V – Unos i prikaz podataka

- Programi čitaju i ispisuju podatke sa spoljašnjih uređaja
- Postoji prolazni U/I (tastatura i monitor) i trajni U/I podataka (fajlovi)
- **printf** i **scanf** pišu/čitaju na monitor ili sa tastature-prolazni ulaz/izlaz
- Zbog mnogih razlika ulazne i izlazne mogućnosti nisu deo C-jezika
- Ulaz i izlaz (prikaz) podataka ostvaruje se pomoću funkcija koje su definisane i koje se nalaze u standardnoj biblioteci **stdio.h**.
- Ove funkcije su obične C funkcije, koje se služe direktno servisima operativnog sistema da bi obavile svoj zadatak tj. namenu.
- Za korišćenje ovih funkcija neophodno je uključiti zaglavlje **stdio.h** navođenjem direktive #include<stdio.h> pre definicije funkcije main()
- Ovo zaglavlje je običan 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().
- Obradićemo šest funkcija iz standardne ulazno-izlazne biblioteke i to:

  getchar, putchar, gets, puts, scanf i **printf**.

V - Funkcije getchar i putchar

- Funkcija **getchar** čita jedan znak sa standardnog ulaza-obično tastature
- Funkcija nema argumenata pa je sintaksa poziva:

  c_var = getchar(); ili **int getchar(void);**
- Funkcija **putchar** šalje jedan znak na standardni izlaz - obično monitor
- Funkcija ima jedan argument (znak koji se ispisuje) i vraća celobrojnu vrednost (int vrednost) koja se obično ignoriše.
- Najčešće poziv funkcije ima oblik

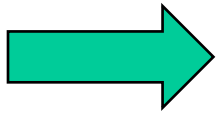
  putchar(c_var); ili **int putchar(int c);**
- Kada funkcija **getchar** naiđe na kraj ulaznih podataka vraća vrednost **EOF** (skraćenica od ***End of File***).
- EOF je simbolička konstanta koja se nalazi u <stdio.h> koja označava kraj datoteke i kraj ulaznih podataka (ulaz je tretiran kao datoteka).
- Konstanta EOF mora da se razlikuje od znakova koje računar koristi.
- Zato funkcija **getchar** ne vraća vrednost tipa char već vrednost tipa int što daje dovoljno prostora za kodiranje konstante EOF.
- Isto tako **putchar** uzima vrednost tipa int i vraća vrednost tipa int.
- Vraćena vrednost je znak koji je ispisan ili EOF ako nije uspeo ispis

V - Funkcije getchar() i putchar()

Primer: *Napisati program koji svaki uneti znak sa ulaza kopira na izlaz pri čemu svaki uneti karakter pretvara u velika slova na izlazu.*

Rešenje:

```
#include <stdio.h>
#include <ctype.h>
/* kopiranje ulaza na izlaz */
int main(void)
{
int c;
c=getchar();
while(c!=EOF)
   {
      putchar(toupper(c));
      c=getchar();
   }
return 0;
}
```



```
#include <stdio.h>
#include <ctype.h>
/* kopiranje ulaza na izlaz */
int main(void)
{
int c;
while((c=getchar())!=EOF)
putchar(toupper(c));
}
```

V - Funkcije datoteke ctype.h

- Datoteka zaglavlja <ctype.h> sadrži deklaraciju niza funkcija koje služe za testiranje znakova.
- Svaka od tih funkcija uzima jedan argument tipa int koji treba biti znak ili EOF i vraća vrednost tipa int koja je različita od nule (istina) ako je uslov ispunjen ili nula ako nije. Neke od funkcija su sledeće:

isalnum()	Alfanumericki znak	**isalpha()**	Alfabetski znak
isctrln()	Kontrolni znak	**isdigit()**	Broj
isgraph()	Printabilni znak(bez ' ‘)	**islower()**	Malo slovo
isprint()	Printabilni znak	**ispunct()**	Znak interpunkcije
isspace()	Razmak	**isupper()**	Veliko slovo

