PROGRAMIRANJE 1
UVOD U PROGRAMSKI JEZIK C

LEKSIČKE KONVENCIJE

- U upotrebi su velika i mala slova, cifre i specijalni simboli iz ASCII skupa
- Programski jezik C razlikuje velika i mala slova!
- Komentari se navode između sekvenci /* i */. Mogu se prostirati u više linija. Ne mogu biti ugnježdeni.
- **int x, X;** /*To su dve razlicite promenljive!!!*/

TOKENI

- Postoji šest vrsta tokena:
  - identifikatori
  - ključne reči
  - operatori
  - separatori
  - stringovi
  - konstante
- Tokeni se razdvajaju belinama, tabulatorima i novim redovima.

IDENTIFIKATORI

- **Identifikatori** se sastoje iz slova, cifara i znaka _ pri čemu prvi karakter nije cifra.
- Koriste se za imena promenljivih, tipova, funkcija, itd.

KLJUČNE REČI

- **Ključne reči** su rezervisane reči koje imaju posebnu ulogu, i ne mogu se koristiti kao identifikatori.
- Ključne reči se koriste za:
  - definisanje jezičkih konstrukcija (*if*, *while*, *for*),
  - imena tipova (*int*, *float*, *char*), itd.

PROGRAM U C-U

- Izvorni program (source code) u C-u je običan tekstualni fajl, kreiran u bilo kom editoru teksta (npr. Notepad).
- Program u C-u sastoji se iz definicija funkcija.
- Funkcija koja mora da postoji u svakom programu je main(). Izvršavanje programa se svodi na izvršavanje tela ove funkcije.
- Telo funkcije se navodi iza zaglavlja funkcije main(), između vitičastih zagrada { i }.

- U telu funkcije se na početku navode deklaracije pomenljivih, nakon čega sledi proizvoljan niz naredbi .

```
main() /* zaglavlje funkcije main */
{ /* ovde pocinje telo */
/* deklaracije promenljivih */
/* naredbe */
} /* kraj tela */
```

PRVI PRIMERI U C-U

- Napisati program koji na standardnom izlazu štampa ”Zdravo!”.

```
#include <stdio.h>
main()
{
printf("Zdravo, svete!\n");
}
```

- Izlaz iz programa:

```
Zdravo, svete!
```

- Šta je izlaz iz sledećeg programa?

```
#include <stdio.h>
main()
{
printf("Zdravo, ");
printf(„svete!");
printf("\n");
}
```

PROMENLJIVE – DEKLARACIJA I INICIJALIZACIJA

- Sve promenljive u C-u se moraju deklarisati. Time se za promenljivu u memoriji rezerviše potreban prostor, a ostatak programa postaje svestan postojanja promenljive i njenog tipa.
- Deklaracija se sastoji iz imena tipa za kojim slede imena promenljivih (identifikatori) koje se deklarišu, i koja su razdvojene zarezima.
- Deklaracija se završava simbolom ';'.
- Svako ime promenljive u deklaraciji može biti praćeno inicijalizatorom koji se sastoji iz karaktera '=' za kojim sledi inicijalna vrednost.

- Dakle, da bi se promenljiva mogla upotrebljavati u programu ona se mora na početku programa deklarisati!
- Prilikom deklaracije može se izvršiti i početna inicijalizacija.

```
int broj; /* Deklaracija celog broja */
int vrednost=5; /* Deklaracija i
                   inicijalizacija celog broja */
```

- Postoji i kvalifikator const koji može biti dodeljen deklaraciji bilo koje promenljive da bi označio da se ona neće menjati

```
const double e=2.71828182845905 ;
```

- Uvođenje promenljivih u program.

```
#include <stdio.h>
main()
{
/* Deklaracija vise promenljivih istog tipa */
        int rez,pom1,pom2;
        pom1=20;
        pom2=15;
        rez=pom1-pom2;
 /* Ispisivanje rezultata */
        printf("Rezultat je %d - %d = %d\n",
pom1, pom2, rez);
}
```

OSNOVNI TIPOVI PODATAKA U C-U

Tip	Opis	Uobičajena veličina
char	Mali ceo broj (ASCII kod karaktera)	1 bajt
int	Ceo broj	4 bajta
float	Realan broj	4 bajta
double	Realan broj	8 bajtova

OSNOVNI TIPOVI PODATAKA U C-U

- Tipovi iz tabele se mogu modifikovati ključnim rečima short, long, signed i unsigned.
- Na **int** se mogu primeniti **short** i **long** (u tom slučaju ključna reč int nije obavezna).
- Na **double** se može primeniti **long.**
- Na **float** se ne može primeniti nijedan modifikator.

