# PROGRAMIRANJE 1
# UVOD U PROGRAMSKI JEZIK C

Staša Vujičić
Čas 7

# PROGRAM UČITAVA DVA BROJA SA ULAZA I ISPISUJE NJIHOV ZBIR.

```
#include <stdio.h>
/* Linije koje pocinju znakom # su
   tzv. pretprocesorske direktive.
   Ove direktive se izvrsavaju pre
   prevodjenja i tako uticu na
   rezultat prevodjenja. Direktiva
   #include u ovom primeru
   ukljucuje na datom mestu
   sadrzaj datoteke cije je ime dato
   izmedju znakova < i >. */
main()
{
  /* Deklaracije promenljivih */
   int a, b;
   int c;
```

```
/* Osim funkcija koje mi pisemo
   postoje i tzv. funkcije standardne
   bibilioteke koje se isporucuju sa
   razvojnim okruzenjem i koje su
   nam uvek      dostupne. Da bismo ih
   koristili, potrebno je da ukljucimo
   odgovarajuce zaglavlje koje sadrzi
   njihove deklaracije. U tu svrhu se
   koristi #include      preprocesorska
   direktiva. Direktiva sa vrha ovog
   fajla ukljucuje zaglavlje stdio.h
   koje sadrzi deklaracije funkcije
   ulaza i izlaza. Neke od funkcija
   koje su deklarisane u ovom
   zaglavlju su printf, scanf, getchar,
   putchar, fgets, fputs, itd. */
scanf("%d %d", &a, &b);
c = a + b;
printf("Zbir je: %d\n", c);
 }
```

# GRANANJE U PROGRAMU. ODREĐIVANJE VEĆEG OD DVA BROJA.

**Program učitava dva cela broja sa ulaza i prikazuje na izlazu veću od te dve učitane vrednosti.**

```
#include <stdio.h>

main()
{
   /* Deklaracije promenljivih */
   int a = 0, b = 0;

   printf("Uneti dva cela broja: ");

    /* Ucitavamo vrednosti a i b sa ulaza */
   scanf("%d", &a);
   scanf("%d", &b);
```

```
if(a > b)
    printf("MAX(%d,%d) = %d\n", a,
   b, a);
  else
    printf("MAX(%d,%d) = %d\n", a,
   b, b);

}
```

# VIŠESTRUKO GRANANJE. KVADRATNA JEDNAČINA.

- Program učitava koeficijente kvadratne jne a zatim određuje i ispisuje rešenja ako postoje.

```
#include <stdio.h>
#include <math.h>
/* Zaglavlje math.h sadrzi
   deklaracije matematickih
   funkcija poput sqrt(), pow(),
   sin(), cos(), exp() itd. */
main()
{
   float a, b, c;
   float D, x1, x2;
```

```
/* Ucitavanje koeficienata kvadratne
   jednacine. Realni brojevi float tipa
   ucitavaju se pomocu %f
   specifikatora konverzije u scanf-u.
   */
   printf("Uneti koeficiente kvadratne
   jednacine (a b c): ");
   scanf("%f %f %f", &a, &b, &c);

  /* Izracunavamo diskriminantu */
  D = b*b - 4*a*c;
/* Ako je diskriminanta veca od
   nule...*/
  if(D > 0)
    {
      x1 = (-b + sqrt(D))/(2*a);
      x2 = (-b - sqrt(D))/(2*a);
      printf("Jednacina ima dva
   razlicita realna resenja: %f i %f\n",
   x1, x2);
    }
```

```
  /* Ako je jednaka nuli...*/
   else if(D == 0)
     {
       x1 = -b/(2*a);
       printf("Jednacina ima jedno
    realno resenje: %f\n", x1);
     }
   /* Preostali slucaj: ako je manja
    od nule...*/
   else
     {
       printf("Jednacina nema realnih
    resenja\n");
     }
}
```

# PETLJE.
# SUMA BROJEVA SA ULAZA.

- Program učitava cele brojeve sve dok se ne učita nula, a zatim ispisuje zbir svih učitanih brojeva.

```
#include <stdio.h>

main()
{
  /* Suma se obavezno inicijalizuje
   nulom. Slicno, proizvod bi se
     inicijalizovao jedinicom, jer je 1
   neutralni element za mnozenje */
  int S = 0, x;

  /* Ucitavamo prvi element */
  scanf("%d", &x);
```

```
/* Petlja se izvrsava dokle god je x
   razlicito od nule (!= u C-u znaci
   razlicitost). */
  while(x != 0)
    {
      /* Dodajemo ucitani broj na
    prethodnu sumu */
      S = S + x;

      /* Ucitavamo sledeci broj */
      scanf("%d", &x);
    }

  /* Prikazujemo izracunatu sumu
   */
  printf("Suma svih unetih brojeva
   je %d\n", S);
}
```

# SAVRŠEN BROJ.

- Program ispituje da li je dati broj savršen. Broj je savršen ako je jednak zbiru svih svojih pravih delilaca.

```
#include <stdio.h>

main()
{
  int n;  /* Uneti broj */
  int i;  /* Brojac u petlji */
  int S = 0;  /* Suma pravih delilaca
   broja n */

  /* Unosimo broj */
  printf("Uneti broj za koji ispitujemo
   da li je savrsen: ");
  scanf("%d", &n);
```

/* Brojacka petlja je u C-u veoma fleksibilna i ima sledecu sintaksu:

for(izraz1 ; izraz2 ; izraz3) naredba

Najpre se izracunava **izraz1** (on se izracunava samo jednom i najcesce se koristi za inicijalizaciju brojaca i/ili drugih parametara u petlji).

