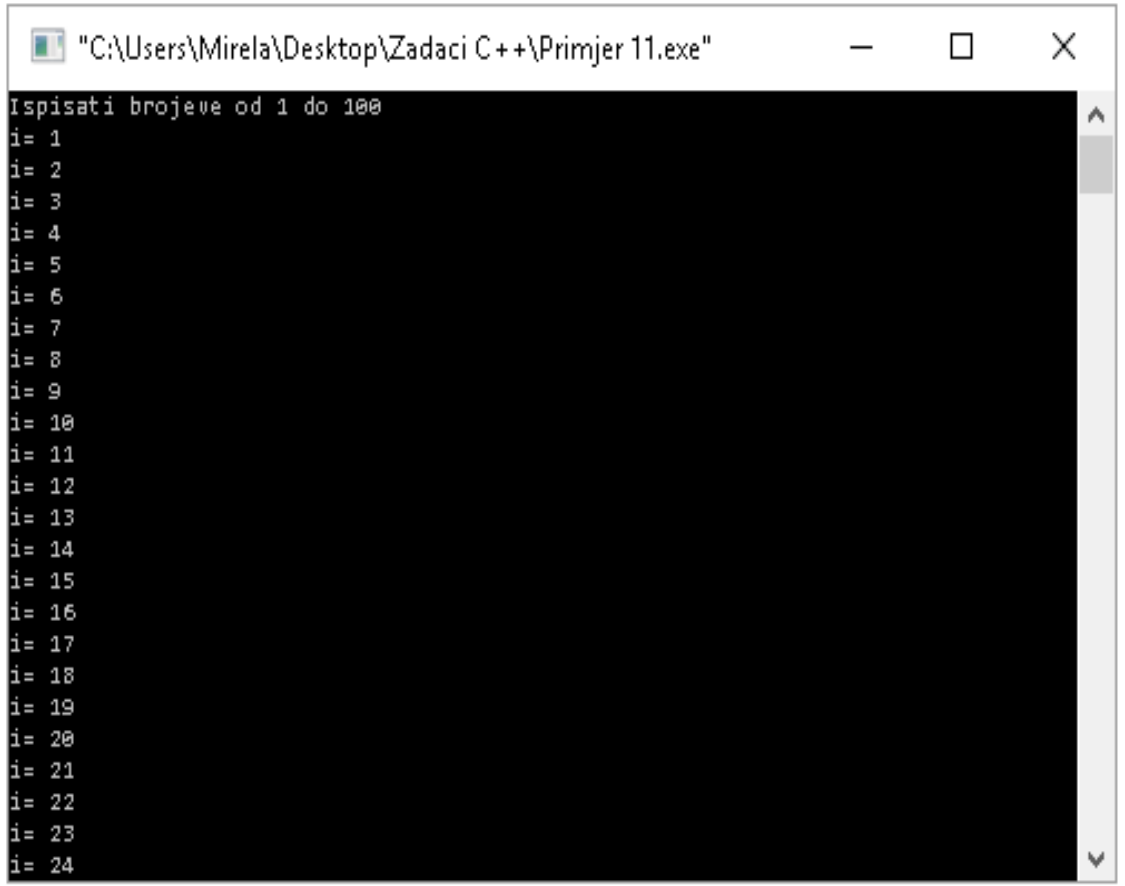
- while ključna (rezervisana) riječ u C++;
- uvijet je izraz čiji je rezultat logička (bool) vrijednost (istina/laž);
- naredbe su skup naredbi koje je potrebno izvršavati sve dok je uvjet istini;

- Smisao naredbe while je u sljedećem:
- Uvjet (izraz) u zagradi se izračunava, i ako je on tačan (preciznije, različit od 0), izvršavaju se naredbe u bloku navedenom unutar vitičastih zagrada neposredno iza naredbe while.
- Nakon što se čitav blok izvrši, uvjet se ponovo provjerava, i ako je i dalje istinit, blok naredbi se ponovo izvršava, itd.
- Sve dok je uvjet tačan blok naredbi se izvršava, kada postane laž program se nastavlja izvršavati poslije while bloka. Ako postoji samo jedna naredba u bloku vitičaste zagrade se ne moraju pisati.

- Primjer 11: Napisati program koji će ispisivati sve brojeve od 1 do 1000.

Start here | Primjer 11.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      int i;
8        cout<<"Ispisati brojeve od 1 do 100"<<endl;
9        i=1;
10                 while (i <= 100)  {
11                    cout<<"i= "<<i<<endl;
12                    i=i+1;
13                 }
14
15 return 0;
16 }
17
```



Start here | Primjer 11.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int i;
      cout<<"Ispisati brojeve od 1 do 100"<<endl;
      i=1;
                while (i <= 100)  {
                   cout<<"i= "<<i<<endl;
                   i++;
                 }

return 0;
}
```

- Primjer 12: Napišite program koji će omogućiti unos pozitivnih cijelih brojeva, te izračunati njihovu sumu. U trenutku kad korisnik unese negativan broj, program ispisuje izračunatu sumu (ne računajući

Start here | Primjer 12.cpp

```
2   #include <string>
3   using namespace std;
4
5   int main()
6   {
7       int suma=0, broj=0;
8         cout<<"Unesite brojeve koji ce se sabirati:"<<endl;
9         cout<<"Negativan broj ce znaciti prekid:"<<endl;
10
11        while (broj>=0)
12
13  {
14      suma+=broj;
15      cout<<"Unesite broj"<<endl;
16      cin>> broj;
17
18  }
19
20                    cout<<"Suma unesenih brojeva je:"<<suma<<endl;
21
22
23  return 0;
24  }
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 12.exe"

```
Unesite brojeve koji ce se sabirati:
Negativan broj ce znaciti prekid:
Unesite broj
2
Unesite broj
3
Unesite broj
3
Unesite broj
3
Unesite broj
3
Unesite broj
3
Unesite broj
3
Unesite broj
-5
Suma unesenih brojeva je:17

Process returned 0 (0x0)   execution time : 14.457 s
Press any key to continue.
```

gs & others

- Greška prilikom korištenja while petlje je najčešća kada se iza uvjeta stavi ;.+ 8tačka-zarez). Kompajler neće prijaviti grešku, radit će neispravno jer će smatrati da ne postoje nikakve naredbe unutar petlje i onda će izvršavati naredbe izvan nje.

- Formalno gledajući ovo nisu sintaksičke (jezičke) greške: petlje bez tijela nisu zabranjene u jeziku C++.

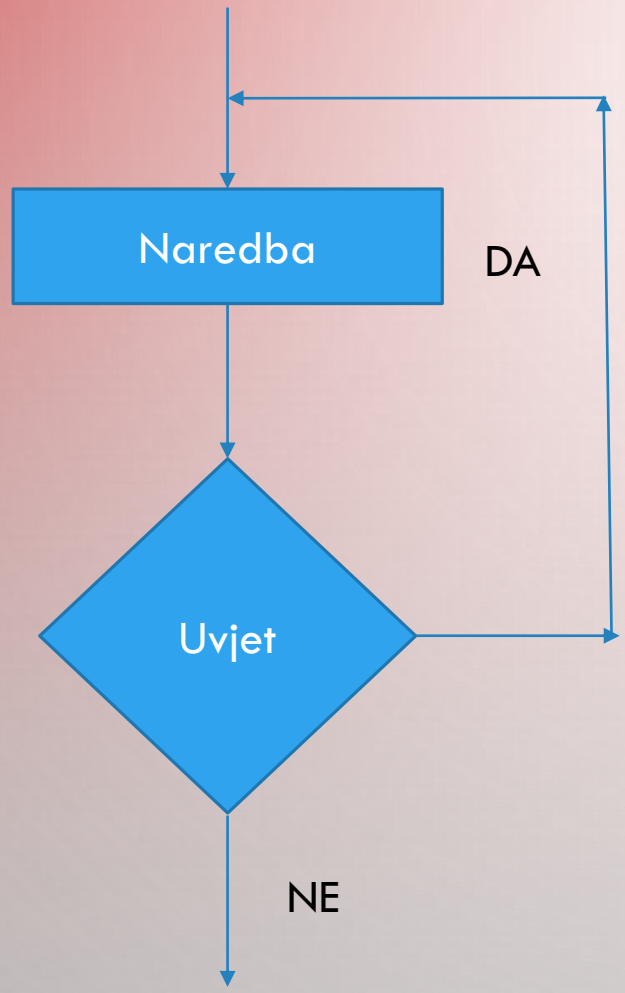
# ISKAZ „DO-WHILE“

- U do – while petlji izvršenje programa kreće od dijela u kojem se opisuje koji kod se treba izvesti tek nakon toga do uvjeta.
- Upravo je to svojstvo do-while petlje i to je jedina petlja za koju kažemo da će se sigurno izvršiti barem jednom.
- Sinataksu do-while iskaza možemo predstaviti:

```
do
 {
naredba_1;
naredba_2;
naredba_3;
 }
while (uvjet);
```

- do i while rezervisane (ključne riječi;
- uvjet je izraz čiji je rezultat bool vrijednost (istina/laž);
- naredbe su sku naredbi koje treba izvršavati sve dok je uvjet istinit;

- Na dijagramu toka do – while iskaz se predstavlja na sljedeći način:




Kad izvršavanja programa dosegne do – while petlju izvršavaju se sve naredbe unutar tijela petlje, nakon toga se provjerava istinitos uvjeta. Ako je uvjet istinit naredbe se izvršavaju unutar tijela sve dok uvijet ne postanene laž.

- Primjer: Napisati program koji korisniku omogućava unos pozitivnih cijelih brojeva (većih od nule). Program se treba izvršavati sve dok korisnik ne unese broj 20. Na kraju program treba ispisati sumu svih parnih unesenih brojeva.

Start here | Primjer 13.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int suma=0;
7      int broj=0;
8
9      do
10 {
11         if ((broj%2==0) && (broj> 0))
12             suma+=broj;
13
14  cout<<"Unesite broj:"<<endl;
15         cin>>broj;
16      }
17     while(broj!=20);
18 cout<<"Suma parnih brojeva je:"<<suma<<endl;
19
20 return 0;
21 }
22
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 13.exe"

```
Unesite broj:
5
Unesite broj:
2
Unesite broj:
6
Unesite broj:
20
Suma parnih brojeva je:8

Process returned 0 (0x0)   execution time : 10.487 s
Press any key to continue.
```

# ZADACI ZA VJEŽBU

1. Napraviti aplikaciju u koju krisnik upisuje brojeve sve dok ne unese malo slovo „e“ ili veliko slovo „E“. Sve brojeve koje korisnik unese treba sabrati i ispisati konačni rezultat.
2. Napraviti aplikaciju koja ispisuje sve cijele brojeve od 1 do 999 i ispisuje koliko je prostih brojeva utom rasponu.
3. Napisati program koji izračunava zbir parnih brojeva od 1 do 1000 koji su sjeljivi sa 7 i 3 ili koji su djeljivi sa 8. Program treba, osim rezultata, da ispisuje i brojeve koji zadovoljavaju uslov.
4. Napisati program koji učitava n parova brojeva, a zatim za svaki učitani par ispisuje manji. Također, treba odrediti koja je srednja vrijednost učitanih brojeva.
5. Napraviti aplikaciju u koju korisnik upisuje cijele brojeve sve dok ne unese broj 0 (nula). Aplikacija treba ispisati koliko je uneseno parnih, a koliko neparnih brojeva, zbir parnih i zbir neparnih brojeva te ukupan zbir brojeva.

# FOR PETLJA

- Nječešće korišteni algoritmi za ponavaljanje su oni gdje želimo ponoviti neku akciju zadani broj puta. Takvi algoritmi se u C++ predstavljaju for petljom. Isto kao i kod while petlje, i kod for petlje uvjet izvršavanja se provjerava na početku tijela petlje.
- Sintaksa for iskaza se može predstaviti:

```
for (inicijalizacija brojača; uvjet ponavljanja; izmjena vrijednosti brojača);
                                    {
                                naredbe;
                                    }
```

- for rezervisana (ključna) riječ, inicijalizacija brojača, uvjet ponavljanja i izmjena vrijednsti brojača su izrazi odnosno skup naredbi koje treba izvršavati sve dok je uvjet istinit.

- Kad izvršavanje programa dosegne for petlju događa se sljedeće:

1. Evaluira se izraz kojim predstavljamo inicijalnu vrijednost brojača;
2. Evaluira se izraz kojim je predstavljen uvjet ponavljanja;
3. Ukoliko je uvjet ponavljanja istinit
   a) Izvršavaju se naredbe unutar tijela petlje;
   b) Evaluira se izraz kojim mijenjamo vrijednost brojača;
   c) Vraćamo se na korak 2

**Ukoliko uvjet ponavljanja nije istinit izvršavanje programa se nastavlja na prvoj naredbi iza for petlje.**

- Primjer: Potrebno je ispisati brojeve od 5 do 55. Potrebno je inicijalizirati vrijednosti brojača (int i=5), zatim postaviti uvjet (i<=55), i na kraju izmjena vrijednosti brojača – inkrement (i++). Naredba koja se izvodi konačan broj puta je ispis trenutne vrijednosti varijable i.

Start here | Primjer 14.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout<< "Ispis brojeva od 5 do 55:"<<endl;
7      for (int i=5; i<=55; i++)
8
9
10 {
11 cout<< i<<endl;;
12 }
13
14 return 0;
15 }
16
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 14.exe"

```
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```

# BESKONAČNA PETLJA

- Beskonačna petlja predstavlja jedan od najčešćih problema u petlji. Često je možemo izazvati tako što zaboravimo ažurirati neke vrijednosti u petlji.
- Može se dogoditi da je uvjet petlje uvijek ispunjen. Petlja će se u tom slučaju izvoditi beskonačno puta. Ako se dogodi da se pokrene program u kojem se nalazi beskonačna petlja, možemo ga prekinuti zatvaranjem prozora u kojemse izvršava program. Beskonačna petlja se ponavlja sve dok ne popuni memorijsku aplikaciju, što znači da će se izvršavati dok računar posjeduje dovoljno RAM memorije. Što je u računaru više RAM memorije petlja će se duže ponavljati.