- Pod razmakom smatramo: blanko, novi red, formfeed, carriage return, tabulator i vertikalni tabulator (' ', 'nn', 'nf', 'nr‘, 'nt', 'nv').
- Dve funkcije omogućavaju konverziju velikih slova u mala i obrnuto a da pri tome ostale znakove ne diraju su:

int tolower(int c) - Veliko slovo u malo

int toupper(int c) - Malo slovo u veliko

V - Funkcije gets i puts

char *gets(char *s); **int puts(const char *s);**

- F-je **gets** i **puts** služe čitanju i pisanju znakovnih nizova (stringova)
- Funkcija **gets** čita niz znakova sa standardnog ulaza (tastature), a funkcija **puts** ispisuje niz znakova na standardni izlaz (ekran).
- Funkcija **gets** uzima kao argument znakovni niz u kome će biti učitan niz znakova s ulaza.
- Znakovi sa ulaza učitavaju se sve dok se ne naiđe na kraj linije ('**nn**') koji se zamenjuje znakom '**n0**'.
- F-a vraća pokazivač na char koji pokazuje na učitani znakovni niz ili NULL ako se došlo do kraja ulaznih podataka ili se javila greška.
- Simbolička konstanta NULL uneta je u **<stdio.h>** i njen iznos je 0.
- To je jedina cjelobrojna vrednost koja se može pridružiti pokazivaču
- Funkcija **puts** uzima kao argument znakovni niz koji će biti ispisan na standardnom izlazu.
- F-ja vraća broj ispisanih znakova ako je ispis uspeo a EOF ako nije.
- Pre ispisa **puts** dodaje znak '**nn**' na kraju znakovnog niza.

V - Funkcije gets() i puts()

Primer: *Napisati pogram koji kopira ulaz na izlaz ali liniju po liniju.*

Rešenje:

```
#include <stdio.h>
/* kopiranje ulaza na izlaz */
int main(void)
{
char red[128];
while(gets(red)!=NULL)
puts(red);
}
```

- Unutar testiranja while petlje gets će pročitati ulaznu liniju i vratiti pokazivač različit od NULL ako ulaz nije prazan i ako nema greške.
- U tom slučaju izvršava se komanda puts(red) koja ispisuje učitanu liniju, a povratna vrednost funkcije puts se zanemaruje.
- Osnovni nedostatak funkcije gets je u tome što nije moguće odrediti maksimalni broj znakova koji će biti učitan.
- Ukoliko je broj znakova na ulazu veći od dimenzije polja koje je definisano (argument funkcije gets) doći će do greške.
- Zato je bolje umesto funkcije **gets** koristiti funkciju **fgets**

V - Funkcija unosa Scanf()

- Podaci koje scanf čita dolaze sa standardnog ulaza - tipično tastature.
- Opšta forma funkcije je

  scanf(kontrolni_string, arg_1, arg_2, ... ,arg_n)

  gde je kontrolni string konstantni znakovni niz koji sadrži informacije o vrednostima koje se učitavaju u argumente arg 1,. . . ,arg n.
- Kontrolni znakovni niz (string) je konstantan znakovni niz koji se sastoji od individualnih grupa znakova konverzije pri čemu je svakom argumentu pridružena jedna grupa.
- Svaka grupa znakova konverzije započinje znakom (**%**) koji sledi znak konverzije koji upućuje na tip podatka koji se učitava (%c ili %d itd.)

Primer: **%[<w>] [h | l | L] <tip konverzije>**

gde je **[<w>]** opcioni ceo broj, **[h | l | L]** opciona oznaka formata i **<tip konverzije>** obavezan tip konverzije: **d, u, o, x, i, f, e, g, c** ili **s**.

- Ako se unosi više podataka oni moraju biti odvojeni blanko znacima što u sebi uključuje i prelaz u novi red, koji se računa kao blanko znak
- Numerički podaci na ulazu moraju imati isti oblik kao i numeričke konstante koje su definisane znakom konverzije.

V - Funkcija unosa Scanf()

- Unutar kontrolnog niza znakova, grupe kontrolnih znakova mogu se nastavljati jedna na drugu bez razmaka ili mogu biti odvojene blankom
- Blanko znaci će u ulaznim podacima biti učitani i ignorisani.
- Argumenti funkcije scanf mogu biti samo pokazivači na promenljive.
- Ukoliko podatak treba učitati u neku promenljivu, onda scanf uzima kao argument adresu te promenljive, a ne samu promenljivu.
- To znači da pri pozivu funkcije scanf ispred imena promenljive u koju scanf treba učitati vrednost moramo staviti adresni operator &.
- Tako će program

```
int x;
.....
scanf("%d",&x);
```

učitati ceo broj sa ulaza u promenljivu x, dok će program

```
int x;
.....
scanf("%d",x); /* pogrešno */
```

generisati grešku.