- Modifikatori **unsigned** i **signed** se mogu primeniti na celobrojne tipove (char, int, short i long).
- Veličine tipova su platformski zavisne.
- Obavezna relacija za celobrojne tipove:

  short ≤ int ≤ long.

- Tip char zauzima jedan bajt ali u zavisnosti od sistema ovaj tip moze da se odnosi na oznacene ili na neoznacene brojeve.
- Zato postoje kvalikatori signed i unsigned koji preciziraju da li se misli na oznacene ili neoznacene cele brojeve.
- Npr.

signed char: -128 do 127

dok je

unsigned char: od 0 do 255.

- Velicina za int je razlicita u zavisnosti od sistema i moze biti 2 ili 4 bajta. Medutim, ako je promenljiva tipa short int onda ona sigurno zauzima samo dva bajta a to znaci da u nju mogu da stanu celobrojne vrednosti iz intervala -32 768 do +32 767, odnosno mogu se koristiti sledeci sinonimi

short <=> short int
<=> signed short int
<=> -32 768 do 32 767

- Ukoliko se koristi unsigned short int onda je interval

  unsigned short int <=> 0 do 65 535

- Za realne tipove podataka koriste se float, double i long double.
- Njihove velicine mogu da zavise od sistema.
- Na primer:
- float (4 bajta)

float minimalna pozitivna vrednost

1.175494351e-38

float maximalna pozitivna vrednost

3.402823466e+38

- double (8 bajta)
- double minimalna pozitivna vrednost
  2.2250738585072014e-308
- double maximalna pozitivna vrednost
  1.7976931348623158e+308
- long double (10 bajta)
- long double minimalna pozitivna
  3.36210314311209350 63e-4932
- long double maximalna pozitivna
  1.18973149535723176 5e+4932

KONSTANTE

- **Karakterske konstante** se navode između jednostrukih navodnika. One su tipa **char**.
- **Celobrojne konstante** mogu biti dekadne, oktalne i heksadekadne. Sve ove konstante su tipa **int**.
  - **Oktalne** počinju nulom
  - **Heksadekadne** počinju sekvencom 0x ili 0X.
- **Sufiksi u ili U** daju **neoznačenu konstantu**.
- **Sufiksi l ili L** daju konstantu tipa **long**.

- Konstante realnih brojeva sadrže decimalnu tačku(123.4) ili eksponent(1e-2) ili i jedno i drugo. Njihov tip je double osim ako:
- se završavaju sufiksom **f ili F** – tada su tipa **float**.
- se završavaju sufiksom **l ili L –** tada su tipa **long double**.
- Znakovna konstanta '\0' predstavlja znak cija je vrednost nula, treba ga razlikovati od '0' koja je znak cija je vrednost 48.

PRIMERI:

- 'a'
- 'ab'
- '\n'
- 34
- 034
- 093
- 0xa
- 2
- 321
- 0x12U
- 034ul
- 12Lu
- 4.2
- 2e-3
- 1.e-3
- .5e-1f
- 5.4e-3.4
- 5.4e-3L

OPERATORI

- U C-u postoji veliki broj operatora. Mogu biti **unarni** i **binarni**.
- **Unarni operatori** mogu biti prefiksni i sufiksni.
- **Binarni operatori** su po pravilu infiksni.
- Operatori imaju svoj prioritet i asocijativnost.

IZRAZI

- Kombinovanjem promenljivih, konstanti i operatora dobijamo **izraze**.
- Svaki izraz ima svoj **tip** i svoju **vrednost**.

