

Varijable i tipovi podataka

Ciljevi:

- proučiti i shvatiti pojam varijable
- shvatiti način na koji se radi s varijablama
- shvatiti odnos varijable i tipova podataka
- upoznati se sa tipovima podataka
- shvatiti osnovne operacije nad varijablama

Pregled lekcije

Pretpostavimo da nam je zadatak neki broj uvećati za jedan. Matematički gledano stvar je trivijalna a ništa teže nije ni izraditi pripadajući C++ kod.

Ono što je potrebno napraviti na samom početku je kroz određeni niz koraka opisati algoritam kojim ćemo doći do rješenja zadanog problema. U sljedećem primjeru dan je pseudokod algoritma danog problema.

Primjer 1.

Ulaz: Cijeli broj A

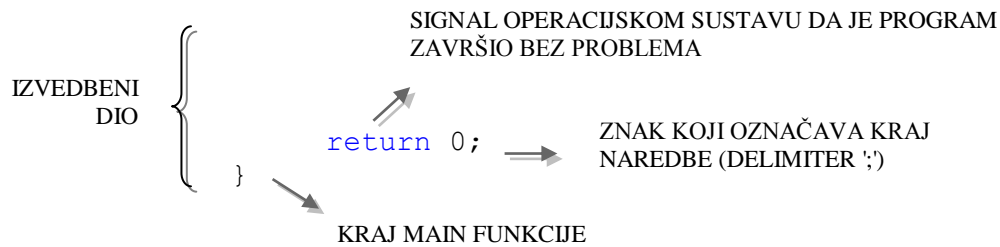
Izlaz: Broj A uvećan za 1

1. Učitaj broj A 2. Uvećaj broj A za 1 3. Ispiši broj A	}	PSEUDOKOD RJEŠENJA ZADANOG PROBLEMA
---	---	--

Prema prethodnom primjeru odredili smo da ćemo u tri koraka doći od početnog problema do konačnog cilja. U prvom koraku ćemo učitati neki broj sa tipkovnice, nakon toga ćemo taj broj uvećati za jedan i na kraju ćemo taj uvećani broj treba ispisati na ekranu računala. Međutim, da bismo sve to uradili morat ćemo koristiti između ostalog i varijable te tipove podataka. Ako ubacimo naš algoritam u C++ program dobit ćemo sljedeći rezultat.

Primjer 1.1.

			UKLJUČUJEMO BIBLIOTEKU FUNKCIJA
DEKLARACIJSKI DIO	←	#include <iostream>	
			POČETAK MAIN FUNKCIJE
		int main () {	⇒
IZVEDBENI DIO	{	1. Učitaj broj A 2. Uvećaj broj A za 1 3. Ispiši broj A	} LOGIKA NAŠEG PROGRAMA KOJU TREBA RAZRADITI



Navedeni primjer prikazuje naš algoritam , odnosno pseudokod, u okruženju standardnih dijelova koje sadrži svaki C++ program.

Svaki C++ program se sastoji od *deklaracijskog* i *izvedbenog dijela*. U deklaracijskom dijelu se nalaze predprocesorske i deklarativne naredbe. Ovaj dio uključuje definicije biblioteka koje se uključuju u program, konstanti, tipova podataka i drugih elemenata bitnih za rad samog programa. U izvedbenom dijelu se nalazi izvršni dio programa koji sadrži realizaciju algoritma odnosno samo programsko rješenje. Njega čine funkcije s time da je jasno istaknuta glavna funkcija od koje sve počinje a to je funkcija `'main ()'`. Dakle svaki put kada se program pokreće, pokreće se upravo funkcija `'main ()'` a tek nakon toga se izvršava ostatak koda odnosno ostale funkcije koje možemo a i ne moramo imati. Svaki C++ program, bez obzira kako velik bio mora sadržavati točno jednu `'main ()'` funkciju.

U našem programu na samom početku imamo predprocesorsku naredbu `'#include <iostream>'` što znači da želimo u program uključiti biblioteku u kojoj se nalaze funkcije za rad sa standardnim ulazno-izlaznim jedinicama kao što su tipkovnica i zaslon (dakle, naredbe za unos i ispis podataka). Kako je programski jezik C++ nastao od programskog jezika C, koji je prvenstveno bio namijenjen izradi programa vezanih uz operacijski sustav, jedno od važnih njegovih svojstava jest minimalnost, tj. zahtjev da izvršni kod bude što manji i kompaktniji. Stoga se u samoj jezgri programskog jezika C++ nalaze samo najosnovnije naredbe, dok se sve dodatne naredbe i funkcije nalaze u bibliotekama i uključuju se po potrebi. Kasnije ćemo vidjeti i neke druge predprocesorske naredbe koje se javljaju u programskom jeziku C++. Zajedničko svim predprocesorskim naredbama programskog jezika C++ je to što uvijek započinju znakom `#`. Za razliku od ostalih deklarativnih i izvedbenih naredbi programskog jezika C++, koje se interpretiraju tijekom prevođenja (kompilacije), predprocesorske se naredbe interpretiraju prije samog prevođenja.