Funkcija scanf je blokirajuća f-ja jer blokira izvršavanje programa

V – Funkcija unosa Scanf()

- Najčešće korišćeni znakovi konverzije navedeni su u sledećoj tabeli:

znak konverzije	***tip podatka koji se učitava***
%c	jedan znak (char)
%d	decimalni celi broj (int)
%e,%f,%g	broj sa pokretnim zarezom (float)
%h	kratak celi broj (short)
%i	decimalni, heksadecimalni ili oktalni celi broj (int)
%o	oktalni celi broj (int)
%u	neoznačeni celi broj (unsigned int)
%x	heksadecimalni celi broj (int)
%s	string (char *)
%p	pokazivač (void *)

V – Učitavanje celih brojeva

- Celi brojevi mogu biti uneseni kao decimalni (**%d**), oktalni (**%o**) i heksadecimalni (**%x**)
- Može da se koristi i znak konverzije **%i** pri čemu se ulazni podatak tretira kao oktalan broj ako mu prethodi nula ili kao heksadecimalan broj ako mu prethodi **0x** ili **0X**.

Primer:

Koje su vrednosti promenljivih x, y i z nakon izvršavanja programa:

```
int x,y,z;
.....
scanf("%i %i %i",&x,&y,&z);
```

a učitavaju se sledeći ulazni podaci sa tastature: **13 015 0Xd**

- Isti program možemo napisati i u sledećem obiliku:

```
int x,y,z;
.....
scanf("%d %o %x",&x,&y,&z);
```

Ali sada možemo uneti podatke u sledećem obliku: **13 15 d**

V – Učitavanje realnih brojeva

- Znakovi konverzije e, f, g služe za učitavanje promenljivih tipa float.
- Ukoliko se učitava vrednost u promenljivu tipa double treba koristiti prefiks **l** (**le**, **lf** ili **lg**).

Primer:

```
float x;
double y;
.....
scanf("%f %lg",&x,&y);
```

- Prefiks L koristi se ako je argument pointer na long double.

znak konverzije	***tip podatka koji se ucitava***
%e,%f,%g	broj tipa float
%le,%lf,%lg	broj tipa double
%Le,%Lf,%Lg	broj tipa long double

V – Učitavanje znakovnih nizova

- Znak konverzije **%s** učitava niz znakova; niz završava prvim blanko znakom u ulaznom (učitanom) nizu znakova.
- Iza poslednjeg učitanog znaka automatski se dodaje nul-znak (**n0**).
- U primeru:

```
char string[128];
int x;
.....
scanf("%s%d",string,&x);
```

  funkcija scanf učitava jedan niz znakova i jedan celi broj.
- Kako se svako polje kao argument funkcije interpretira kao pokazivač na prvi elemenat polja, ispred promenljive string ne stavlja se operat.**&**
- Znakom konverzije **%s** nije moguće učitati niz znakova koji sadrži u sebi blanko znake jer oni služe za ograničavanje ulaznog polja.
- Za učitavanje nizova znakova koji uključuju i blanko znak možemo koristiti srednje zagrade kao znak konverzije %[...].
- Unutar srednjih zagrada upisuje se niz znakova.
- Funkcija scanf će učitati u pripadni argument najveći niz znakova sa ulaza koji se sastoji od znakova navedenih unutar srednjih zagrada.

V – Učitavanje znakovnih nizova

Primer: char linija[128];

.....

scanf("␣%[␣ABCDEFGHIJKLMNOPRSTUVWXYZ]", linija);

- Učitava najveći niz znakova sastavljen od velikih slova i razmaka.
- Argument linija mora naravno imati dovoljnu dimenziju da primi sve znakove i zavrsni nul-znak n0.
- Uočimo da smo pre **%[** ostavili jedan razmak koji govori funkciji scanf da preskoći sva blanka koje prethode znakovnom nizu.
- To je nužno ukoliko smo imali prethodni poziv scanf funkcije.
- Naime scanf uvek ostavlja završni znak prelaza u novi red u ulaznom nizu, tako da bi naredba:
  scanf("%[␣ABCDEFGHIJKLMNOPRSTUVWXYZ]", linija);
  pročitala prethodni znak prelaza u novi red i budući da on nije u unutar zagrada završila bi čitanje ulaznih podataka i linija ne bi bila učitana.

Primer: scanf("␣%[^niz znakova]", linija);

- Sada se u odgovarajući argument učitava najveći mogući niz znakova sastavljen od svih znakova osim onih koji se nalaze u sred.zagradama.