Nakon toga se izracunava **izraz2**. Ovaj izraz predstavlja uslov, kao onaj u while petlji. Ako je uslov ispunjen tada se izvrsava naredba u telu petlje, nakon cega se izracunava **izraz3** (ovaj izraz se obicno koristi za promenu vrednosti indeksa petlje, tipicno uvecanje ili umanjenje).

Nakon toga se ponovo proverava **uslov u izrazu 2**, nakon cega se ako je uslov ispunjen ponovo **izvrsava naredba**, pa opet **izraz3,** itd. sve dok uslov prestane da vazi. Tada se izlazi iz petlje. Ako u telu petlje postoji potreba za vise od jedne naredbe, tada se koristi blok naredba.

*/

```
  /* Za sve brojeve od 1 do n – 1 ...
   */
  for(i = 1; i < n; i++)
    /* ...ispitujemo da li dele n, i
   ako je odgovor potvrdan,
   dodajemo ga na sumu delilaca */
    if(n % i == 0)
      S = S + i;
   /* Ako je suma delilaca jednaka
   broju n...*/
  if(n == S)
    printf("Broj %d je savrsen\n", n);
  else /* ... u suprotnom */
    printf("Broj %d nije savrsen
   (suma delilaca je %d)\n", n, S);
}
```

# PETLJA SA POSTUSLOVOM KONVERZIJA ZAPISA BROJA.

- Program učitava ceo broj u dekadnom zapisu, a zatim ispisuje heksadekadne cifre istog broja u obrnutom poretku.

```
#include <stdio.h>

main()
{
  int x;
  int c;
   /* Algoritam izracunava vrednost
   broja x na osnovu datog niza
   dekadnih   cifara */
  x = 0;
```

```
while((c = getchar()) != EOF)
    if(c >= '0' && c <= '9')
      x = x * 10 + c - '0‘;
```

- /* NAPOMENA: Funkcija **getchar()** ucitava jedan karakter sa ulaza i vraca ceo broj koji predstavlja vrednost ASCII koda ucitanog karaktera.
- U gornjoj petlji se najpre ova vrednost dodeli promenljivoj c, a zatim se proverava da li je to razlicito od EOF.
- Zagrade oko izraza c = getchar() su neophodne zato sto je operator = nizeg prioriteta od operatora !=.
- EOF je konstanta koja je definisana u zaglavlju stdio.h i predstavlja oznaku kraja fajla (najcesce je vrednost ove konstante -1). Kada se dodje do kraja fajla (ili kraja unosa), tada getchar() vraca ovu vrednost kao indikator. Tada se gornja petlja zavrsava. */

- /* NAPOMENA: U gornjoj petlji se koriste tzv. karakterske konstante ('0‘, '9' itd.). Ove konstante se navode izmedju jednostrukih navodnika i predstavljaju ceo broj cija je vrednost ASCII kod navedenog karaktera.
- Drugim recima, kada napisete '0', to je isto kao da ste napisali 48, kada napisete 'A' to je isto kao da ste napisali 65.
- Prednost karakterskih konstanti je to sto se kodovi karaktera predstavljaju na intuitivan nacin (ne moramo da pamtimo ASCII kodove pojedinih karaktera).

*/

- /* NAPOMENA: Izraz:

  **c - '0'**

  predstavlja pomeraj (engl. offset) u okviru ASCII tabele u odnosu na karakter '0'.

  S obzirom da je c ucitana cifra (npr. '5') tada je c - '0’ upravo vrednost broja koji je predstavljen cifrom

  (npr. '5' - '0' == 5, odnosno, ekvivalentno: 53 - 48 = 5, gde su 53 i 48 respektivno ASCII kodovi karaktera '5' i '0').

  */

- /* Petlja sa postuslovom u C-u ima sledecu sintaksu:

  do naredba while(uslov);

  Najpre se izvrsava naredba, nakon cega se proverava uslov. Ako je ispunjen, naredba se ponovo izvrsava, nakon cega se ponovo proverava uslov, itd. Prvi put kada uslov NE BUDE ispunjen, izlazi se iz petlje.

  Samim tim, jasno je da se naredba u telu petlje uvek izvrsava bar jednom.