Nakon deklarativnog dijela slijedi izvedbeni dio koda odnosno glavna funkcija `'main ()'` u kojoj se nalazi naš algoritam. U toj funkciji, osim našeg trenutno algoritma popisanog pseudokodom nalazi se i naredba `'return 0'`. Naredba `return` je naredba kojom se završava rad funkcije i vraća rezultat programu ili funkciji koja ju je pozvala. Glavna funkcija našeg programa pozvana je od strane operacijskog sustava, pa će se operacijskom sustavu vratiti vrijednost 0. Pravilo dobrog programiranja je da se kod uspješnog završetka rada programa operacijskom sustavu uvijek vraća vrijednost 0. Ova naredba, a i sama `'main ()'` funkcija je možda trenutno mnogima zbunjujuća, no za sada prihvatite da to tako mora biti a kako budemo napredovali s lekcijama bit ćete u mogućnosti bolje shvatiti i taj dio.

Isto tako, nakon svake naredbe, osim predprocesorskih naredbi u C++ programskom jeziku mora ići znak ';' koji se zove delimiter. Predprocesorske naredbe ne trebaju takav delimiter jer se one uvijek nalaze u jednom retku i završavaju prelaskom u novi redak. Dok god ne stavimo delimiter C++ sve interpretira kao jednu naredbu. Dakle, moguće je jednu naredbu koja nije predprocesorska pisati kroz više redaka, ali isto je tako moguće i više takvih naredbi pisati u jednom retku, sve dok se iza svake od njih nalazi delimiter ';'.

Na samom kraju ono što nam je ostalo osim algoritma su vitičaste zagrade. Naime, svaka funkcija odnosno blok naredbi se uvijek nalazi u vitičastim zagradama. Taj dio koda se uvijek izvršava slijedno liniju po liniju i zbog toga se takav niz naredbi naziva *slijed* ili *sekvenca*.

Treba napomenuti da programski jezik C++ kod pisanja naredi razlikuje velika i mala slova. To znači da se naredbe programskog jezika moraju pisati upravo onako kako ih ovdje navodimo – malim slovima. Pisanje neke naredbe velikim slovima uzrokovat će neprepoznavanje naredbe od prevodioca.

Pojam varijabla i tip podatka će biti objašnjen kroz osvrt na programski primjer koji slijedi.

Primjer 1.2.

```
// deklarativni dio programa
#include <iostream>

// glavna, main funkcija
int main()
{
    // deklaracija varijable A
    int A;

    // ispis na ekran
    std::cout << "Unesite vrijednost varijable A= ";
    // unos sa tipkovnice u varijablu A
    std::cin >> A;

    // varijabla A se povećava za jedan
    A = A + 1;

    // ispis uvećane varijable A
    std::cout << "Nova varijabla A= " << A << std::endl;

    // zaustavljanje programa
    system("pause");

    // aplikacija je regularno završila i stoga vraćamo
    // nulu
    return 0;
}
```

VARIJABLA A
TIPA INT ←

NAREDBA ZA ISPIS NA
ZASLON, NALAZI SE U
IOSTREAM BIBLOTECI ←

NAREDBA ZA UNOS
PODATAKA U VARIJABLU,
DAKLE OVDJE UNOSIMO
BROJ U VARIJABLU A ←

POKREĆEMO
KOMANDU PAUSE
OPERACIJSKOG
SUSTAVA ←

ZBRAJAMO VARIJABLU A I BROJ 1 TE
REZULTAT SPREMAMO U VARIJABLU A

ISPISUJEMO SADRŽAJ VARIJABLE A

Prethodni primjer u potpunosti rješava naš zadatak s početka. Nakon deklarativnog dijela, u kojem u program uključujemo biblioteku `iostream`, dolazi funkcija `'main ()'` u kojoj se nalazi naš algoritam. Na početku deklariramo varijablu `A` (`int A`). Varijabla nije ništa drugo nego mjesto gdje možemo spremati naše podatke odnosno brojeve, znakove i sl. Drugim riječima varijabla je imenovani dio memorije. Dakle, varijabla predstavlja dio memorije koji ima svoj naziv i unutar kojeg možemo spremiti neki podatak. Svaka varijabla ima svoj identifikator (ime) i tip podataka. Za razliku od varijabli, kako se definira u matematici, vrijednost varijable u programiranju može se tijekom rada programa proizvoljno mnogo puta promijeniti. Nadalje, napomenimo ponovo da programski jezik C++ razlikuje velika i mala slova, tako da se u programu varijabla mora nazivati točno onako kako je navedena u deklaraciji. Drugim riječima, u programskom jeziku C++ varijable `A` i `a` su dvije različite varijable.