V – Maksimalna širina polja unosa

- Uz svaki kontrolni znak moguće je zadati maksimalnu širinu ulaznog polja koje će se učitati tako što se ispred kontrolnog znaka stavi broj koji određuje sirinu polja.
- Tako na primjer %3d učitava ceo broj od najviše tri znaka, a %11c učitava 11 znakova.
- Ukoliko podatak sadrži manje znakova od zadate maksimalne širine polja on se učitava samo do prvog blanko znaka.
- Ako podatak ima više znakova od maksimalne širine polja, višak znakova biće učitan sledećim konverzijskim znakom ili sledećom scanf funkcijom.

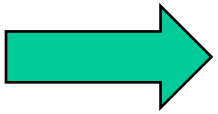
Primer: uzmimo naredbu **scanf(" %3d %3d %3d", &i, &j, &k);**

1. Ako na ulazu imamo 1□2□3, bit će ucitano **i=1, j=2, k=3**.
2. Ako na ulazu imamo 123□456□789 učitaće se **i=123, j=456, k=789**
3. Ako na ulazu imamo 123456789 onda je **i=123, j=456, k=789**
4. Ako na ulazu imamo 1234□56□789 učitaće se **i=123, j=4 i k=56**. Preostali znakovi ostaće na ulazu i biće pročitani novim pozivom scanf funkcije (ili neke druge ulazne funkcije).

V - Funkcija unosa Scanf()

- Funkcija **scanf** vraća broj uspešno učitanih podataka ili EOF.
- Tu činjenicu možemo iskoristiti za proveru da li su svi traženi podaci ispravno učitani.
- Uzmimo jedan primer u kome učitavamo i procesiramo samo pozitivne cele brojeve.

```
int n;
scanf("%d",&n);
while(n >= 0)
{
// radi nesto s brojem
scanf("%d",&n); // ucitaj novi broj
}
```



```
int n;
while(scanf("%d",&n) == 1 && n >= 0)
{
// radi nesto s brojem
}
```

- Ovakav kod ne može uspešno tretirati slučajeve u kojima korisnik učini grešku priklikom upisa (npr. upiše slovo).
- Ponašanje programa možemo popraviti ako ispitujemo da li je funkcija scanf uspešno učitala broj.
- To možemo učiniti testom scanf("%d",&n) ==

V - Funkcija printf()

- Opšta forma funkcije printf() je:

  printf(kontrolni_string, arg_1, arg_2, ... ,arg_n)

- Funkcija **printf** je funkcija standardne biblioteke stdio.h koja prikazuje izlazne podatke u određenom formatu**.**
- Ovom funkcijom se ispisuje poruka zadata format-stringom na standardni izlaz koji je obično monitor našeg računara.
- 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 izlazu.
- Ova funkcija vraća kao vrednost broj ispisanih 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 tu biti ispisana celobrojna vrednost argumenta koji je sa njim u paru.
- Argumenti f-je printf mogu biti konstante, promenljive, izrazi ili polja
- Svaka **%** konstrukcija (specifikacija konverzije) je u paru sa odgovarajućim argumentom koji sledi.

V – Znakovi konverzije

znak konverzije	*tip podatka koji se ispisuje*
%d,%i	decimalni celi broj (int)
%u	celi broj bez predznaka (unsigned int)
%o	oktalni celi broj bez predznaka (unsigned int)
%x	heksadecimalni celi broj bez predznaka (unsigned int)
%e,%f,%g	broj sa pokretnim zarezom (double)
%c	jedan znak (char)
%s	string (char *)
%p	pokazivač (void *)
%Lf	long double ispis bez eksponenta
%hd	short dekadni ispis
%%	koristi se za ispis znaka %
\\	koristi se za ispis znaka \
\”	koristi se za ispis znaka ”

V - Ispisivanje celih brojeva

- Pomoću znakova konverzije %O i %X celi brojevi se ispisuju u oktalnom i heksadecimalnom obliku bez vodeće nule odnosno 0X.

Primer:
```
short i=64;
.....
printf("i(okt)=%O: i(hex)=%X: i(dec)=%d\n",i,i,i);
```

Ispisuje se: **i(okt)=100: i(hex)=40: i(dec)=64**

- Izrazi tipa long ispisuju se pomoću preksa **l**.