- Primjer : Vrijednost brojača i je 5. Da bi smo ispisali „Beskonačna petlja“, uvjet je da je brojač i<10. U ovom slučaju brojač i je uvijek manji od 10 pa će se petlja ponavljati neograničen broj puta.

Start here | Primjer 15.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6
7      for (int i=5; i<10;)
8
9
10 {
11 cout<< "Beskonacna petlja"<<endl;;
12 }
13
14 return 0;
15 }
16
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 15.exe"

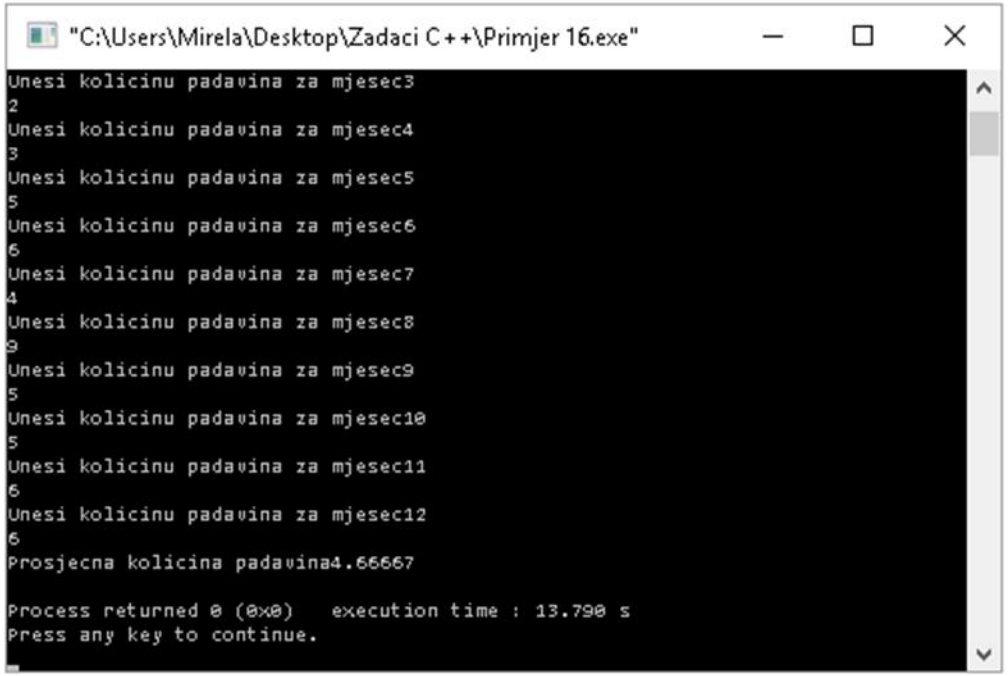
```
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
Beskonacna petlja
B
```

# ZADACI ZA VJEŽBE

1. Napisati program za izračunavanje prosječne godišnje količine padavina. Prosječna godišnja količina padavina se dobije sabiranjem količine padavina za svaki mjesec i dijeljenjem s brojem mjeseci (12). Dakle, ptrebno je omogućiti unos količine padavina 12 puta (za svaki mjesec), te unesene vrijednosti sabrati. Budući da se određena aktivnost (unos količine padavina) treba izvršiti zadani broj puta (12) for petlja je idealan izbor.

Start here | Primjer 15.cpp | **Primjer 16.cpp**

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  double ukupno=0;
7  double padavine=0;
8  double prosjek=0;
9
10     for (int mjesec=1; mjesec<=12;mjesec++)
11 {
12
13 cout<< "Unesi kolicinu padavina za mjesec"<<mjesec<<endl;
14 cin>> padavine;
15 ukupno+=padavine;
16 }
17 prosjek=ukupno/12;
18 cout<<"Prosjecna kolicina padavina";
19 cout<<prosjek<<endl;
20 return 0;
21 }
22
```



2. Napraviti program koji će pitati korisnika koliko želi unijeti brojeva. Zatim program treba zahtjevati unos toliko cijelih brojeva. Nakon što je korisnik unio sve te brojeve, program treba ispisati koliko ima neparnih brojeva i koliko ima negativnih brojeva.

- Pomoć:
  - Od korisnika zahtjevajte unos varijable n da znate koliko ćete brojeva zahtjevati za unos od korisnika
  - Deklarisati ćemo dva dodatna brojača b1 i b2 (ovo nisu brojači for petlje)
  - Brojač 1 će brojati neparne brojeve – njegova početna vrijednost prije for petlje mora biti nula
  - Brojač b2 će brojati sve negativne brojeve (i njega ćemo inicijalizirati na nulu)
  - U for petlju dodajte : unos broja u varijablu x koja je tipa int.

Start here | Primjer 17.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  int n, b1=0,b2=0,x;
7
8  cout <<"Koliko brojeva zelite unijeti?";
9  cin>>n;
10
11
12      for (int i=1; i<=n; i++)
13 {
14
15 cout<< "Unesite"<<i<<" broj"<< endl;
16 cin>> x;
17 if (x%2!=0)
18      b1=b1+1;
19 if (x<0)
20      b2=b2+1;
21 }
22 cout<<"Neparni brojevi:"<< b1<<endl;
23
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 17.exe"

```
Koliko brojeva zelite unijeti?
5
Unesite1 broj
1
Unesite2 broj
2
Unesite3 broj
3
Unesite4 broj
-4
Unesite5 broj
-5
Neparni brojevi:3
Negativni brojevi:2

Process returned 0 (0x0)   execution time : 14.335 s
Press any key to continue.
```

# ZADACI ZA SAMOSTALAN RAD

1. Napraviti petlju koja će ispisati na ekran parne trocifrene brojeve.
2. Napraviti program za računanje sume kvadrata brojeva od a do b. Korisnik treba unijeti vrijednosti a i b.
3. Napravii program za računanje sume kvadrata parnih brojeva i sumu kubova neparnih brojeva od a do b. Korisnik treba unijeti vrijednosti a i b.
4. Napraviti program koji će pitati korisnika koliko želi unijeti brojeva. Zatim program treba zahtjevati unos toliko brojeva (mogu biti i decimalni brojevi). Nakon što je korisnik unio sve brojeve, program treba ispisati aritmetičku sredinu kvadrata unesenih brojeva.

# NIZOVI - POLJA

- Niz je skup elemenata istog tipa. Elementi niza memorišu se u nizu susjednih memorijskih lokacija.
- Svakom pojedinačnom elementu niza može se pristupiti navođenjem jedinstvenog identifikatora (imena) niza i indeksa odgovarajućeg elementa niza.
- DEKLARACIJA NIZA
- Kao i varijable, niz moramo deklarisati prije nego ga upotrijebimo.
- Sintaksa deklaracije niza:

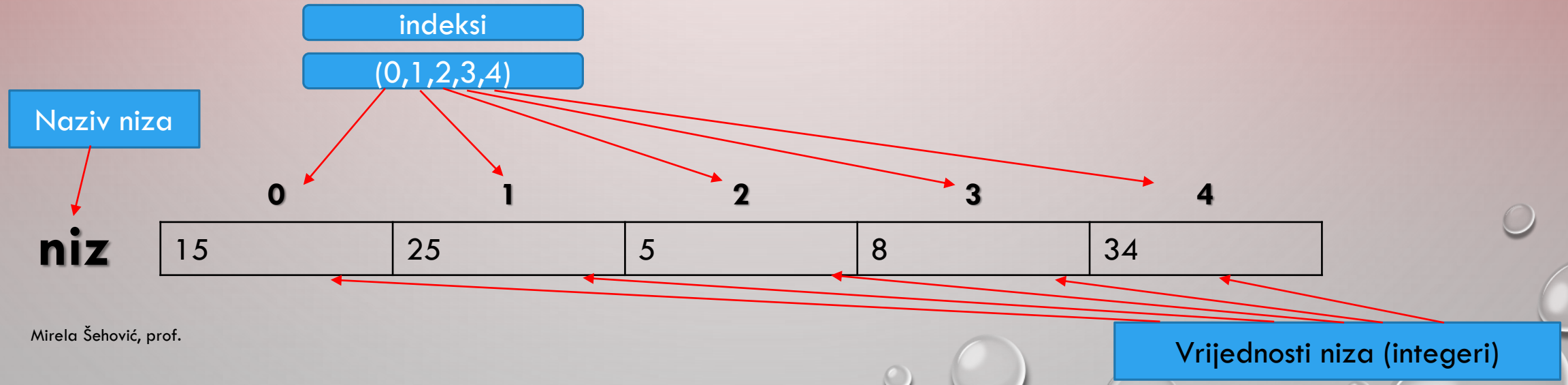
**tip identifikator [broj elemenata];**

- **Tip** može biti bilo koji validni tip podataka u C++ (npr. Int, float, char...);
- **Ime niza** je bilo koji validan identifikator u C++;
- U uglastim zagradama se nalazi broj elemenata niza (Kod statičkih nizova broj elemenata je obavezno neka konstantna vrijednost.)

- S naredbom int a=10; int b=4; int c=6; deklarišemo i inicijaliziramo tri integera a,b,c i tako da svaki od njih „pamti“ neku vrijednost. Ta vrijednost mora biti cijeli broj – integer. Svaka varijabla može pamtiti samo jednu vrijednost:

| | |
|---|---|
| a: | 10 |
| b: | 4 |
| c: | 6 |

**S naredbom int niz[5]; deklarišemo niz integera dužine 5. To znači, da sada niz može pamtiti više od jedne vrijednosti, jer se radi o nizu. Niz ni može pamtiti 5 vrijednosti (integera). Prva vrijednost se smješta na indeks poziciju 0, druga vrijednost na indeks poziciju 1,..., a peta vrijednost na indeks poziciju 4.**



- Sintaksa za stvaranje niza gdje koristi,o konstantnu vrijednost veličine niza:

```
const   int   velicina = 8;
  int   niz   [velicina];
```

- Deklarišemo konstantnu varijablu i u nju smjestimo veličinu niza. Razlika je jedino što umjesto upisivanja broja upisujemo varijablu kojoj smo prethodno dodijelili vrijednost,
- Praksa je pokazala da je najbolje koristiti ovaj vid deklaracije jer polje koje jednom stvorimo s određenim brojem elemenata nije moguće naknadno proširivati niti smanjivati. U memoriji će program zauzeti tačno određenu veličinu potrebnu za to polje.
- Ukoliko pri inicijalizaciji niza ostavite uglaste zagrade prazne kompajler će pretpostaviti da je veličina niza jednaka broju vrijednosti navedenih u vitičastim zagradama.