Pravila za imenovanje varijabli su sljedeća:

- Identifikatori varijabli mogu se sastojati od malih i velikih slova, brojki i specijalnog znaka
- Prvi znak u identifikatoru varijable ne smije biti brojka

Sljedećom naredbom ispisujemo tekst `"Unesite vrijednost varijable A= "` na zaslon. Cijelu ovu naredbu treba dodatno objasniti. Ova se naredba promatra kao izraz, kojem je s lijeve strane tok podataka koji opisuje jedinicu na koju se podatak ispisuje, a s desne strane operatora podatak koji želimo ispisati. U programskom jeziku C++ `cout` predstavlja izlazni tok podataka vezan uz standardnu izlaznu jedinicu, tj. zaslon, dok je `<<` operator u izrazu koji predstavlja prosljeđivanje podataka s desne strane izraza na izlaznu jedinicu. Naime, programski jezik C++ s okolinom komunicira preko tokova podataka koji teku od programa prema okolini i od okoline prema programu. Tokove podataka možemo shvaćati kao sučelja koja povezuju naš program s okolinom u kojoj se nalazi. U ovom slučaju želimo ispisati gore navedeni tekst. U takvom slučaju se tekst koji ispisujemo mora staviti unutar dvostrukih navodnika.

Sljedeća naredba također predstavlja ulazno-izlaznu naredbu, no ovaj puta želimo učitati podatak s tipkovnice. Naredba je po svom formatu slična prethodnoj. Dakle i nju promatramo kao izraz. Operator ovog izraza jest `>>`, operator koji označava čitanje podataka s ulaznog toka podataka. S `cin` se u programskom jeziku C++ označava standardna ulazna jedinica, tj. tipkovnica. Kod izraza za učitavanje podataka kroz ulazni tok podatka s lijeve strane izraza mora se nalaziti identifikator varijable u koju želimo upisati učitane vrijednosti.

Sljedeća naredba također predstavlja izraz. Glavni operator ovog izraza je operator `=`. Odmah treba reći da ovaj operator u programskom jeziku C++ ima drugačije značenje nego u matematici. Dok u matematici znak `=` predstavlja logički operator koji uspoređuje vrijednost lijeve i desne strane izraza, u programiranju ovaj operator predstavlja operator pridruživanja vrijednosti s lijeve strane varijabli koja se nalazi na lijevoj strani izraza. Kao i svi drugi operatori u programskom jeziku C++ i operator `=` se može povezivati u veće izraze. Tako je, npr. izraz `a=b=c=d+1` u potpunosti regularan izraz u kojemu će se vrijednost varijable `d` uvećana za 1 upisati u varijable `a`, `b`, i `c`. Postoji jedno ograničenje na izraz koji sadrži operator `=`: jedino što se

smije nalaziti s lijeve strane operatora `=` jest identifikator varijable. Tako, na primjer izraz `a=b=c+1=d+2` ne bi bio dobar izraz u programskom jeziku C++ jer se s lijeve strane posljednjeg operatora `=` ne nalazi identifikator varijable, već izraz. S desne strane operatora `=` nalazi se aritmetički izraz `A + 1`, kojim se vrijednost varijable povećava za 1. Aritmetički se izrazi u programskom jeziku C++ tvore vrlo slično kao i u matematici. Za sada ćemo objasniti tek jednostavne aritmetičke izraze, dok ćemo se složenijima baviti u sljedećoj lekciji. U programskom jeziku C++ postoji 5 binarnih aritmetičkih operatora:

Tabela 1: Aritmetički operatori

Operator	Značenje	Primjer	Red
+	Zbrajanje	<code>A + B</code>	1
-	Oduzimanje	<code>A - B</code>	1
*	Množenje	<code>A * B</code>	2
/	Dijeljenje	<code>A / B</code>	2
%	Modulo	<code>A % B</code>	2

U tablici su opisani osnovni binarni aritmetički operatori, koji su, osim posljednjeg svi preuzeti iz matematike, pa ih nije potrebno posebno opisivati. Posljednji operator je pomalo specifičan. Za razliku od ostalih operatora, koji se mogu primjenjivati kako na cijele tako i na decimalne brojeve, posljednji se operator primjenjuje samo na cjelobrojne operande i vraća ostatak od cjelobrojnog dijeljenja prvog operanda s drugim. Posljednji stupac u tablici daje red operacije, što je važno kod složenijih izraza, kako bi se odredilo koji će se od više operatora u nizu prvi izvesti.

Sljedeća naredba je drugačija od prije opisanih ulazno izlaznih naredbi po tome što se u njoj kroz standardni izlazni tok podataka šalju tri različita podatka. Najprije se šalje tekst, nakon toga vrijednost varijable `A`, a na kraju vrijednost konstante `endl`. Kod ovakvih uzastopnih slanja podataka kroz isti ulazni ili izlazni tok nije potrebno svaki podatak slati posebnom naredbom, već ih je moguće sve poslati u jednoj naredbi. U tom slučaju se ti podaci moraju međusobno razdvojiti operatorom `<<`, odnosno `>>`. Nadalje, posljednji podatak koji se šalje kroz standardni izlazni tok podataka je vrijednost konstante `endl`. Konstanta `endl` je definirana u biblioteci `iostream` kao konstanta tip `char` koja sadrži vrijednost `'\n'`, odnosno 13, koja označava prijelaz u novi red. Dakle, konstanta `endl` će na zaslonu uzrokovati prelazak kursora u novi red.