Primer:
```
#include <stdio.h>
#include <limits.h>
long i=LONG_MAX;
int main(void){
printf("i(okt)=%lo: i(hex)=%lx: i(dec)=%ld\n",i,i,i);
}
```

U zavisnosti od računara na kome se izvršava biće ispisano:

i(okt)=17777777777: i(hex)=7fffffff: i(dec)=2147483647

- Simbolička konstanta LONG_MAX nalazi se u datoteci **<limits.h>** i predstavlja najveći broj tipa long.

V - Ispisivanje realnih brojeva

- Brojeve tipa float, double i long double možemo ispisivati pomoću znakova konverzije %f, %g i %e.
- U konverziji tipa **f** broj se ispisuje bez eksponenta
- U konverziji tipa **e** broj se ispisuje sa eksponentom.
- U konverziji tipa **g** način ispisa (sa eksponentom ili bez) zavisi o vrednosti koja se ispisuje.

Primer:

```
double x=12345.678;
.........
printf("x(f)=%f: x(e)=%e: x(g)=%g\n",x,x,x);
```

Biće ispisano: **x(f)=12345.678000: x(e)=1.234568e+004: x(g)=12345.7**

- Znakovi konverzije **e, f, g** dobijaju preks **l** ako se ispisuje promenljiva tipa **double** a **L** promenljiva tipa **long double**.

znak konverzije	***tip podatka koji se ispisuje***
%e,%f,%g	broj tipa float
%le,%lf,%lg	broj tipa double
%Le,%Lf,%Lg	broj tipa long double

V – Širina i preciznost poja ispisa

- Uz svaki kontrolni znak moguće je zadati minimalnu širinu ispisa tako da se ispred kontrolnog znaka stavi broj koji određuje širinu ispisa.
- Tako na primer **%3d** ispisuje celi broj sa najmanje tri znaka.
- Ukoliko podatak sadrži manje znakova od zadate minimalne širine polja, do pune širine biće dopunjen vodećim blanko znacima.
- Podatak koji ima više znakova od minimalne širine ispisa biće ispisan sa svim potrebnim znakovima.
- Desno poravnanje je podrazumevano poravnanje.
- Da bi se izvršilo levo poravnanje, između % i odgovarajućeg karaktera dodaje se znak -.

Primer:

```
double x=1.2;
.........
printf("%1g\n%3g\n%g\n",x,x,x);
```

Ispisuje se:

(%1g) **1.2** // koriste se tri znaka
(%3g) **1.2** // koriste se tri znaka
(%g) □□**1.2** // koriste se 5 znakova (2 blanka + 3 znaka)

V – Širina i preciznost polja ispisa

- Pored minimalne širine ispisa kod realnih brojeva moguće je odrediti i preciznost ispisa tj. broj decimala koje će biti ispisane.
- Sintaksa je sledeća: **%a.bf** ili **%a.bg** ili **%a.be** gdje je **a** minimalna sirina ispisa, a **b** preciznost.
- Na primer **%7.3e** znači ispis u **e** formatu sa najmanje 7 znakova, pri čemu će biti dato najviše 3 znaka iza decimalne tačke.

Primer:

```
#include <stdio.h>
#include <math.h>
int main(void){
double pi=4.0*atan(1.0);
printf("%5f\n %5.5f\n %5.10f\n",pi,pi,pi);
}
```

Rezultat ispisa ce biti:

(%5f) **3.141593**

✓ Ispis bez specicirane preciznosti zaokružuje se na šest decimala

(%5.5f) **3.14159**

(%5.10) **3.1415926536**

V – Širina i preciznost polja ispisa

- Širinu i preciznost ispisa moguće je odrediti dinamički tako da se na mesto širine ili preciznosti umesto broja stavi *.
- Celobrojna promenljiva (ili izraz) na odgovarajućem mestu u listi argumenata određuje širinu odnosno preciznost.

Primer:

```
#include <stdio.h>
#include <math.h>
int main(void){
double pi=4.0*atan(1.0);
int i=10;
printf("%*f\n %*.*f\n %5.*f\n",11,pi,16,14,pi,i,pi);
}
```

Ispisuje se:

```
   3.141593          // 11 znaka i 6 decimala
3.14159265358979     // 16 znaka i 14 decimala
3.1415926536         // 12 znaka i 10 decimala
```

V - Ispis znakovnih nizova

- Konverzija tipa **%s** primenjuje se na znakovne nizove (nizove znakova čiji je kraj signaliziran nul-znakom **n0**).
- Zato se mogu ispisati i nizovi znakova koji sadrže blanko znakove.