```
int   niz  [ ]  =  {5,6,12,25,87,1};
```

- Moguće je inicijalizovati manje članova nego što se deklariše, a vrijednost neinicijalizovanim članovima niza možemo kasnije dodijeliti:

- **int nizE[5]={7,6,5};//deklaracija niza sa 5 članova i inicijalizacija članova sa indeks pozicijom 0,1 i 2.**
- **nizE [3]=10;//izmjena vrijednosti članu sa indeksom 3**
- **cin >> nizE [9];//upis člana niza sa indeksom 3**
- **cin >> nizE [8];//upis člana niza sa indeksom 4**

- Ako pri deklaraciji niza vršimo i inicijalizaciju onda ne moramo navoditi dužinu niza, jer će kompajler kreirati niz onolike dužine koiliko smo inicijalizovali članova niza:

- **int nizE[ ]={1, 3, 5, 7, 9}; // deklaracija niza dužine 5 i inicijalizacija članova**

- Kako smjestiti neku vrijednost u niz na željenu indeks poziciju?
- To možemo izvršiti na sličan način kao i sa običnim varijablama, tako što nakon naziva niza (niz) slijede uglaste zagrade [ ] , u u njima vrijednost indeks pozicije.
- Ako želimo popuniti vrijednost u niz to bi smo uradili na sljedeći način:

```
niz [0]=15;
niz [1]=25;
niz [2]=5;
niz [3]=8;
niz [4]=6;
```

- Inicijalizacija članova niza pomoću unosa s tastature
- Želimo li unijeti elemente u niz koji se sastoji od unaprijed definisane veličine koristit ćemo for petlju. Unutar uvjet for petlja umjesto upisivanja broja do koliko da for petlja broj koristiti ćemo varijablu (konstantu) velicina.
- Dođe li do naknadne promjene u programu, ovakvim korištenjem smo osigurali tačnost prilikom stvaranja polja s ispravnim brojem elemenata.
- Primjer: Napisati program u kome će se deklarisati niz od 10 cijelih brojeva i u taj niz smjestiti vrijdnosti koje će unijeti korisnik preko tastature.

Start here | Primjer 17.cpp | **Primjer 18.cpp**

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  const int  velicina=10;
7
8
9  int niz [velicina];
10
11 cout<< "Unesite clanove niza "<< endl;
12 for (int i=0; i<velicina; i++)
13     cin>>niz[i];
14 return 0;
15 }
16
```

"C:\Users\Mirela\Desktop\Zadaci C++\Primjer 18.exe"

```
Unesite clanove niza
2
3
2
5
6
2
3
1
2
3

Process returned 0 (0x0)   execution time : 15.410 s
Press any key to continue.
```

ogs & others

Code::Blocks | Search results | Cccc | Build log | **Build messages** | CppCheck | CppCheck messages | Cscope | Debugger | DoxyB

| File | Line | Message |
|---|---|---|
| | | === Build file: "no target" in "no project" (compiler: unknown) === |
| | | === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) === |

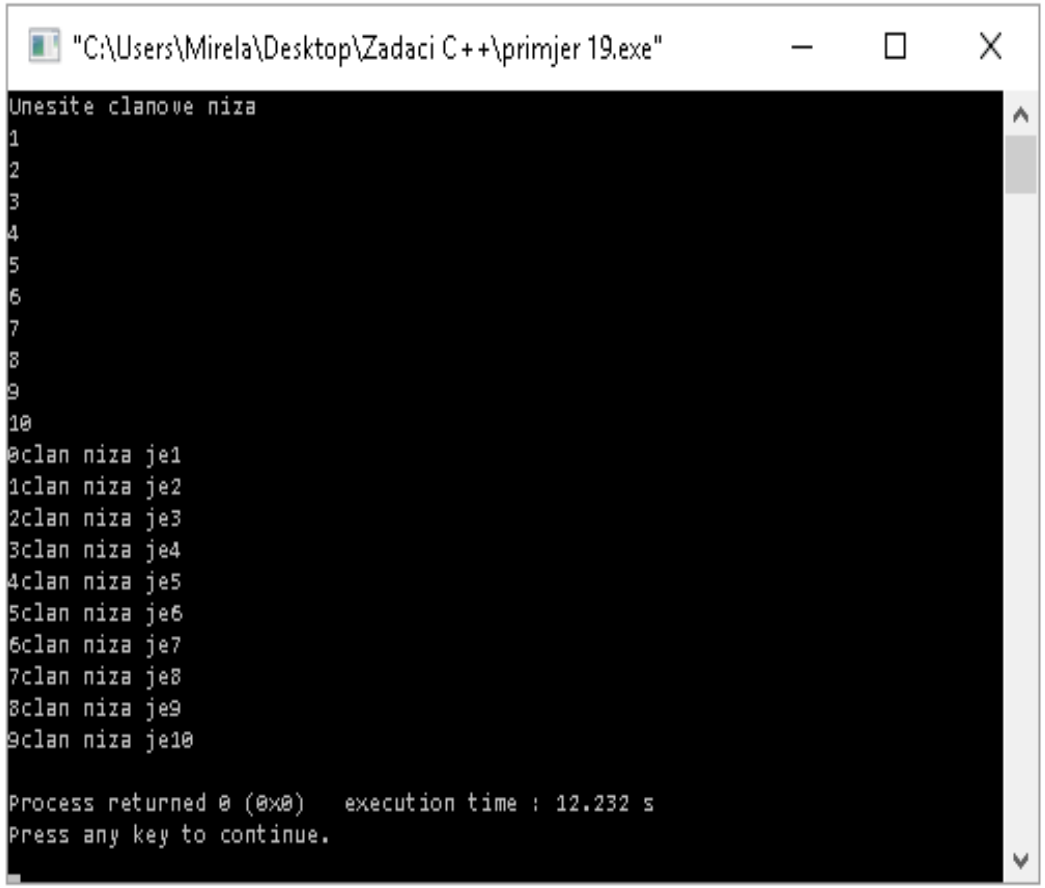
- Ispis vrijednosti elemenata niza
- Ispis elemenata niza vrüimo pomoću for petlje

```
for(int   i=0;   i<velicina;   i++);
cout<< i << „clan niza je „ << niz [i]<<endl;
```

- Primjer: Potrebno je prethodnom primjeru dodati ispis članova niza:

Start here | primjer 19.cpp

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

int main()
{
const int  velicina=10;


int niz [velicina];

cout<< "Unesite clanove niza "<< endl;
for (int i=0; i<velicina; i++)
    cin>>niz[i];

for (int i=0; i<velicina; i++)
cout<< i << "clan niza je"<< niz[i]<<endl;

return 0;
}
```



- Nad članovima (elementima) niza moguće je, ovisno o tipu podataka koje niz sadrži, vršiti: aritmetičke operacije, indekrement:

```
niz [0] =  2 * niz [1] + niz [2];
           niz [0]++;
           -- niz [1];
```

- Uporedbu (upotreba relacijskih operatora):

```
if (niz [0] > niz [2];
```

- Unos i ispis:

```
cin >>niz [0] ;
cout<<niz [0] ;
```

- Primjer: Napisati program u kome će se deklarisati niz od 10 cijelih brojeva koristeći unaprijed deklarisanu konstantu. Niz inicijalizirati vrijednostima parnih brojeva u rasponu od 2 do 20. Na kraju program treba ispisati sve vrijednosti niza na ekran.

Start here | primjer 20.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  const int  velicina=10;
7
8
9  int niz [velicina];
10
11
12 for (int i=0; i<velicina; i++)
13     niz[i]=2+2*i;
14
15
16 for (int i=0; i<velicina; i++)
17 cout<< i << "clan niza je"<< niz[i]<<endl;
18
19 return 0;
20 }
21
```

"C:\Users\Mirela\Desktop\Zadaci C++\primjer 20.exe"

```
0clan niza je2
1clan niza je4
2clan niza je6
3clan niza je8
4clan niza je10
5clan niza je12
6clan niza je14
7clan niza je16
8clan niza je18
9clan niza je20

Process returned 0 (0x0)   execution time : 0.033 s
Press any key to continue.
```

Logs & others

# ZADACI ZA SAMOSTALAN RAD

1. Napisati program koji će računati sumu članova niza kojeg sačinjava deset cjelobrojnih vrijednosti koje unosi korisnik preko tastature.
2. Napisati program koji će računati sumu članova niza s neparnim indeksom. Niz sačinjava 12 cjelobrojnih vrijednosti koje se unose s tastature.
3. Dat je cjelobrojni niz od 8 elemenata. Napisati program koji će omogućiti korisniku da unese vrijednosti članova niza, te provjeriti i ispisati koliko je članova niza veće od 10.

# FUNKCIJE

- Najbolji način razvoja i održavanja velikih programa jeste napisati program iz više manjih dijelova ***modula***.
- Moduli u C++ se nazivaju ***funkcije***.
- Osim lakšeg razvoja i održavanja programa, dodatni motiv za korištenje funkcija je mogućnost ponovne upotrebe jednom napisanog koda, te izbjegavanja pisanja istog koda unutar jednog programa više puta.
- Svaki C++ program je zapravo funkcija.
- Funkcija je temeljni matematički koncept.
- Npr. matematički zapis:

$$F(x)=x^2$$

# $F(x)=x^2$

- Ova funkcija se koristi za opisivanje funkcije kvadrata vrijednosti x.
- U ovom primjeru:
  - F-ime funkcije
  - X-varijabla koja se pojavljuje unutar zagrada i naziva se ***parametrom*** (argumentom) funkcije.
- Ponašanje funkcije se opisuje parametrom x*x tj. x na kvadrat.
- Dakle, ako je:

**x=2 slijedi f(x) = 4**
**x=3 slijedi f(x) = 9**
**x=4 slijedi f(x) = 16**

- Znači u prvom slučaju kada je x=2, funkcija vraća vrijednost 4 itd.
- Vrijednost koju funkcija vraća je određena ***definicijom funkcije i vrijednošću njenih parametara.***
- Fnkcije u C++ su slične matematičkim funkcijama i možemo ih podjeliti na:
  - ugrađene funkcije
  - Korisnički definisane funkcije
- Bilo koja od navedenih funkcija mora imati:
  - ***svoje ime,***
  - ***nakon čega slijede zagrade u koje se navode liste parametara;***
  - ***nakon liste parametara u tijelu funkcije (tijelo funkcije započinje i završava vitičastim zagradama{}) navode se naredbe koje opisuju ponašanje funkcije (definicija funkcije).***

# DEFINICIJA FUNKCIJE

- Sintaksa definicije funkcije:

```
tip_povratne_vrijednosti ime funkcije (lista parametara)
{
naredbe;
}
```

- Gdje je:
  - ***tip_povratne_vrijednosti*** tip vrijednosti koju funkcija vraća (int, double, char; može biti i void);
  - ***ime_funkcije*** naziv (identifikator) funkcije;
  - ***lista_parametar lista***, popis parametara; obavezno se navode i tip i ime parametra; važan redoslijed kojim se parametri navode;
  - ***naredbe*** su niz naredbi koje opisuju ponašanje funkcije; ukoliko povratni tip funkcije nije void- jedna od naredbi mora biti naredba return (izraz);
  - ***Definicija funkcije se piše izvan main funkcije!***

# POZIV FUNKCIJE

- Da bi ste mogli upotrijebiti definisanu funkciju potrebno ju je pozvati unutar glavnog programa (u funkciji main9 ili u nekoj drugoj funkciji.
- Funkcija se poziva imenom i popisom parametara:

**ime_funkcije (parametri);**

- gdje je:
  - ***ime_funkcije*** naziv (identifikator) funkcije;
  - ***parametri imena*** (identifikatori) parametara; važan je redoslijed parametara;
  - ***U pozivu funkcije se ne navode niti tip povratne vrijednosti funkcije, niti tipovi parametara!***

# POZIV FUNKCIJE

```
int main()
{
    int broj1;
    cout<<"Funkcija treba da kubira uneseni broj"<<endl,
    cout<<"Unesite broj"<<endl;
    cin>>broj1;
    cout<<funkcijaKub(broj1);
    return 0;
}
```

```
int funkcijaKub(int broj1)
{
    int rez;
    rez=broj1*broj1*broj1;
    return rez;
}
```

- Kad izvršavanje programa dosegne poziv funkcije (funkcijaKub(broj1)) parametar (argument) funkcije (u našem slučaju broj1) se evaluira i kopira u parametar broj1 u funkciji funkcijaKub. Izvršavanje programa se premješta iz funkcije main u funkciju funkcijaKub, te počinje izvršavanje funkcije funkcijaKub. Izračunava se izraz broj1*broj1*beoj1, a naredbom return izračunata vrijednost se vraća u glavni program.

- Ukoliko se definicija funkcije nalazi ispred glavnog programa nije potrebno pisati deklaraciju ili prototip funkcije.
- No, ukoliko najprije pišete glavni program, a tek nakon toga definiciju funkcije, nužno je napisati i deklaraciju (prototip) funkcije.