Ono što nam je još preostalo za objasniti je naredba `'system("pause")'`. Funkcija `system` je funkcija preko koje se pozivaju komande operacijskog sustava. Naredba `pause` je naredba DOS operacijskog sustava koja ispisuje poruku `Press any key to continue...` te zaustavlja daljnje izvođenje programa sve dok korisnik ne pritisne neku tipku na tipkovnici. Treba upozoriti da se funkcija `system` ne nalazi u jezgri programskog jezika C++, već u biblioteci `cstdlib`. Sučelje DEV-C++ koje mi koristimo automatski pridružuje biblioteku `cstdlib` svakom C++ programu, no treba upozoriti da to ne čine sva programska sučelja za programiranje u programskom jeziku C++. Želimo li biti sigurni da će naš program raditi u svakom programskom

okruženju, dobro je, koristimo li ovu funkciju dodati u deklaracijskom dijelu programa predprocesorskom naredbom `#include <cstdlib>`.

Primijetimo da se ispred svakog toka podataka koji smo spominjali, ali i ispred konstante `endl` u programu nalazi `std::`. To znači da je tok podataka koji koristimo definiran u klasi `std`, koja se naziva prostor imena. Kako ne bismo ovo morali pisati svaki puta kada koristimo nešto što je opisano u ovom prostoru imena, u deklaracijskom dijelu programa možemo navesti da se sva imena nalaze u prostoru imena pod nazivom `std`, kako je to učinjeno u sljedećem primjeru.

Primjer 1.3.

```
// deklarativni dio programa
#include <iostream>
using namespace std;

// glavna, main funkcija
int main()
{
    int A;

    // ispis na ekran
    cout << "Unesite vrijednost varijable A= ";
    // unos sa tipkovnice u varijablu A
    std::cin >> A;

    // varijabla A se povećava za jedan
    A = A + 1;

    // ispis uvećane varijable A
    std::cout << "Nova varijabla A= " << A << std::endl;

    // zaustavljanje programa
    system("pause");

    // aplikacija je regularno završila i stoga vraćamo nulu
    return 0;
}
```

Opišimo sada detaljnije tipove podataka u programskom jeziku C++.

Tip podataka definira

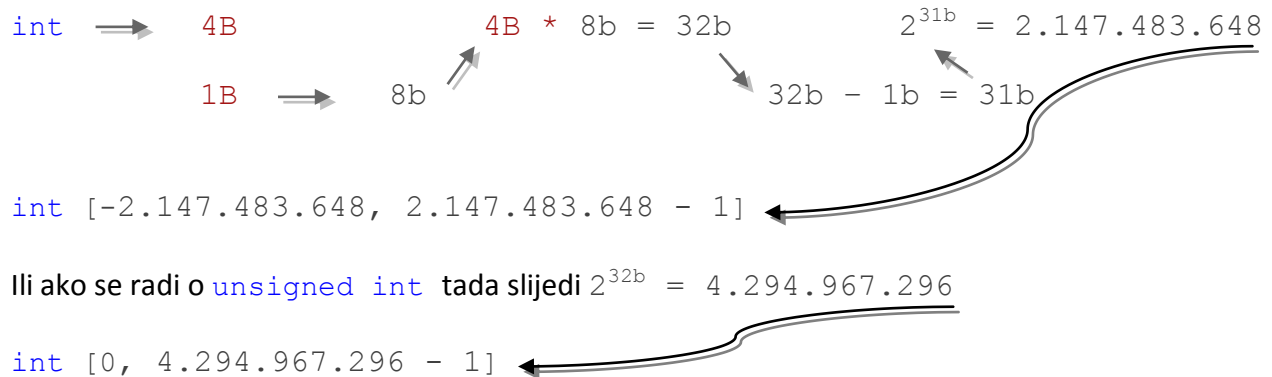
- Veličinu memorijskog prostora koji podatak zauzima
- Domenu podatka, odnosno vrijednosti koje podatak može poprimiti
- Operacije koje se nad podatkom mogu izvršavati

6

Ako se primjerice u varijabli nalazi neki znak odnosno slovo naravno da nad tim tipom nećemo raditi operaciju množenja ili dijeljenja.

Ako pak želimo znati koji raspon vrijednosti možemo spremiti u neki tip podatka, moramo znati dvije stvari. Prvo svaki tip zauzima određeni broj byte-ova u memoriji računala. Tako, primjerice tip `int` zauzima 4B. Osim toga, treba znati sadrži li podatak i negativne brojeve (`signed`) ili ne (`unsigned`).

Primjer 2.