TIP IZRAZA I KONVERZIJA

- Tip izraza zavisi od tipova podizraza koji ga čine, kao i od operatora kojim se ti podizrazi povezuju.
- Ako su **operandi** neodgovarajućeg tipa, tada se implicitno konvertuju u odgovarajući tip, ako je to moguće. Koja će se konverzija izvršiti zavisi od operatora.

PRIMER - KONVERZIJA

- Naredba dodele:

```
int i=5;
float f=2.3;
f=i; /* f ce imati vrednost 5.0*/
```

- obrnuto:

```
int i=5;
float f=2.3;
i=f; /* i ce imati vrednost 2*/
```

KONVERZIJE TIPOVA

- Tip izraza se može eksplicitno promeniti tzv. **cast** operatorom (ispred izraza se u zagradi navede ime tipa u koji želimo da konvertujemo izraz).

  (tip)<izraz>

- float x;

  x=2.3+4.2; /* x ce imati vrednost 6.5 */

  x=(int)2.3+(int)4.2; /*x ce imati vrednost 6 */

  x=(int)2.3*4.5; /* x ce imati vrednost 9.0 jer zbog prioriteta operatora konverzije prvo ce biti izvrsena konverzija broja 2.3 u 2 pa tek onda izvrseno mnozenje. */

- x=(int)(2.3*4.5) /* x ce imati vrednost 10.0 */

PRIMER

- *Kako izbeći celobrojno deljenje*
- **int a,b;**
  float c;
  a = 5;
  b = 2;
  c = a/b; /* Celobrojno deljenje, c=2*/
  c = (1.0*a)/b; /*Implicitna konverzija: 1.0*a je realan broj pa prilikom deljenja sa b dobija se realan rezultat c=2.5*/
  c = (0.0+a)/b; /* Implicitna konverzija: (0.0+a) je realan broj pa prilikom deljenja sa b dobija se realan rezultat c=2.5*/
  c = (float)a/(float)b; /* Eksplicitna konverzija*/

ARITMETIČKI OPERATORI

- Operatori **+**,**-**,*****,**/**,**%** nazivaju se aritmetički operatori.
- Ovi operatori se primenjuju na celobrojne i realne tipove.
- Operator **/** primenjen na cele brojeve daje celobrojni količnik.
- Operator **%** se primenjuje samo na cele brojeve i daje ostatak pri deljenju.

- Rezultat aritmetičkih operacija je izraz istog tipa kao i operandi.
- Ako operandi nisu istog tipa tada se vrši implicitna konverzija užeg u širi tip.

RELACIONI OPERATORI

- Operatori ==,!=,<,>,<=,>= nazivaju se relacioni operatori.
- Ovi operatori se primenjuju na celobrojne i realne tipove.
- Rezultat relacionih operacija je izraz celobrojnog tipa sa vrednošću 1 ako je relacija tačna, 0 u suprotnom.
- Ako operandi nisu istog tipa tada se vrši implicitna konverzija užeg u širi tip.

LOGIČKI OPERATORI

- Logički operatori su:
- **!** – unarna negacija
- **&&** – logičko i
- **||** – logičko ili.

LOGIČKI OPERATORI

- U C-u ne postoji logički tip.
- Svaki izraz koji se može porediti na jednakost sa nulom se u C-u smatra logičkim izrazom.
- Ako je vrednost izraza različita od 0, tada se on smatra **logički tačnim**. Izraz jednak nuli se smatra **logički netačnim**.
- Rezultat logičkih operacija je izraz celobrojnog tipa, jednak 0 ako je rezultat logičke operacije netačno, odnosno 1, ako je rezulatat logičke operacije tačno.

PRIMERI - LOGIČKI OPERATORI

- 5 && 4 – vrednost je tačno,
- 10 || 0 – vrednost je tačno,
- 0 && 5 – vrednost je 0,
- !1 – vrednost je 0,
- !9 – vrednost je 0,
- !0 – vrednost je 1,
- !(2>3) – vrednost je 1.

OPERATORI SA BOČNIM EFEKTIMA

- Pojava da se prilikom izračunavanja nekog izraza menja vrednost neke promenljive naziva se **bočni efekat** (eng. side effect).
- Operatori koji imaju bočni efekat su operatori dodele i operatori uvećanja i umanjenja.
- Naredba dodele se u C-u predstavlja izrazom dodele, a njeno izvršavanje se zasniva na bočnim efektima operatora dodele.