# DEKLARACIJA (PROTOTIP) FUNKCIJE

- Deklaracija (prototip) funkcije „govori“ kompajleru ime funkcije, tip podatka kojeg funkcija vraća, broj parametara koje funkcija očekuje, tip tih parametara, te redoslijed po kojem funkcija očekuje da primi parametre.
- Kompajler koristi deklaraciju (prototip) funkcije za validaciju poziva funkcije. Programski jezik C nije imao ovu mogućnost provjere, pa je bilo moguće nepropisno pozvati funkciju.
- Sintaksa deklaracije (prototipa) funkcije:

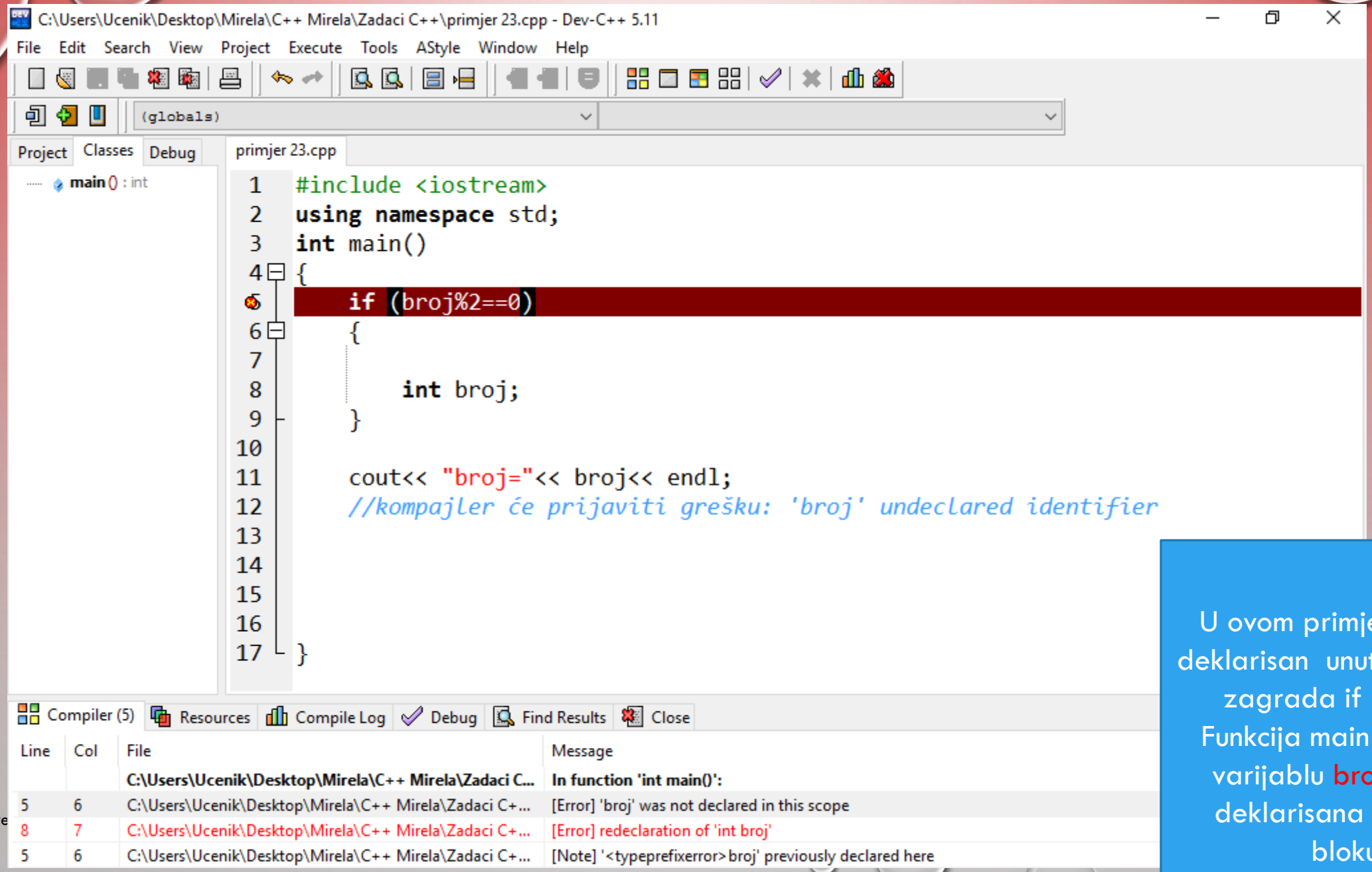
**tip_povratne_vrijednosti  ime_funkcije (lista parametara);**

- Gdje je:
  - ***tip_povratne_vrijednosti*** tip vrijednosti koju funkcija vraća (int, double, char; može biti i void);
  - ***ime_funkcije*** naziv (identifikator) funkcije;
  - lista_parametara lista, popis parametara; za razliku od definicije funkcije gdje je potrebno navesti i tip i ime parametra, u prototipu je dovoljno navesti tipove parametara, ukoliko navedete imena – kompajler će ih zanemariti; također, važan je redoslijed kojim se parametri navode;

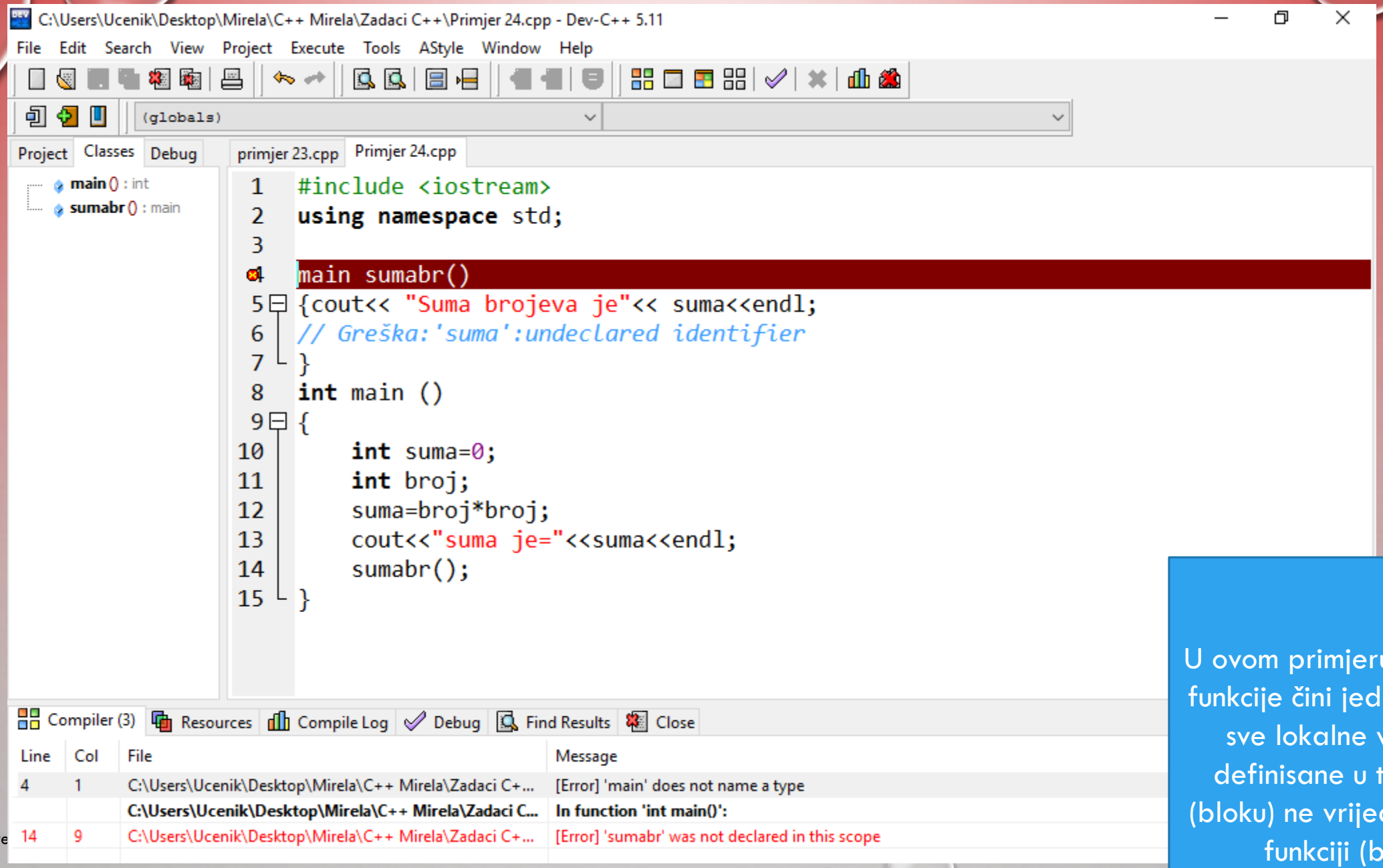
- Postoje dva tipa funkcija: korisnički definisane i igrađene. Ugrađene funkcije su dio paketa vašeg kompajlera (biblioteke funkcija) – njih obezbjeđuje proizvođač, primjer : sqrt(), pow(), sin(), cos()...
- Prilikom pokretanja nekog programa, prvo se izvršava funkcija main, bez obzira gdje se ona nalazila.
- Ako funkcija main pozove neku drugu pomoćnu funkciju onda će se tek izvršiti pozvana funkcija.

# OPSEG VAŽENJA LOKALNIH VARIJABLI

- Lokalna varijabla je varijabla koja važi unutar bloka u kojem je deklarisana. Varijabla definisana u jednom bloku ne vrijedi ( ne postoji) izvan tog bloka. (Blok čini bilo koji par vitičastih zagrada {}).
- Opseg važenja varijable je samo blok u kojem smo je deklarisali.



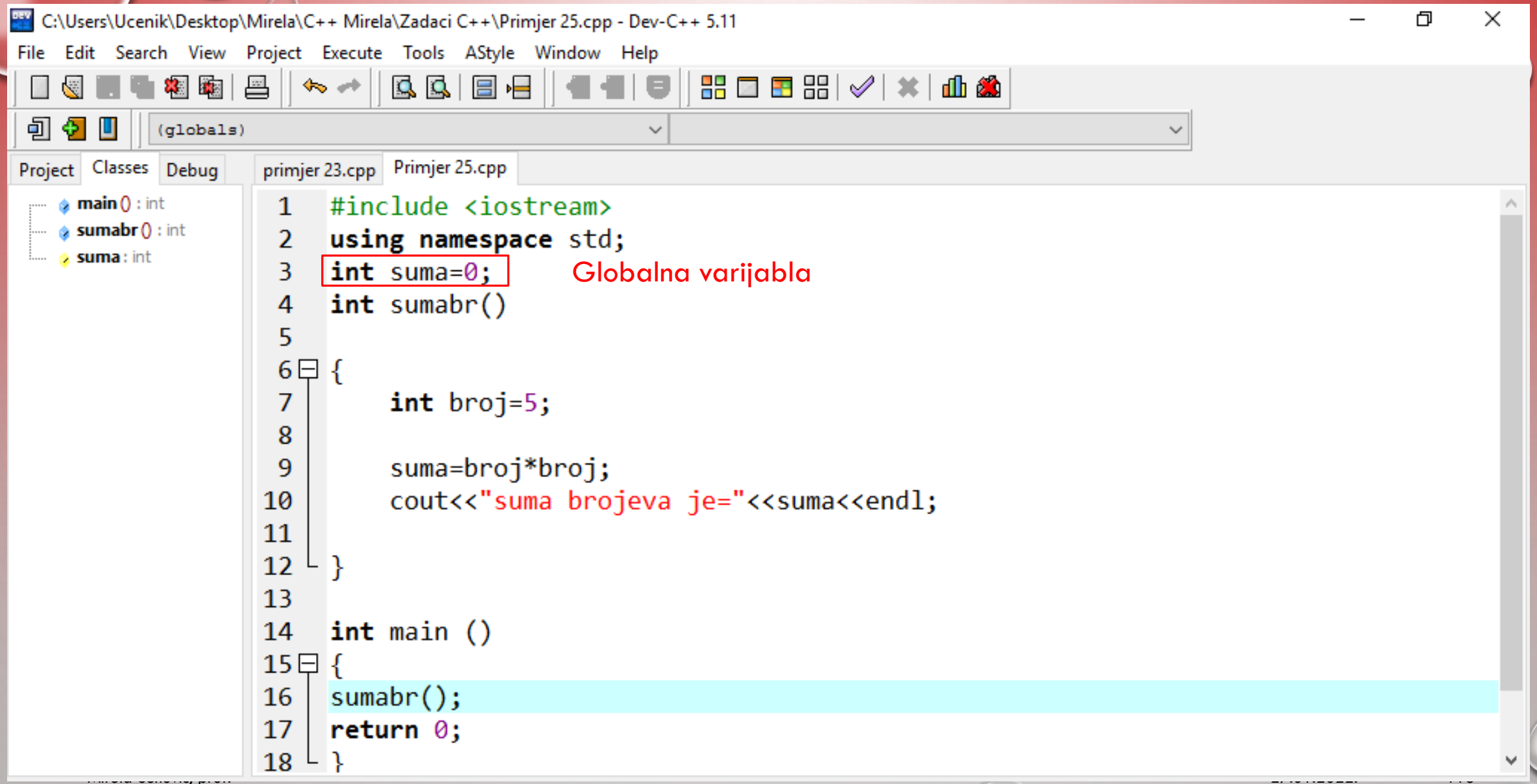

U ovom primjeru broj je deklarisan unutar vitičastih zagrada if funkcije. Funkcija main „ ne vidi“ varijablu broj, jer nije deklarisana u njenom bloku.