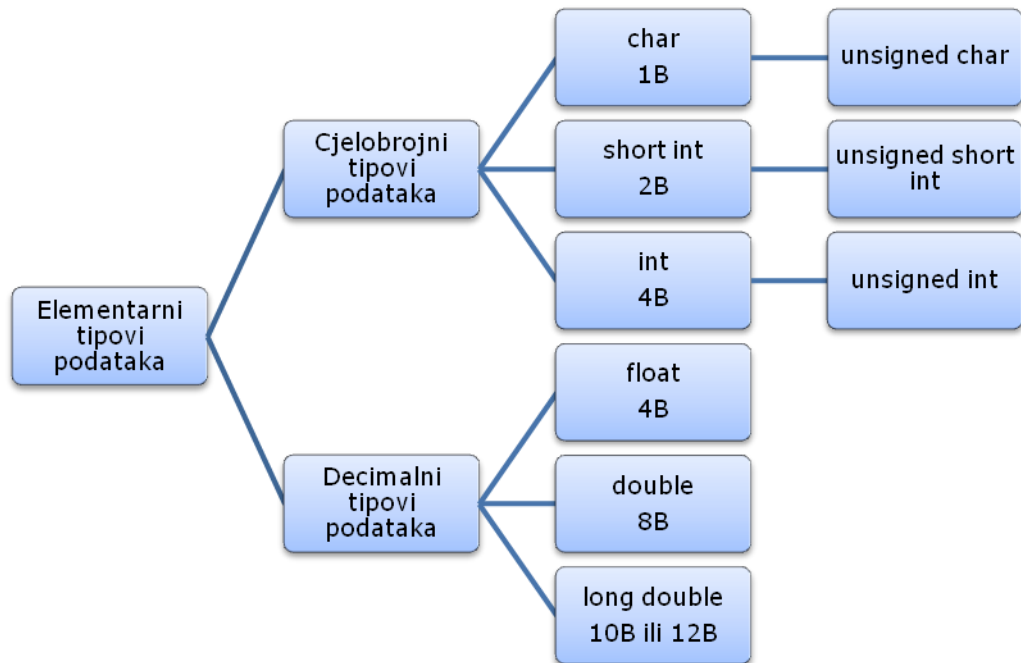
Podaci tipa podatka `int` (tip podatka koji sprema pozitivne ali i negativne brojeve) zauzima 4B. Svaki od tih 4 byte-a ima po 8 bitova, dakle sveukupno je to 32 bita međutim jedan se bit koristi za predznak tako da je to 31 bit. Pošto svaki bit može imati dva stanja (0 i 1) a bitova sveukupno ima 31, bazu binarnog brojevnog sustava (2) potenciramo sa 31 i dobijemo raspon brojeva koji možemo smjestiti u tip podatka `int`.

Međutim, ako se pak radi o tipu podatka `unsigned int` koji sprema samo pozitivne brojeve tada naravno na raspolaganju imamo sva 32 bita jer bit za predznak nije potreban i može se iskoristiti za pohranu brojeva. Stoga pomoću njega možemo prikazati dvostruko više pozitivnih brojeva.

Napomena: negativnih brojeva uvijek ima za jedan više zbog toga što se negativni brojevi radi izbjegavanja istog zapisa brojeva zapisuju preko dvojnog komplementa. (Ponoviti dvojni komplement iz predmeta Informatika 1)

Tip podatka `int` samo je jedan od elementarnih tipova podatka u programskom jeziku C++. Sljedeći grafikon prikazuje potpuni popis elementarnih tipova podatka u programskom jeziku C++, kao i njihove veličine u byte-ovima¹.

¹ Kao priprema za 64-bitna računala u programski jezik C je u posljednje vrijeme dodan tip podatka `long long int` (`int64`), koji zauzima 8B, no ovaj tip podatka još uvijek nije podržan u programskom jeziku C++.



Slika 1: Elementarni tipovi podataka

Kao što se može i vidjeti iz prethodnog grafikona, u programskom jeziku C++ ne postoje **unsigned** decimalni brojevi pa nemojte na to zaboraviti kada budete izrađivali svoje programe.

U sljedećem primjeru ćemo upisati dva pozitivna cijela broja i ispisati njihov cjelobrojni kvocijent, ostatak i decimalni kvocijent.

Primjer 3

Ulaz: Cijeli brojevi A i B

Izlaz: Cjelobrojni kvocijent brojeva A i B, ostatak od dijeljenja i njihov decimalni kvocijent

1. Učitaj brojeve A i B
2. Ispiši cjelobrojni kvocijent i ostatak
3. Ispiši decimalni kvocijent

Ovaj je algoritam krajnje jednostavan pa ćemo ga izravno pretvoriti u C++ kod.