OPERATORI DODELE

- Operator proste dodele je =. Ne treba ga mešati sa relacionim operatorom ==.
- Levi operand ovog operatora je leva vrednost (ime promenljive). Desni operand je proizvoljni izraz.
- Najpre se izračunava izraz na desnoj strani. Njegova vrednost se zatim po potrebi konvertuje u tip promenljive na levoj strani. Nakon toga se dobijena vrednost upisuje u memorijsku lokaciju koja je rezervisana za čuvanje te promenljive.

- Tip izraza dodele je tip promenljive na levoj strani. Vrednost izraza dodele je vrednost koja je dodeljena promenljivoj.
- Operatori složene dodele su **+=,-=,*=,/=,%=**.
- Izraz oblika **a+=E** je ekvivalentan izrazu a=a+(E).
- Slično je sa ostalim operatorima.

OPERATORI UMANJENJA I UVEĆANJA

- Unarni operatori **++** i **--** nazivaju se operatori **uvećanja** (inkrementacije) i **umanjenja** (dekrementacije) respektivno.
- Mogu biti prefiksni i sufiksni.
- U oba slučaja vrši se uvećanje promenljive za 1 ali izraz **++n** uvećava promenljivu n *pre* nego što se njena vrednost koristi, dok **n++** uvećava n *nakon što se njena vrednost koristi*. Tako se x=++n; razlikuje od x=n++;.
- Slično za operator --, s tim što je u pitanju umanjenje za jedan.

PRIMERI

- a=5; nakon a++, a=6
- a=5; nakon ++a, a=6
- a=5; nakon b=a++, a=6, b=5
- a=5; nakon b=++a, a=6, b=6

PRIORITET I ASOCIJATIVNOST OPERATORA

- Unarni operatori imaju viši prioritet od binarnih.
- Aritmetički operatori su višeg prioriteta od relacionih, a ovi višeg od logičkih.
- Operatori dodele su najnižeg prioriteta.
- Ako dva operatora imaju isti prioritet, onda se u obzir uzima asocijativnost, koja može biti s leva na desno, ili s desna na levo.
- Prioritet operatora se može promeniti korišćenjem zagrada ().

PRIMERI

- unarni operatori su višeg prioriteta od binarnih: -1+2=1
- aritmetički operatori su višeg prioriteta od relacionih: 3+5<6=0

  3+(5<6)=4
- relacioni operatori su višeg prioriteta od logičkih: 0<-1||0=0

  0<(-1||0)=1

- Binarni operatori su leve asocijativnosti, osim operatora dodele koji imaju desnu asocijativnost. Unarni operatori su mahom desne asocijativnosti, ali ima izuzetaka.

x=y==z je isto što i x=(y==z)

-a*b+c je isto što i ((-a)*b)+c

a=b=c=d je isto što i (a=(b=(c=d)))

FUNKCIJE ULAZA I IZLAZA

- Ulaz i izlaz ostvaruju se posredstvom funkcija koje su definisane u standardnoj biblioteci stdio.h.
- Ove funkcije su obične C funkcije, koje se služe direktno servisima operativnog sistema prilikom svog rada.

- Za korišćenje ovih funkcija neophodno je uključiti zaglavlje stdio.h navođenjem direktive #include<stdio.h> pre definicije funkcije main().
- Ovo zaglavlje je obilan tekstualni fajl u kome su navedene deklaracije funkcija ulaza i izlaza.
- Direktiva #include na mestu poziva uključuje kompletan sadržaj fajla koji je naveden, čime funkcije i podaci deklarisani u njemu postaju dostupni funkciji main().

FUNKCIJA PRINTF()

- Ovom funkcijom se ispisuje poruka zadata format stringom na standardni izlaz.
- Eventualni konverzioni specifikatori se zamenjuju vrednostima izraza koji u tom slučaju slede nakon format stringa, kao argumenti funkcije printf(), razdvojeni zarezima i u onom poretku u kome su odgovarajući konverzioni specifikatori navedeni.
- Tipovi izraza moraju biti u skladu sa tipovima koje određuju konverzioni specifikatori.