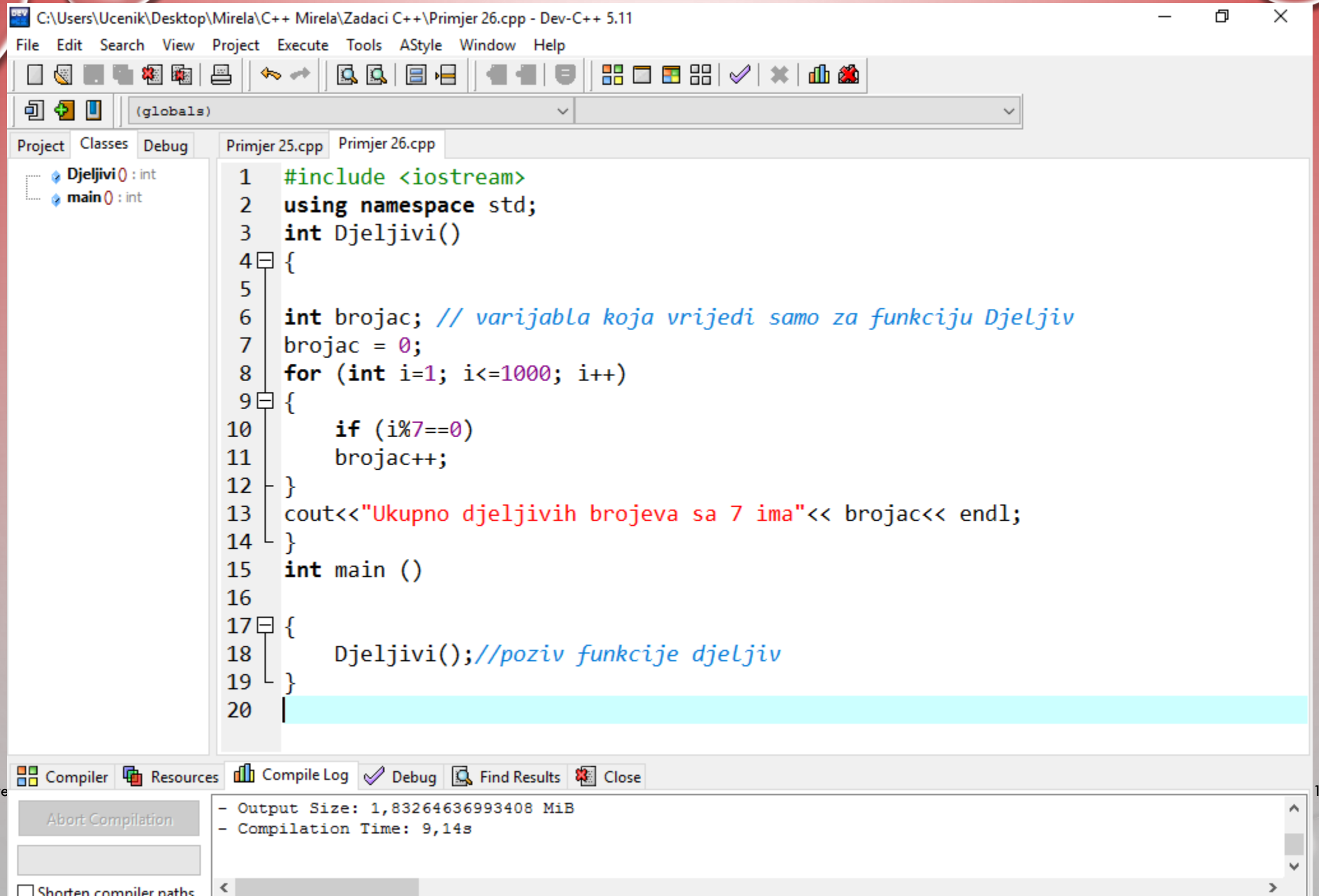



U ovom primjeru tijelo neke funkcije čini jedan blok, pa sve lokalne varijable definisane u toj funkciji (bloku) ne vrijede u drugoj funkciji (bloku).

# OPSEG VAŽENJA GLOBALNIH VARIJABLI

- Globalne varijable vrijede u glavnoj funkciji main i u svim ostalim pomoćnim funkcijama datog programa, jer se deklaracija globalne varijable ne vrši ni u jednom bloku.
- Globalne varijable se najčešće deklarišu na početku programa.

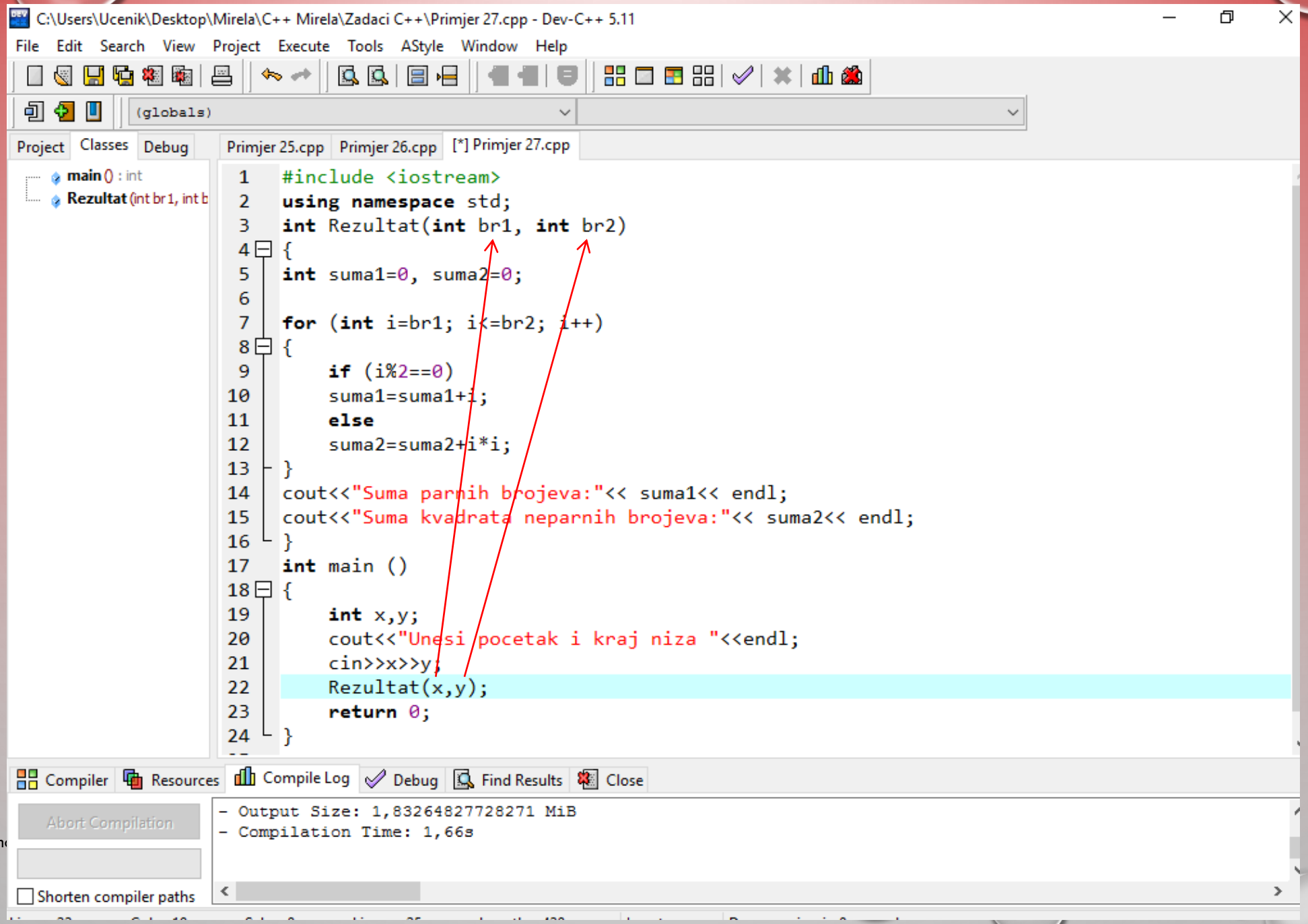

C:\Users\Ucenik\Desktop\Mirela\C++ Mirela\Zadaci C++\Primjer 25.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
primjer 23.cpp Primjer 25.cpp
main() : int
sumabr() : int
suma : int
1 #include <iostream>
2 using namespace std;
3 int suma=0;
Globalna varijabla
4 int sumabr()
5
6 {
7 int broj=5;
8
9 suma=broj*broj;
10 cout<<"suma brojeva je="<<suma<<endl;
11
12 }
13
14 int main ()
15 {
16 sumabr();
17 return 0;
18 }


C:\Users\Ucenik\Desktop\Mirela\C++ Mirela\Zadaci C++\Primjer 26.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
Djeljivi() : int
main() : int
Primjer 25.cpp
Primjer 26.cpp
#include <iostream>
using namespace std;
int Djeljivi()
{
int brojac; // varijabla koja vrijedi samo za funkciju Djeljiv
brojac = 0;
for (int i=1; i<=1000; i++)
{
if (i%7==0)
brojac++;
}
cout<<"Ukupno djeljivih brojeva sa 7 ima"<< brojac<< endl;
}
int main ()
{
Djeljivi();//poziv funkcije djeljiv
}
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
- Output Size: 1,83264636993408 MiB
- Compilation Time: 9,14s
Shorten compiler paths

# PRIMJER:

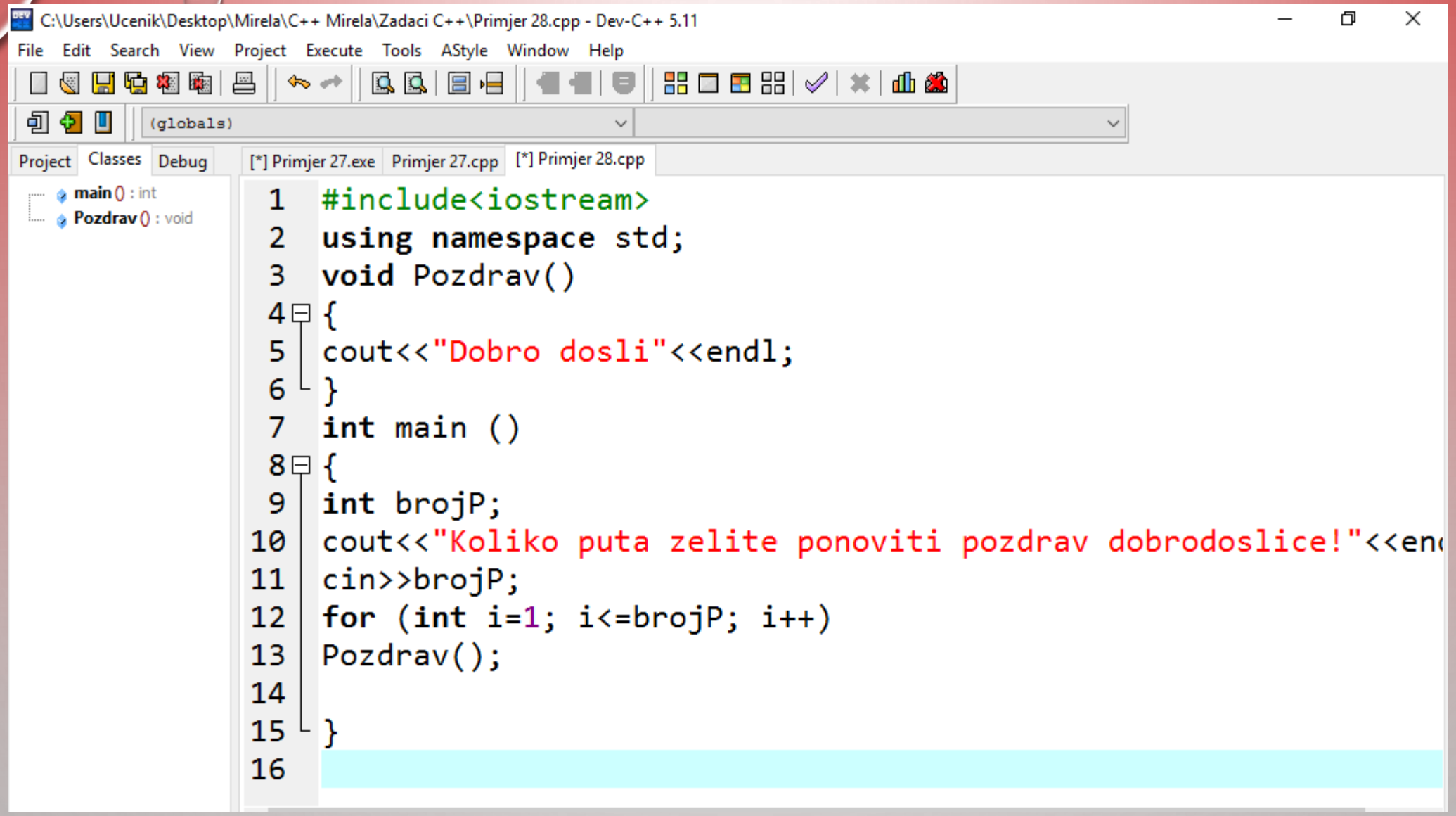
- Napraviti program u kome će se pomoću jedne funkcije Rezultat, ispisati suma parnih brojeva i suma kvadrata neparnih brojeva od m do n. U funkciji main će se od korisnika tražiti da unese cijeli broj x za početak niza i y za kraj niza.