Primjer 3.1

```

#include <iostream>
using namespace std;

int main()
{
    unsigned int A,B;

```



```

cout << "Upisite A: ";
cin >> A;

cout << "Upisite B: ";
cin >> B;


cout << "A / B = " << A/B << " i "
    << A % B << " ostatka. " << endl;

cout << "A / B = " << (double)A/B << endl;

system("pause");

return 0;
}

```


 PRETVARANJE PODATKA U
 VARIJABLI A U PODATAK TIP
 double

Kod aritmetičkih operacija u programskom jeziku C++ rezultat aritmetičke operacije bit onog tipa podataka kojeg su operandi. U našem primjeru to znači da, podijelimo li dva cijela broja, dobit ćemo cijeli broj, odnosno rezultat njihova cjelobrojnog dijeljenja.

Ako su operandi različitih tipova podataka, onda će rezultat uzeti općenitiji tip od dvaju tipova operandi. Sljedeća tablica daje popis tipova podataka operandi i rezultata.

Tabela 2: Tipovi podataka operandi te rezultat

<i>Operandi</i>	Rezultat
char short	short
char int	int
char float	float
char double	double
short int	int
short float	float
short double	double
int float	float
int double	double
float double	double

Dakle, kako bismo postigli da rezultat bude decimalni broj, moramo bar jedan od argumenata pretvoriti u decimalni broj. Pretvaranje vrijednosti u neki drugi tip podataka se u programskom jeziku C++ radi jednostavno tako da se prije vrijednosti u izrazu u okruglim zagradama navede tip podataka u koji ga želimo pretvoriti.

Sljedeći primjer pokazuje neke od mogućnosti formatiranja ispisa.

Upišimo dva pozitivna cijela broja. Ispišimo brojeve u oktalnom i heksadecimalnom sustavu, a nakon toga ih podijelimo i njihov rezultat ispišimo najprije njihov cjelobrojni kvocijent i ostatak od dijeljenja, a nakon toga njihov decimalni kvocijent s točnošću na 10 znamenaka.

Primjer 4

Ulaz: Dva cijela broja

Izlaz: brojevi u oktalnoj i heksadecimalnoj notaciji, njihov cjelobrojni kvocijent i ostatak od dijeljenja te decimalni kvocijent s točnošću od 10 znamenaka

1. Učitaj broj A
2. Učitaj broj B
3. Ispiši brojeve A i B u oktalnom brojevnom sustavu
4. Ispiši brojeve A i B u heksadecimalnom brojevnom sustavu
5. Ispiši cjelobrojni kvocijent brojeva A i B i ostatak od dijeljenja
6. Ispiši decimalni kvocijent brojeva A i B s točnošću na 10 znamenaka

Primjer 4.1

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

int main()
{
    int A,B;

    cout << "Upisite A: ";
    cin >> A;
    cout << "Upisite B: ";
    cin >> B;

    cout << oct << "A_8 = " << A << endl << "B_8 = " << B << endl;
    cout << hex << "A_16 = " << A << endl << "B_16 = " << B << endl;
    cout << dec << "A / B = " << A/B << " i "
    << A % B << " ostatka. " << endl;
    cout << "A / B = " << setprecision(10) << (double)A/B << endl;

    system("pause");

    return 0;
}
```

PRIKAZ CJELOBROJNIH VRIJEDNOSTI U OKTALNOM SUSTAVU

PRIKAZ CJELOBROJNIH VRIJEDNOSTI U OKTALNOM SUSTAVU

PRIKAZ CJELOBROJNIH VRIJEDNOSTI U OKTALNOM SUSTAVU

BROJ ZNAMENAKA U PRIKAZU DECIMALNIH BROJEVA

Prilikom ispisa podataka kroz standardni izlazni tok podataka moguće je formatirati ispis. Tako postoji mogućnost da se ispis cjelobrojnih vrijednosti prebaci u oktalni i heksadecimalni sustav. To se radi tako da se unutar naredbe za ispis podataka kroz standardni izlazni tok podataka navede klauzula `oct`, odnosno `hex`. Treba napomenuti da će ovakva definicija vrijediti, ne samo u naredbi u kojoj je navedena, već i u svim sljedećim ispisima, dok god se ne definira drugačije.

Stoga je nakon što ispišemo vrijednosti u heksadecimalnom brojevnom sustavu ispis cjelobrojnih vrijednosti vratiti u decimalni sustav. To se radi navođenjem klauzule `dec`.

Na kraju, postoji također i klauzula kojom se definira broj znamenaka u ispisu decimalnih brojeva. Ta se klauzula naziva `setprecision` i u njoj se u zagradama navodi broj znamenaka koje se prikazuju. Slično prethodno navedenim klauzulama, definicija broja znamenaka u ispisu decimalnih brojeva vrijedit će i u sljedećim ispisima, sve dok se ne definira drugačije.

Zadaci za vježbu

1. Što je to varijabla?
2. Čemu služi tip podataka?
3. Koliki je raspon/vrijednosti za `unsigned short`?
4. Modificirajte Primjer 1.2. tako da se za vrijednost varijable A unosi decimalni tip podatka.

Programski primjeri

Zadatak 1.

Izradite program u kojem:

1. se na početku pokretanja ispisuje proizvoljna pozdravna poruka.
2. korisnik unosi cjelobrojnu vrijednost koja se pohranjuje u odgovarajuću varijablu.
3. se unesena vrijednost smanjuje za 1 i ispisuje na zaslone.
4. korisnik unosi drugu cjelobrojnu vrijednost koja se pohranjuje u odgovarajuću varijablu te se na zaslon ispisuju obje unesene vrijednosti onim redoslijedom kojim su unesene.
5. se druga unesena vrijednost povećava za 2 i rezultat se ispisuje na zaslon.

Rješenje:

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

int main()
{
    // 1. Proizvoljna pozdravna poruka
    cout << "Dobrodošli!" << endl;
```

```
// 2. Unos cjelobrojne vrijednosti
int PrviOperand;

cin >> PrviOperand;

// 3. Smanjivanje za jedan i ispis na zaslon
PrviOperand = PrviOperand - 1;

cout << PrviOperand << endl;

// 4. Unos druge vrijednosti te ispis na zaslon
//   redoslijedom unosa
int DrugiOperand;

cin >> DrugiOperand;

cout << PrviOperand;
cout << DrugiOperand << endl;

// 5. Povećanje druge vrijednosti za 2 i ispis na zaslon
DrugiOperand = DrugiOperand + 2;

cout << DrugiOperand << endl;

// ... zaustavljanje programa
system("pause");

return 0;

}
```