KONVERZIONI SPECIFIKATORI

Specifikator	Tip	Napomena
%d	int	Opciono označeni dekadni broj
%f	float	Realan broj sa opcionim eksponentom
%lf	double	Realan broj sa opcionim eksponentom
%Lf	long double	Realan broj sa opcionim eksponentom
%hd	short	Opciono označeni dekadni broj
%ld	long	Opciono označeni dekadni broj
%c	char	Ispis karaktera iz ASCII skupa

- Funkcija printf je bibliotečka funkcija koja prikazuje izlazne podatke u određenom formatu.
- Primer korišćenja funkcije printf je:

```
printf("%d\t%d\n", broj1, broj2);
```

- Prvi argument ove funkcije je uvek između ” ” i određuje format u kome će se podaci ispisati na izlaz. Ova funkcija vraća kao vrednosti broj upisanih znakova na izlazu.

- Sekvenca \n u okviru prvog argumenta funkcije printf je C oznaka za prelazak u novi red, \t je oznaka za tabulator, dok %d označava da će na tom mestu biti ispisana celobrojna vrednost argumenta koji je sa njim u paru. Svaka % konstrukcija je u paru sa odgovarajućim argumentom koji sledi.
- %% koristi se za ispis znaka %
- \\ koristi se za ispis znaka \
- \” koristi se za ispis znaka ”

- Postoji mogućnost da se precizira i širina polja u kome će se ispisati odgovarajuće vrednosti.
- Na primer, koristimo %3c za štampanje karaktera na tri pozicije poravnato zdesna.
- Koristimo %3d za štampanje broja na tri pozicije ili %6d za štampanje broja na 6 pozicija.

- Važi:
- %f — štampaj kao realan broj
- %6f — štampaj kao realan broj širok najviše 6 znakova
- %.2f — štampaj kao realan broj sa dve decimale
- %6.2f — štampaj kao realan broj širok najviše 6 znakova pri čemu su 2 iza decimalne tačke.
- Da bi se izvršilo levo poravnanje, između % i odgovarajućeg karaktera dodaje se znak -.

PRIMER PRINTF()

- #include <stdio.h>

```
main()
{
printf("Slova:\n%3c\n%5c\n", 'z' , 'Z');
}
```

- Izlaz iz programa:

```
Slova:
  z
    Z
```

FUNKCIJA SCANF()

- Ovom funkcijom se učitavaju podaci sa standardnog ulaza.
- Prvi argument je format string u kome se navode konverzioni specifikatori kojima se definiše tip podatka koji se očekuje.
- Nakon format stringa slede adrese promenljivih, razdvojene zarezima, u koje treba upisati vrednosti učitane sa ulaza. Adresa promenljive a navodi se sa &a.

- Adrese se navode u onom poretku u kom su odgovarajući konverzioni specifikatori navedeni u format stringu.
- Tipovi promenljivih moraju biti u skladu sa tipovima koje određuju konverzioni specifikatori.
- Na primer,

```
scanf("%d %d", &broj1, &broj2);
```

- Ova funkcija čita sa ulaza dva cela broja i smešta ih na adresu promenljivih broj1 i broj2, redom.
- Kao rezultat, ova funkcija vraća broj uspešno dodeljenih ulaznih vrednosti.
- Naredni poziv funkcije scanf nastavlja čitanje neposredno iza poslednjeg znaka koji je već pročitan.

PRIMER - SCANF()

- Program prikazuje unos celog broja koristeći funkciju scanf("%d", &x).

```
#include <stdio.h>
main()
{
int x;
printf("Unesi ceo broj : ");
/* Obratiti paznju na znak &
pre imena promenljive u funkciji scanf. */
scanf("%d",&x);
/* U funkciji printf nije
potrebno stavljati &. */
printf("Uneli ste broj %d\n", x);
}
```

USLOVNI IZRAZ

izraz1 ? izraz2 : izraz3

- z = (a<b)? a : b; /*z=min(a,b)*/
- max = (a>b)? a : b;