- Funkcija može, ali i ne mora da ima argumente. Funkcija bez argumenata deklariše se sa praznim zagradama. Funkcija može da vraća rezultat, ali i ne mora.
- Funkcija koja nema povratnu vrijednost, deklariše se sa tipom void kao tipom rezultata.
- Funkcija može vratiti vrijednost koja je rezultat izraza u naredbi return.


C:\Users\Ucenik\Desktop\Mirela\C++ Mirela\Zadaci C++\Primjer 27.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
main () : int
Rezultat (int br1, int b
Primjer 25.cpp Primjer 26.cpp [*] Primjer 27.cpp
1 #include <iostream>
2 using namespace std;
3 int Rezultat(int br1, int br2)
4 {
5 int suma1=0, suma2=0;
6
7 for (int i=br1; i<=br2; i++)
8 {
9 if (i%2==0)
10 suma1=suma1+i;
11 else
12 suma2=suma2+i*i;
13 }
14 cout<<"Suma parnih brojeva:"<< suma1<< endl;
15 cout<<"Suma kvadrata neparnih brojeva:"<< suma2<< endl;
16 }
17 int main ()
18 {
19 int x,y;
20 cout<<"Unesi pocetak i kraj niza "<<endl;
21 cin>>x>>y;
22 Rezultat(x,y);
23 return 0;
24 }
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
- Output Size: 1,83264827728271 MiB
- Compilation Time: 1,66s
Shorten compiler paths

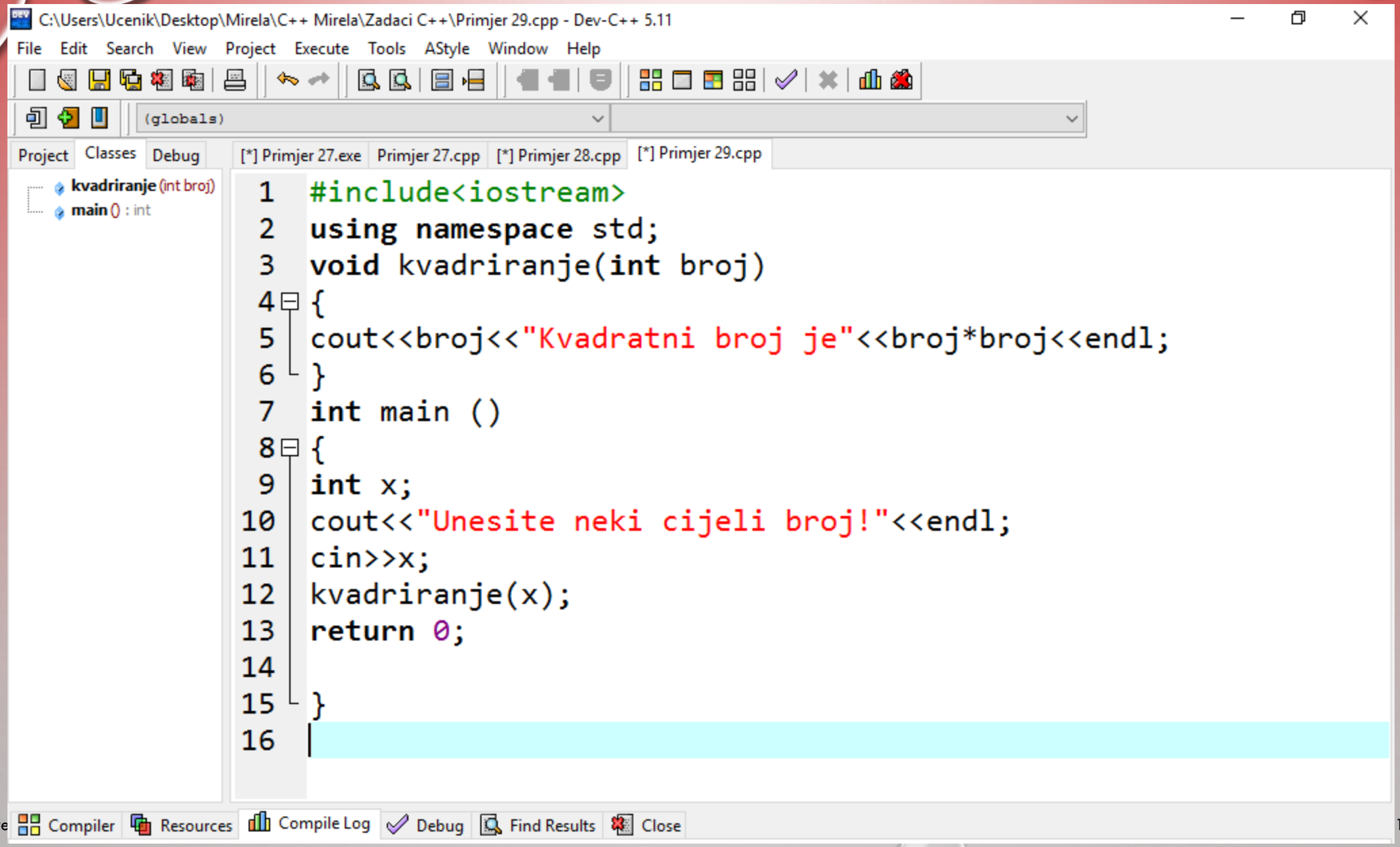
# FUNKCIJA TIPA VOID – FUNKCIJA ZADATKA

- Void funkcije se kreiraju i koriste slično kao i funkcije koje vraćaju vrijednost. Jdina razlika između void funkcija i funkcija koje vraćaju vrijednost je u tome što void funkcije ne vraćaju vrijednost nakon što se funkcija izvrši.
- Funkcije koje ne vraćaju vrijednost (void) se često koriste kako bi promijenile vrijednost referentnih parametara, promjenile vrijednost globalne varijable, te za obavljanje I/O operacija.
- Primjer:
- Napisati program koji će koristeći funkciju Pozdrav() ispisati poruku dobrodošlice „Dobro došli“ onoliko puta koliko korisnik želi.


C:\Users\Ucenik\Desktop\Mirela\C++ Mirela\Zadaci C++\Primjer 28.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
[*] Primjer 27.exe
Primjer 27.cpp
[*] Primjer 28.cpp
main () : int
Pozdrav () : void
#include<iostream>
using namespace std;
void Pozdrav()
{
cout<<"Dobro dosli"<<endl;
}
int main ()
{
int brojP;
cout<<"Koliko puta zelite ponoviti pozdrav dobrodoslice!"<<en
cin>>brojP;
for (int i=1; i<=brojP; i++)
Pozdrav();
}

# VOID FUNKCIJA KOJA PRIMA ARGUMENTE

- Iz funkcije main prosljeđujemo parametre u funkciju koja je nepovratna (void), međutim vrijednost argumenata ćemo iskoristiti za naše potrebe ali nećemo vratiti nikakvu vrijednost kao rezultat izvršavanj funkcije. U funkciju ispisujemo vrijednosti, dok je u man funkciji samo pozivamo.
- Primjer:
- Napraviti aplikciju koja sadži jednu jednostavnu funkciju. Aplikacija će pitati korisnika da unese neki cijeli broj te je potrebno taj broj predati kao argument funkciji koja će izvršiti kvadriranje broja i ispisati njegovu vrijednost.

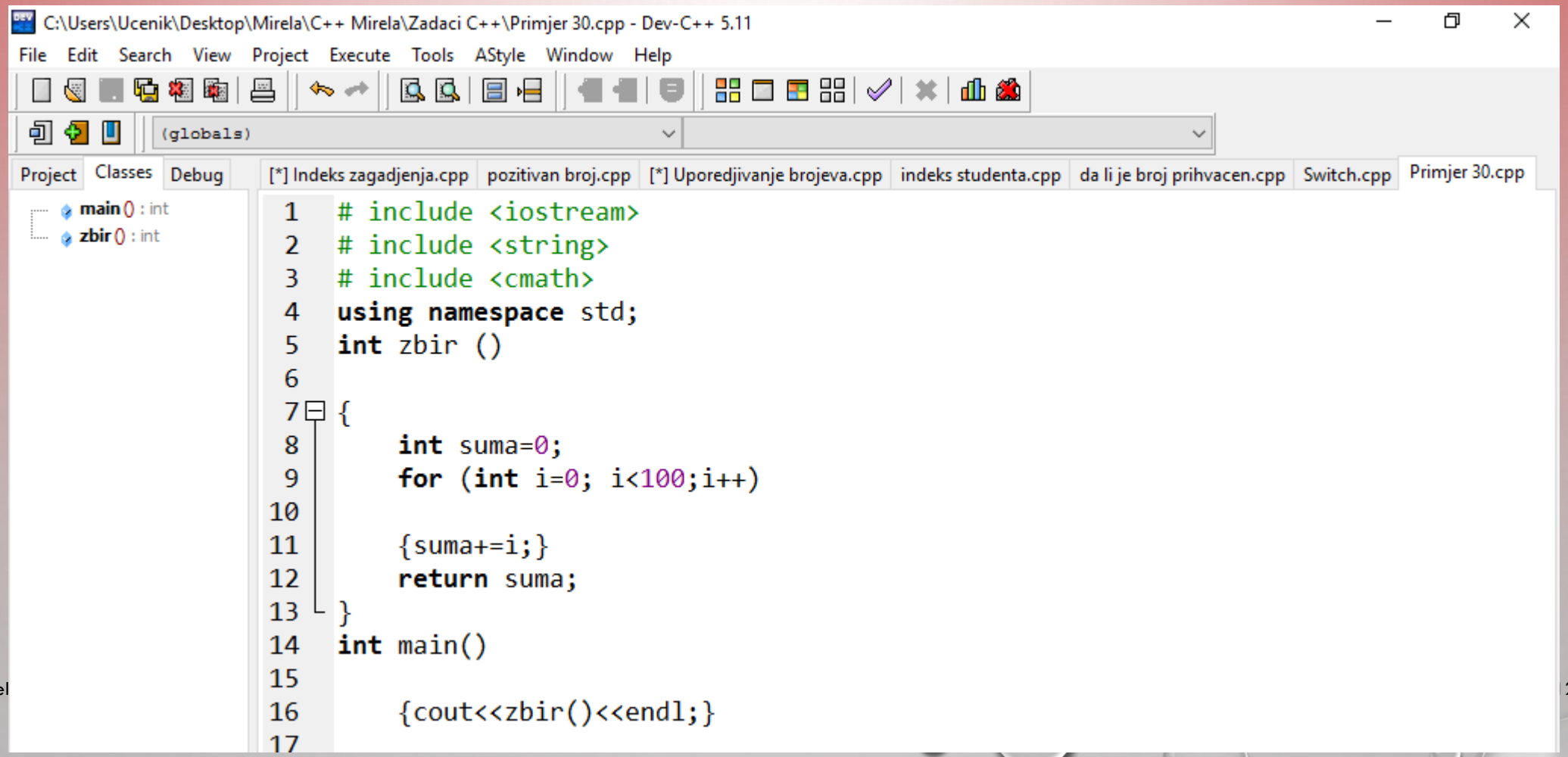

C:\Users\Ucenik\Desktop\Mirela\C++ Mirela\Zadaci C++\Primjer 29.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
[*] Primjer 27.exe Primjer 27.cpp [*] Primjer 28.cpp [*] Primjer 29.cpp
kvadriranje (int broj)
main () : int
1 #include<iostream>
2 using namespace std;
3 void kvadriranje(int broj)
4 {
5 cout<<broj<<"Kvadratni broj je"<<broj*broj<<endl;
6 }
7 int main ()
8 {
9 int x;
10 cout<<"Unesite neki cijeli broj!"<<endl;
11 cin>>x;
12 kvadriranje(x);
13 return 0;
14
15 }
16
Compiler Resources Compile Log Debug Find Results Close

# FUNKCIJA SA POVRATNOM VRIJEDNOŠĆU KOJA NE PRIMA ARGUMENTE

- Kako samo ime kaže, to su funkcije koje vraćaju neku vrijednosti. Ova vrijednost ima svoj tip kao i sve ostale vrijednosti u C++.
- Ona može biti integer, float, char ili bilo koji drugi tip.
- Tip povratne vrijednosti određuje tip naše funkcije.
- Funkcija vraća vrijednost naredbom return.