A screenshot of a Windows command prompt window. The text inside the window is: "Dobrodošli!" followed by four lines of numbers: "4", "3", "5", and "35". Below these is the number "7" and then the text "Press any key to continue . . . _". The window has a standard Windows title bar and a scroll bar on the right.

Zadatak 2.

Izradite program u kojem:

1. se na početku unosi cjelobrojna vrijednost i sprema u varijablu.
2. se nakon prethodne točke unosi druga decimalna vrijednost ali u varijablu tipa `int`.
3. se druga unesena vrijednost ispisuje na zaslon.
4. se unesene vrijednosti povećaju za 1.
5. se unesene vrijednosti ispišu na zaslon obrnutim redoslijedom od redoslijeda unosa.

Rješenje:

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

int main()
{
    // 1. Unos cjelobrojne vrijednosti
    int PrviOperand;

    cin >> PrviOperand;

    // 2. Unos druge decimalne vrijednosti
    int DrugiOperand;

    cin >> DrugiOperand;

    // 3. Ispis druge decimalne vrijednosti na zaslon
    cout << DrugiOperand << endl;

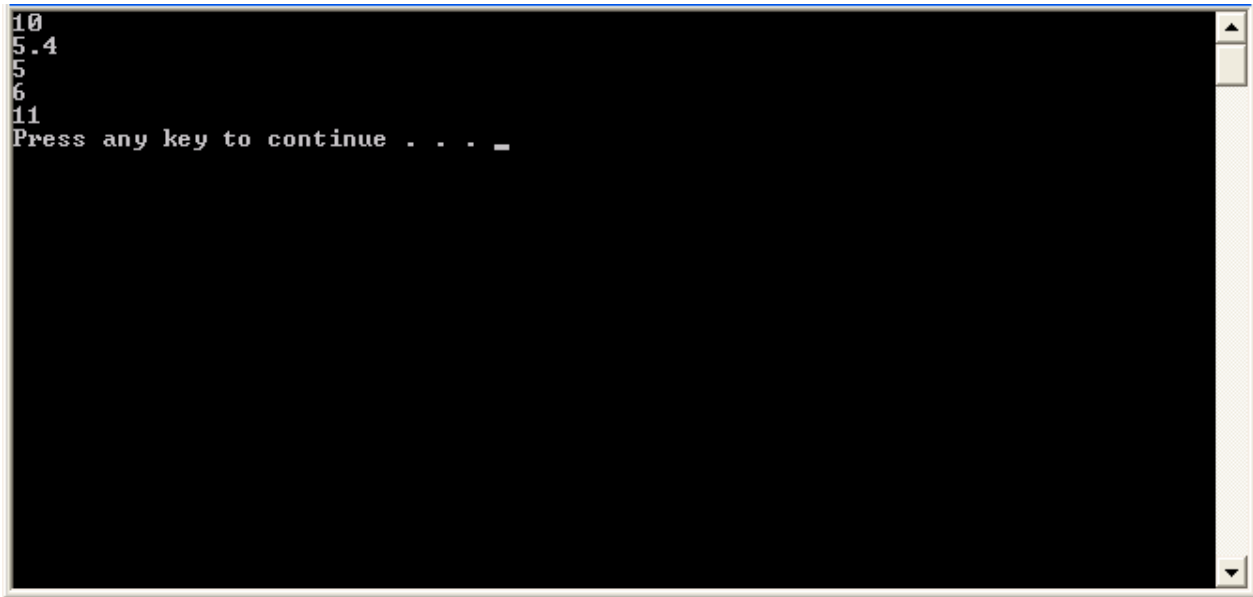
    // 4. Uvecanje unesenih vrijednosti za 1
    PrviOperand = PrviOperand + 1;
    DrugiOperand = DrugiOperand + 1;