Primer:
```
char naslov[]="Programski jezik C";
....
printf("%s\n",naslov);
```
Ispisaće: **Programski jezik C** i preći u novi red.

- Preciznost se može koristiti i kod **%s** konverzije.
- Tada znači maksimalni broj znakova koji ce biti prikazan.

Primer: **%5.12s** speficira da će biti prikazano minimalno 5 znakova (dopunjenih blanko znacima kako treba), a maksimalno 12 znakova. Ako je niz znakova duži od 12, višak znakova neće biti prikazan.

Primer:
```
char naslov[]="Programski jezik C";
....
printf("%.16s\n",naslov);
```
Ispisaće: **Programski jezik**

V - Oznake (Flag)

- Svaka grupa znakova za konverziju može sadržati i oznaku.
- Oznaka je znak koji dolazi odmah nakon znaka %; moguće oznake su: **-, +, 0, ' ‘,** blanko znak i #, a značenja su sedeća:

- podatak će biti levo pozicioniran ako je manji od minimalne širine polja

+ znak + će biti napisan ispred pozitivnog broja;

0 vodeća blanka (ako su nužni) biće zamenjeni nulama. Odnosi se samo na numeričke podatke koji su desno pozicionirani i koji su uži od minimalne širine ispisa;

' ' (blanko znak) jedan blanko će prethoditi svakom pozitivnom broju.

(uz o ili x konverziju) osigurava da će oktalni i heksadecimalni brojevi biti ispisani s vodećom 0 ili 0x;

(uz e, f ili g konverziju) osigurava da će decimalna tačka biti ispisana i da će nule na krajnjoj desnoj poziciji broja biti ispisane.

Primer:

```
int i=66;
....
printf(":%6d\n:%-6d\n:%06d\n%#x\n",i,i,i,i);
```

```
: 66
:66
:000066
:0x42
```

V - Primeri

Primer 1:

```
main() {
printf("dobar dan\n");
printf("\n");
printf("ovo je tekst o osnovama C-a\t");
printf("vreme je 6:00");
}
```

Ova funkcija će nam na ekran izbaciti sledeći rezultat:

dobar dan

ovo je tekst o osnovama C-a vreme je 6:00

Primer 2:

```
#include <stdio.h>
#define PI 3.14
main () {
printf("Broj PI ima vrednost %f\n", PI);
}
```

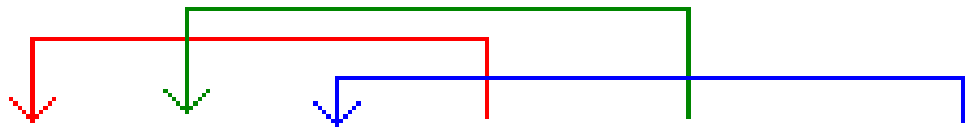
Biće ispisano **Broj PI ima vrednost 3.14**

V - Primeri

Primer 3:

```
#include <stdio.h>
main() {
int broj1, broj2, rezultat;
broj1 = 10;
broj2 = 20;
rezultat = broj1 + broj2;
printf("Rezultat je %d + %d = %d",broj1, broj2, rezultat);
}
```

Dobicemo sledeći ispis na ekranu: **Rezultat je 10 + 20 = 30**



```
printf("Rezultat je %d + %d = %d",broj1, broj2, rezultat);
```

```
#include <stdio.h>
main() {
int broj1=10, broj2=20;
printf("Rezultat je %d + %d = %d",broj1, broj2, broj1+broj2);
}
```

V - Primeri

Primer 4:

```
#include <stdio.h>
main() {
int vrednost;
vrednost = 'A';
printf("%s\nkarakter = %c\nvrednost = %d\n", "veliko slovo", vrednost, vrednost);
vrednost = 'a';
printf("%s\nkarakter = %c\nvrednost = %d\n", "malo slovo", vrednost, vrednost);
}
```

Ovaj program ispisuje:

veliko slovo
karakter = A
vrednost = 65
malo slovo
karakter = a
vrednost = 97

- U prvoj printf() funkciji, promenljiva ima vrednost 'A' i kada ispisujemo char tip promenljive, ipisuje se 'A', ali kada ispisujemo int vrednost (%d), ispisuje se njena ASCII vrednost sto je 65.
- Isto važi i za drugu printf() f-ju gde karakter 'a‘ ima int vrednost 97.

Hvala na pažnji !!!

Pitanja

? ? ?