Primjer:

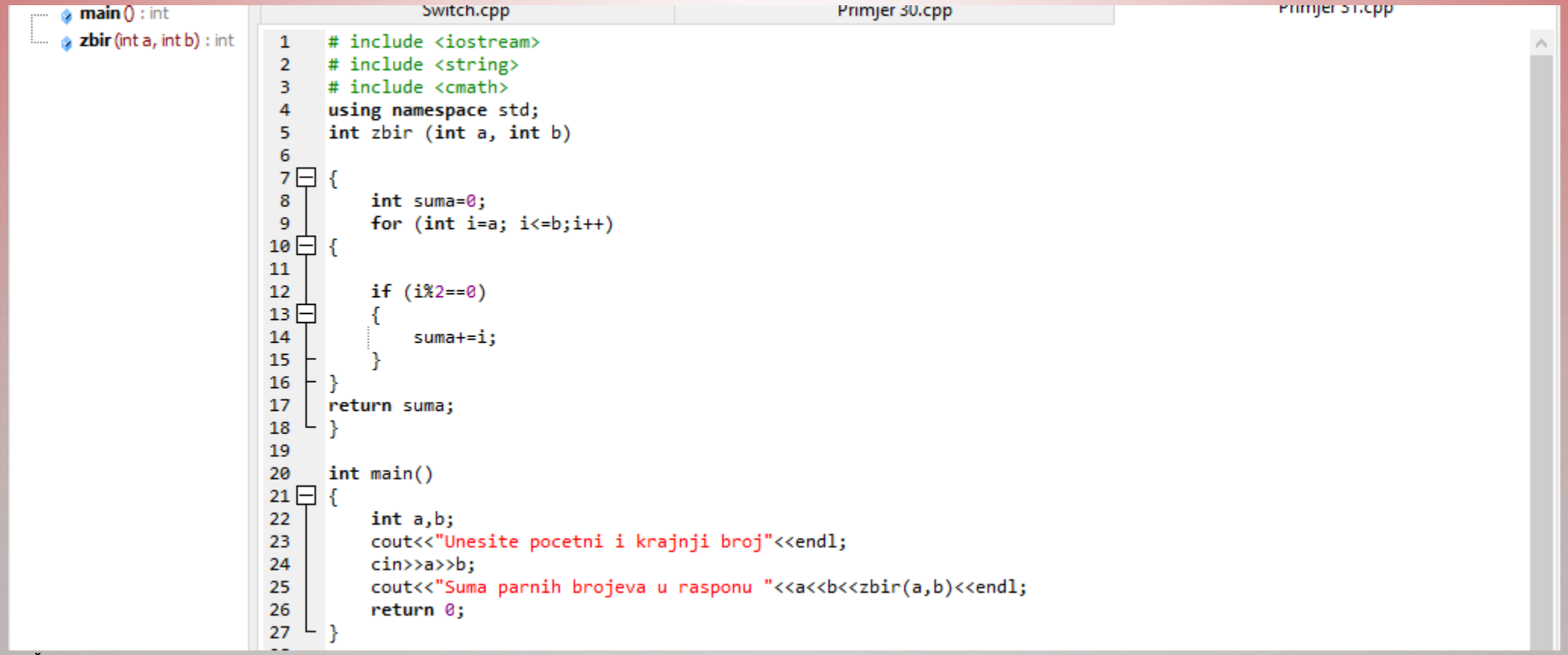
Napraviti program koristeći funkciju koja će vratiti zbir prvih 100 cjelobrojnih brojeva (od 0 do 99).



# FUNKCIJA SA POVRATNOM VRIJEDNOŠĆU KOJA PRIMA ARGUMENTE

- Funkcije koje ne primaju argumente nemaju veliku funkcionalnost jer nemaju interakciju s varijablama odnosno podacima unutar main funkcije.
- Funkcije koje primaju parametre omogućuju interakciju funkcija s ostatkom dijela programa.
- Postoje funkcije koje primaju argumente istog tipa i funkcije koje primaju argumente različitog tipa.

Primjer:
Napraviti program u kome će se u funkciji sumaparnihbrojeva() ispisati suma parnih brojeva u rasponu od a do b korisnik unosi u funkciju main.



# ZADACI ZA SAMOSTALNI RAD

- 1. Napraviti program koji će u funkciji main zahtjevati od korisnika da unese dimenzije bazena (širina, dužina, visina) u metrima i koji će pomoću funkcije izračunati i ispisati zapreminu bazena u litrima, Deklarisati prototip funkcije.
- 2. Napisati funkciju povecaj () čiji je zadatak da uveća vrijednost cjelobrojne varijable koja joj je proslijeđena kao parametar.
- 3. Napisati program koji će omogućiti korisniku unos dva broja x i y. Funkcija main treba da proslijedi parametre funkciji množenje (). Napraviti funkciju za množenje dva broja x*y koja reultat vraća u glavni program i ispisuje ga.
- 4. Napisati funkciju za traženje najmanjeg od tri učitana broja. U funkciji main omogućiti korisniku unos brojeva. Brojeve funkciji proslijediti kao parametre. Funkcija treba vratiti kao rezultat najmanji učitani broj. U funkciji main pozvati napisanu funkciju i ispisati njeno rješenje.

# FORMATIRANJE ISPISA NAREDBA \n

- Ako želimo da rečenicu ispišemo u dvije odvojene linije na konzoli moramo koristiti naredbu \n – prelazak u novi red. \n nam omogućuje prelazak u novi red usred neke rečenice. Koristi se unutar samog teksta koji želimo ispisati.
- Primjer:

cout<<„Danas je \n divan dan“;

Switch.cpp | Primjer 30.cpp

```
1  # include <iostream>
2  # include <string>
3  # include <cmath>
4  using namespace std;
5
6  int main()
7  {
8      cout<<"Danas je \n divan dan ";
9  }
```

```
Danas je
 divan dan
--------------------------------
Process exited after 3.047 seconds with return value 0
Press any key to continue . . .
```

# NAREDBA <<endl;

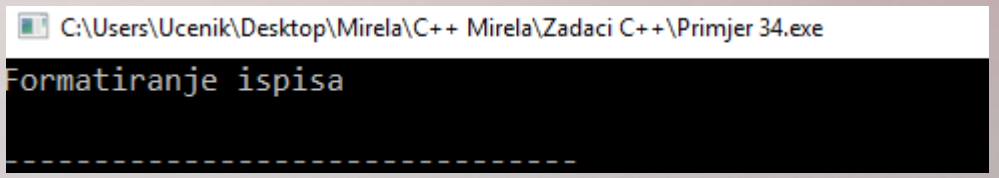
- Slično naredbi \n moguće je koristiti i <<endl naredbu (End – line), prelazak u novi red.
- Jedina razlika između ove dvije naredbe jeste što se naredba <<endl koristi na kraju kodne linije, a dok se naredba \n može koristiti u sredini linije.

```
1   # include <iostream>
2   # include <string>
3   # include <cmath>
4   using namespace std;
5
6   int main()
7   {
8       cout<<"Danas je"<<endl;
9       cout<<"divan dan "<<endl;
10  }
```

```
Danas je
divan dan

--------------------------------
Process exited after 0.251 seconds with return value 0
Press any key to continue . . .
```

Primjer 34.cpp

```cpp
# include <iostream>
# include <string>
# include <cmath>
using namespace std;

int main()
{
    cout<<"Formatiranje ispisa"<<endl;

}

```



# BIBLIOTEKA <iomanip>

- Da bi smo mogli koristiti naprednije opcije formatiranja ispisa morat ćemo uključiti biblioteku koja se zove iomanip. Uključivanje biblioteke vrši se na početku programa i izgleda ovako:

```
#include<iomanip>
```

- Primjer:

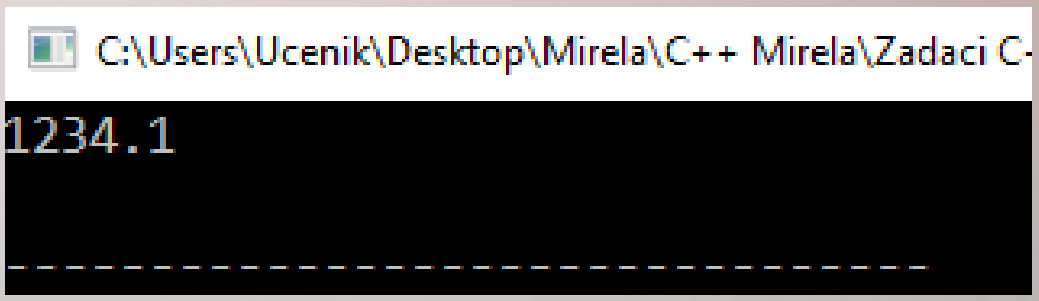
```
#include<iostream>
#include<iomanip>
using namespace std;
```

# FUNKCIJA setprecision()

- Funkcija setprecision precizno određuje maksimalni broj znamenki koje će se prikazati na konzoli, računajući obje znamenke prije i nakon decimalne tačke.
- Funkcija prima jedan argument.
- Funkciju koristimo tako da napišemo cout naredbu za ispis na konzolu, zatim<< iza čega slijedi funkcija i argument, setprecision (), pa opet<< i zatim broj ili varijabla koja ima brojčanu vrijednost koju želimo ispisati.

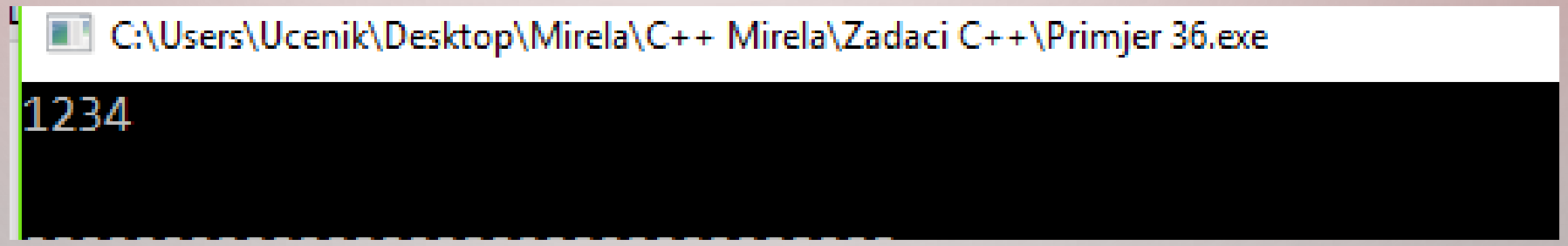
Primjer 35.cpp

```cpp
# include <iostream>
# include <iomanip>

using namespace std;

int main()
{
    float broj = 1234.1234;
    cout<<setprecision(5)<<broj<<endl;
    return 0;
}
```



Primjer 36.cpp

```cpp
# include <iostream>
# include <iomanip>

using namespace std;

int main()
{
    float broj = 1234.1234;
    cout<<setprecision(4)<<broj<<endl;
    return 0;
}
```



# FUNKCIJA width()

- Funkcija width nam omogućava da odredimo minimalnu širinu prilikom ispisa. Ukoliko naš ispis ne sadrži dužinu koju smo naveli kao minimalnu ova funkcija će u ispisu razliku nadopuniti simbolom za nadopunjavanje koji je po početnim vrijednostima prazno mjesto (eng.space). Funkcija prima jedan argument koji označava minimalnu potrebnu širinu prilikom ispisa.
- Sintaksa funkcije width:

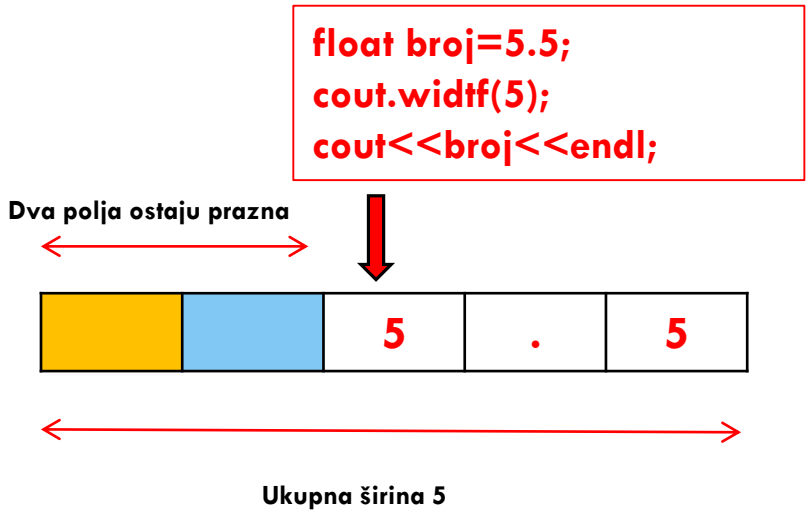
```
cout.width (neki_broj_tipa_integer);
cout<<ono_šta_ispisujemo;
```

- Primjer širine odnosi se samo na prvu cout naredbu koja slijedi nakon specifiranja širine.