    // 5. Ispis na zaslon obrnutim redoslijedom
    //     od redoslijeda unosa
    cout << DrugiOperand << endl;

    cout << PrviOperand << endl;

    // ... zaustavljanje programa
    system("pause");
}
```

```
    return 0;  
}
```



```
10  
5.4  
5  
6  
11  
Press any key to continue . . . _
```

Slika 3: Primjer izvođenja prethodnog zadatka

Zadatak 3.

Izradite program u kojem:

1. korisnik unosi dvije brojčane vrijednosti (brojevi mogu biti i decimalni, neka jedan operand bude tipa `int` a drugi `float`).
2. se uneseni brojevi zbroje ispišu na zaslon.
3. se uneseni brojevi pomnože i ispišu na zaslon.
4. se prvi uneseni broj oduzme od drugog i rezultat se ispiše na zaslon.
5. se prvi uneseni broj podijeli sa drugim i rezultat se ispiše na zaslon.

Rješenje:

14

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

```
// 1. Unos brojčanih vrijednosti
int PrviOperand;
float DrugiOperand;

cin >> PrviOperand;
cin >> DrugiOperand;

// 2. Ispis unesenih brojeva na zaslon
cout << PrviOperand << " i " << DrugiOperand << endl;

// 3. Umnozak brojeva te ispis
cout << PrviOperand * DrugiOperand << endl;

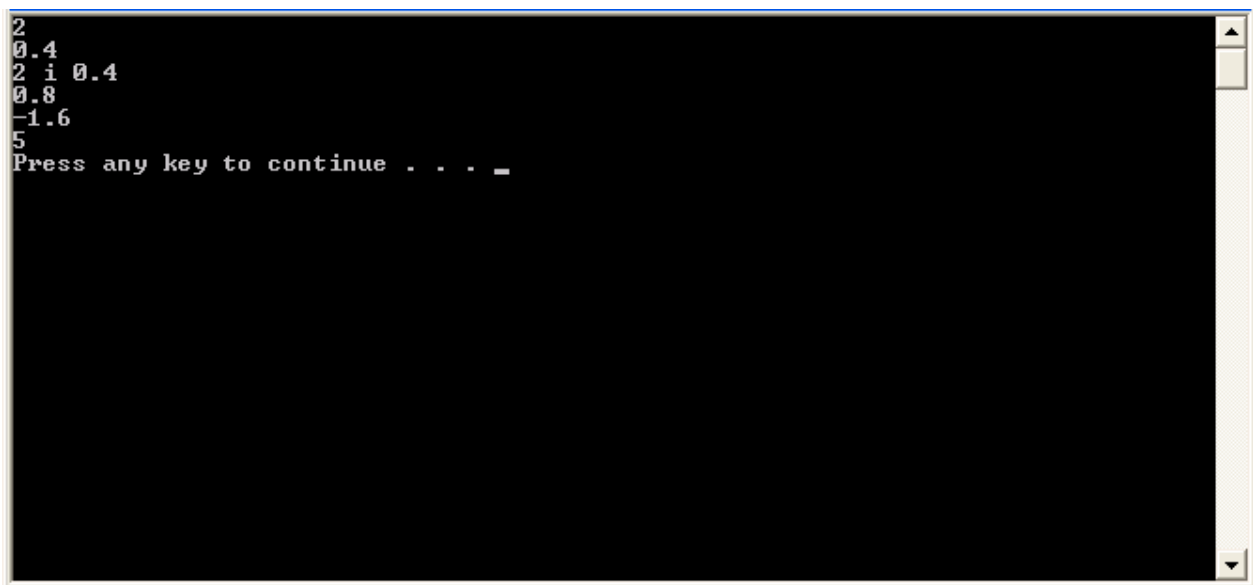
// 4. Razlika drugog i prvog operanda
cout << DrugiOperand - PrviOperand << endl;

// 5. Kvocijent prvog i drugog operanda
cout << PrviOperand / DrugiOperand << endl;

// ... zaustavljanje programa
system("pause");

return 0;

}
```



```
2
0.4
2 i 0.4
0.8
-1.6
5
Press any key to continue . . . _
```

Slika 4: Primjer izvođenja prethodnog zadatka

Programski primjer za laboratorijske vježbeZadatak 1.

Izradite program u kojem:

1. se na početku unose četiri brojčane vrijednosti (brojevi mogu biti i decimalni) te se izračuna i ispiše kvocijent prva dva unesena broja i umnožak druga dva unesena broja.
2. se prethodni rezultat ispiše u oktalnom brojevnom sustavu.
3. se prvi rezultat iz točke 1 pretvori u cjelobrojni zapis i ispiše na ekran.
4. se unese još jedna cjelobrojna vrijednost.
5. se prethodne dvije cjelobrojne vrijednosti (iz točaka 3 i 4) modularno podjele i rezultat se istovremeno ispiše u heksadecimalnom i decimalnom brojevnom sustavu.