  */

```
/* Ispis izracunatog broja */
printf("Broj x u dekadnom zapisu:
 %d\n", x);

printf("Heksadekadne cifre broja
 x:\n");
```

```
/* Algoritam izdvaja cifre broja x u
   njegovom heksadekadnom zapisu
   */
  do
    {
      /* Slucaj cifara '0' do '9' */
      if(x % 16 < 10)
         putchar(x % 16 + '0');
      else /* Slucaj cifara 'A' do 'F' */
         putchar(x % 16 - 10 + 'A');

      putchar('\n');

      x = x / 16;
    }
  while(x != 0);
```

- /* NAPOMENA: funkcija **putchar()** ispisuje na standardni izlaz karakter ciji je ASCII kod dat kao argument funkcije. Tako ce putchar(65) ispisati karakter A. Slican efekat imace i poziv putchar('A'), jer je vrednost konstante 'A' upravo 65. */
- /* NAPOMENA: izraz:

  x % 16 + '0'

  ima vrednost ASCII koda odgovarajuce cifre od '0' do '9'. Naime, izraz

  x % 16

  ima vrednost broja 0 do 9. Kada se ovo doda na 48, dobija se upravo ASCII kod odgovarajuce cifre.

Slicno, ako je x % 16 vece ili jednako 10, tada izraz:

```
x % 16 - 10 + 'A'
```

daje ASCII kod karaktera 'A' do 'F'. */

```
putchar('\n');

}
```

# PREPISIVANJE ULAZA NA IZLAZ

- Program prepisuje ulaz na izlaz, zamenjujuci sva velika slova odgovarajućim malim slovima.

```
#include <stdio.h>

main()
{
  int c;
```

```
    while((c = getchar()) != EOF)
            if(c >= 'A' && c <= 'Z')
                    c = c - 'A' + 'a';

/* Ispis karaktera. Ovaj karakter ce
   biti isti kao i ucitani karakter,
   osim ako nije veliko slovo, u kom
   slucaju je prethodno
   konvertovan u malo slovo. Neka
   citalac sam rastumaci zbog cega
   je izraz c - 'A' + 'a'  u tom slucaju
   ASCII kod odgovarajuceg malog
   slova. */

      putchar(c);
    }
}
```

# BROJANJE ZNAKOVA NA ULAZU

- Program demonstrira for petlju, ulaz i izlaz znakova, funkcije putchar() i getchar().
- Program broji karaktere na ulazu, uz posebno brojanje malih, velikih slova, cifara i znakova za novi red (tj. linija na ulazu).

```
#include <stdio.h>

main ()
{
  int c;

  int br_karaktera = 0;
  int br_linija = 0;
  int br_malih = 0, br_velikih = 0;
  int br_cifara = 0;
```

```
/* Petlja se izvsava sve dok ne
   procitamo znak za kraj fajla.  U
   svakoj iteraciji petlje se broje
   karakteri. U telu petlje se jos
   dodatno broje mala i velika
   slova, cifre i linije teksta
*/
  for (; (c = getchar ()) != EOF;
   br_karaktera++)
    {
      if (c == '\n')
        br_linija++;
      else if (c >= 'A' && c <= 'Z')
        br_velikih++;
      else if (c >= 'a' && c <= 'z')
        br_malih++;
      else if (c >= '0' && c <= '9')
        br_cifara++;
    }
```

```
/* Ispisujemo izlazne rezultate */
  printf ("Broj linija %d\nBroj
   karaktera %d\nBroj cifara %d\n",
          br_linija, br_karaktera,
          br_cifara);
  printf ("Broj malih slova %d\nBroj
   velikih slova %d\n",
          br_malih, br_velikih);

}
```

# ARITMETIČKA SREDINA I NAJVEĆI ELEMENT

- Program demonstrira for petlju. Unosi se broj elemenata, a zatim i sami elementi. Program ispisuje najveći od unetih brojeva, kao i aritmetičku sredinu unetih brojeva.

```
#include <stdio.h>

main ()
{
  int n;          /* Broj elemenata */
  double d;       /* Uneti element */
  int i;          /* Brojac u petlji */
  double max, s;  /* maksimum i
   aritmeticka sredina */

/* Unosimo n */
  printf ("Uneti broj elemenata: ");
  scanf ("%d", &n);
```

```
/* U petlji unosimo elemente i
   odmah vrsimo poredjenje sa do
   sada najvecim, kao i dodavanje
   na sumu. Suma se pre petlje
   inicijalizuje na 0 */
  printf ("Uneti elemente: ");
  s = 0.0;
  for (i = 0; i < n; i++)
    {
      /* Unos elementa */
      scanf ("%lf", &d);
      /* Uslov i == 0 sluzi da prvi put
    postavimo max na prvi uneti
    element. Drugi uslov se koristi u
    kasnijim iteracijama */
      if (i == 0 || d > max)
       max = d;
    /* Dodavanje na sumu */
      s += d;
    }
```

```
/* Delimo sumu brojem unetih
   brojeva, kako bismo dobili
   aritmeticku sredinu. Primetimo
   da se prilikom deljenja vrednost
   n konvertuje u tip double
   implicitno, zato sto je operand
   s tipa double, koji je, opet, siri
   tip od int-a. Rezultat je takodje
   tipa double */
  s /= n;

/* Ispisujemo izlazne rezultate */
  printf ("Najveci uneti broj je:
   %f\n", max);
  printf ("Aritmeticka sredina
   unetih brojeva je: %f\n", s);
}
```