Primjer 37.cpp

```cpp
# include <iostream>

using namespace std;

int main()
{
    float broj = 5.5;
    cout<<"1234"<<endl;
    cout.width (5);
    cout<<broj<<endl;
    return 0;
}
```

```
1234
  5.5
```



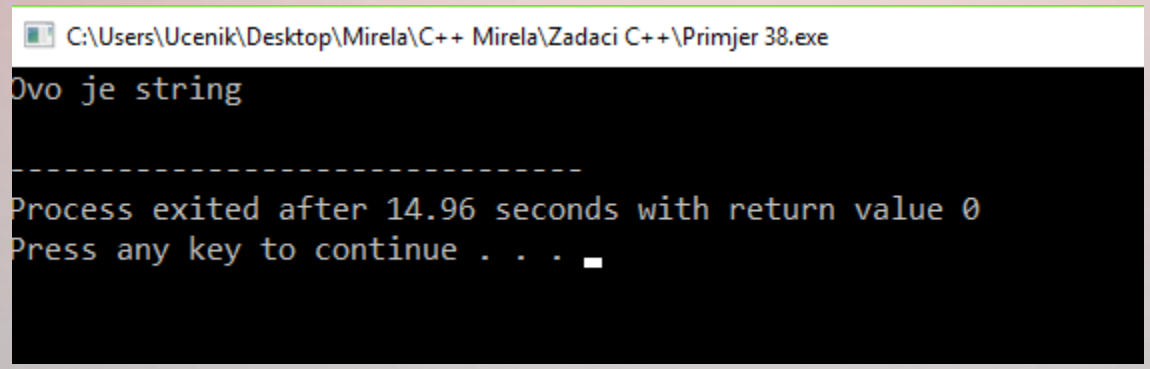
# TIP PODATKA - String

- Varijable tipa string su u mogućnosti pohraniti niz znakova, kao što su riječi ili rečenice. Sve što se nalazi pod dvostrukim navodnicima smatra se string tipom podataka.
- Da bi mogli koristiti ovaj tip podataka moramo uključiti biblioteku #include<string>.
- Sintaksa:

```
string ime varijable = „neka vrijednost“
```

Primjer 38.cpp

```cpp
# include <iostream>
#include<string>

using namespace std;

int main()
{
    string rijec;
    rijec="Ovo je string";
    cout<<rijec<<endl;
    return 0;
}
```



- Nad tipom podatka string mogu se vršiti razne operacije odnosno funkcije.
- String je nakupina slova, brojeva i simbola i s takvom nakupinom možemo raditi svašta.
- Pretvaranje u brojčane vrijednosti, čitanje samo jednog znaka, prebacivanje slova u nekoj riječi i slično samo su neke operacije koje možemo raditi sa stringovima.
- U nastavku slijede neke od funkcija koje je moguće primjeniti nad string tipom podatka.

# FUNKCIJA getline()

- Ako želite unijeti cijelu rečenicu u jedan string, onda morate koristiti funkciju getline().
- Funkcija cin kod unosa više riječi odbacuje sve riječi osim prve, što znači koliko god mi imali riječi u stringu, pomoću funkcije cin ispisati će se samo prva.
- Sintaksa funkcije getline():

```
getline(cin, varijabla);
```

- Primjer 1: Upotreba cin naredbe za unos:

Primjer 39.cpp

```
1   # include <iostream>
2   #include<string>
3
4   using namespace std;
5
6   int main()
7   {
8       string imeprezime;
9       cout<<"Unesi ime i prezime"<<endl;
10      cin>>imeprezime;
11      cout<<"Vaše ime i prezime je"<<imeprezime<<endl;
12      return 0;
13  }
```

```
Unesi ime i prezime
Mirela Sehovic
VaÜe ime i prezime jeMirela


--------------------------------
```

Primjer 1: Upotreba getline() funkcije za unos:

Primjer 40.cpp

```cpp
# include <iostream>
#include<string>

using namespace std;

int main()
{
    string imeprezime;
    cout<<"Unesi ime i prezime"<<endl;
    getline(cin,imeprezime);
    cout<<"Vaše ime i prezime je"<<imeprezime<<endl;
    return 0;
}
```

```
Unesi ime i prezime
Mirela Sehovic
VaÜe ime i prezime jeMirela Sehovic
```

# FUNKCIJA length()

- Uz pomoć funkcije length(), određujemo dužinu stringa. Dužina stringa se mjeri u simbolima i ova funkcija vraća kao rezultat brojčani podatak tipa integer.
- Funkcija ne prima nikakve argumente. U dužinu stringa su uključeni i razmaci.
- Sintaksa:

```
ime_varijable.length();
```

Primjer 41.cpp

```cpp
# include <iostream>
#include<string>

using namespace std;

int main()
{
    string imeprezime;
    cout<<"Unesi ime i prezime"<<endl;
    getline(cin,imeprezime);
    cout<<"Duzina stringa koji ste unijeli je:"<<imeprezime.length()<<endl;
    return 0;
}
```

```
Unesi ime i prezime
Mirela Sehovic
Duzina stringa koji ste unijeli je:14
```

- Svaki string se sastoji od slova, brojeva i/ili simbola i svaki od njih je moguće zasebno izdvojiti.
- String tip podatka možemo posmatrati kao polje elemenata koje također počinje sa indeksom 0 (nula) a svaki simbol, broj ili slovo u stringu se nalazi pod određenim indeksom.
- Uriječi „cvijet“ slovo „c“ se nalazi na indeksu 0 (nula), slovo „v“ se nalazi na indeksu 1 (jedan), slovo „i“ na indeksu 3 (tri) itd.
- Pristupamo im slično kao i polju elemenata.

Primjer 42.cpp

```cpp
# include <iostream>
#include<string>

using namespace std;

int main()
{
    string rijec;
    rijec="cvijet";
    cout<<rijec[0]<<endl;
    cout<<rijec[1]<<endl;
    cout<<rijec[2]<<endl;
    cout<<rijec[3]<<endl;
    cout<<rijec[4]<<endl;
    cout<<rijec[5]<<endl;
    return 0;
}
```

- Ovaj način ispisa slova u stringu možemo jedino koristiti kada korisnik inicijalizira varijablu.
- U ovom slučaju string rijec=„cvijet“.
- Ako od korisnika zahtjevamo da unese riječ tada nećemo znati kolika je dužina znakova u stringu i nećemo moći na ovaj način ga ispisati.
- U kombinaciji sa funkcijom length() i for petljom možemo ispisati sve znakove nekog stringa bez obzira na njegovu dužinu.

Primjer 43.cpp

```cpp
# include <iostream>
#include<string>

using namespace std;

int main()
{
    string rijec;
    cout<<"Unesi rijec"<<endl;
    cin>>rijec;

    int duzina=rijec.length();
    for (int i=0; i<duzina; i++)
    {
        cout<<rijec[i]<<endl;
    }
    return 0;
}
```

```
Unesi rijec
cvijet
c
v
i
j
e
t
```

# FUNKCIJA toupper() i tolower()

- Funkcija koja pretvara malo slovo u veliko i obrnuto. Ove funkcije ne pretvaraju riječ po riječ već slovo po slovo. To znači da ćemo morati proći kroz slovo unutar neke riječi i promijeniti ga u malo, odnosno veliko. Ako je slovo već bilo „malo“ tada će i ostati „malo“.
- Sintaksa:

```
tolower(karakter(jedno slovo) koje funkcija prima kao argument);
```

```
toupper (karakter(jedno slovo) koje funkcija prima kao argument);
```

- Da bi mogli promjeniti cijelu riječ iz malih u velika slova moramo koristiti for petlju da prođemo kroz cijelu riječ, funkciju length() da odredimo dokle da for petlja broji i u tijelu petlje ćemo pozvati funkciju tolower() koja će kao parametar primati i-ti indeks varijable rijec i mijenjati vrijednost na tom indeksu.

Primjer 44.cpp

```cpp
#include<iostream>
#include<cmath>
#include<string>

using namespace std;

int main()
{string rijec;
cout<<"Unesite rijec"<<endl;
cin>>rijec;

for (int i=0; i<rijec.length();i++)
{
    rijec[i]=tolower(rijec[i]);
}
    cout<<rijec<<endl;

}
```

```
Unesite rijec
MIRELA
mirela
```

# OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- U ovom poglavlju se neće detaljno , pa čak niti površinski obrađivati objektno orjentisano programiranje. Svrha ovog poglavlja je samo spomenuti da ovaj način kodiranja postoji, navesti prednosti i istaknuti da je danas, objektno orijentisano programiranje jedan od najzastupljenijih načina pisanja programa.

- Objektno orjentisano programiranje (Object Oriented Programming, OOP) je odgovor na tzv.krizu softvera. OOP pruža način za rješavanje (nekih) problema softverske proizvodnje.

- Softverska kriza je posljedica sljedećih problema softverske proizvodnj:
  - Zahtjevi korisnika su se drastično povećali. Za ovo su uglavnom „krivi“ sami programeri: oni su korisnicima pokazali šta sve računari mogu, i da mogu mnogo više nego što korisnik može da zamisli. Kao odgovor, korisnici su počeli da traže mnogo više, više nego što su programeri mogli da postignu.
  - Neophodno je povećati produktivnost programera da bi se odgovorilo na zahtjeve korisnika.To je moguće ostvariti najprije povećanjem broja ljudi u timu.
  - Produktivnost se može povećati i tako što se neki dijelovi softvera, koji su ranije već negdje korišteni, mogu ponovo iskoristiti, bez mnogo ili imalo dorade. Laku ponovnu upotrebu koda (software reuse) tradicionalni način programiranja nije dopuštao.
  - Povećani su drastično i troškovi održavanja. Potrebno je bilo naći način da projektovani softver bude što bolji i lakši za nadogradnju i modifikovanje.

  Za gore navedene promjene OOP je došlo kao odgvor.

  C++ je trnutno najpopularniji objektno orjentisani jezik.

- Osnovna rješenja koja pruža OOP, a C++ podržava su:

  - Apstrakcija tipova podataka (Abstract Data Types). Kao što u C-u ili nekom drugom jeziku postoje ugrađeni tipovi podataka (int, floa char, ...), u jeziku C++korisnik moŽe prizvoljno definisati svoje tipove i potpuno ravnopravno ih koristiti (complex, disk, point, printer,  jabuka, bankovni_racun, klijenti itd.).

  - Enkapsulacija (*encapsulation)*. Realizacija nekog tipa može (i treba) da se sakrije od ostatka sistema (od onih koji ga koriste). Treba korisnicima tipa precizno definisati samo šta se sa tipom može raditi, a način kako se to radi sakriva se od korisnika (definiše se interno).

  - Preklapanje operatora (operator overloading). Da bi korisnički tipovi bili sasvim ravnopravni sa ugrađenim i za njih se može definisati značenja operatora koji postoje u jeziku.

- Nasljeđivanje (inheritance). Pretpostavimo da je već formiran tip Printer koji ima operacije nalik na print_line, line_feed, format_feed, goto_xy itd. I da je njegovim korištenjem već realizovana velika količina softvera. Novost je da je firma nabavila i štampače koji imaju bogat skup stilova pisma i želja da se oni ubuduće koriste. Nepotrebno je ispočetka praviti novi tip štampača ili prepravljati stari kod. Dovoljno je kreirati novi tip PrinterWithFonts koji je „baš kao i običan“ štampač, samo „još može da“ mijenja stilove štampe.Novi će naslijediti osobine starog, ali će još ponešto moći da uradi.

- Polifonizam (*polymorphism*). Pošto je printerWithFonts već ionako Printer, nema razloga da ostatak programa ne „vidi“ njega kao i običan štampač, sve dok mu nisu potrebne nove mogućnosti štampača. Ranije napisani dijelovi koji koriste tip Printer ne moraju se uopšte prepravljati, oni će jednako dobro raditi i sa novim tipom. Pod određenim uslovima, stari dijelovi ne moraju se čak ni ponovo prevoditi! Karakteristike da se novi tip „odaziva“ na pravi način, iako ga je korisnik „pozvao“ kao da je stari tip , naziva se polimorfizam.