

# Funkcije

---

- **Pojam funkcije**
- **Deklaracija i definicija funkcije**
- **Poziv funkcije**
- **Memorijske klase promjenljivih**
- **Primjeri.**

Za kompajliranje koda koristen DEV-C++ 4.9.9.2 Compiler !!!!

# Funkcije...

- Šta je funkcija?
- Segment programa koji ima tačno definisan zadatak.
- Funkcija ima jedinstveno ime.
- Koristi argumente.
- Izvršava naredbe definisane kodom
- Vraća rezultat

```
//ilustracija funkcija
#include <stdio.h>
#define PI 3.14

// funkcija za izracunavanje povrsine kruga
float površina_kruga(float d)
{
    float r,p;
    r=d/2;
    p=r*r*PI;
    return p;
}

// main funkcija
int main()
{
    float pr, pov;
    printf("\nUnesi precnik kruga: ");
    scanf("%f", &pr);
    // pov= PI*(pr/2)*(pr/2)
    //Preko funkcija
    pov=površina_kruga(pr);
    printf("\nPovrsina kruga je %f\n", pov);
    getchar();
    getchar();
    return 0; }
```

# Funkcije...

---

- **Prednosti programa sa funkcijama**
  - Jednostavnije rešavanje programskog zadatka.
  - Jednostavnije pisanje i održavanje programa.
  - Piše se jednom, a poziva potreban broj puta.
  - Moguće je kreirati biblioteke funkcija.
  - Veći nivo abstrakcije i razumijevanja samog programa.
  - Mogućnost jednostavnijeg ispravljanja, dopuna i izmjena.
  - Jednostavno, program bez funkcija nije dobar program.

# Funkcije...

---

- Deklaracija (prototip) funkcije
  - `tip ime( lista_argumenata) ;`
  - `tip ime( tip1, tip2,...) ;`
  - `tip ime( tip1 ime1, tip2 ime2,...) ;`

**tip** – tip povratne vrijednosti od funkcije, bilo koji tip.

**ime** – bilo koji C identifikator.

**lista\_argumenata** – lista tipova (eventualno i formalnih imena podataka koje koristi funkcija)

# Funkcije...

---

- **Definicija funkcije**

```
tip ime( lista_argumenata) // Deklaracija bez znaka ; na kraju
{
;
// Tijelo funkcije, samo naredbe
;
}
```

**U tijelu funkcije mogu biti različite naredbe**

# Funkcije...

---

- **Argumenti i povratne vrijednosti.**
- **Način razmjene podataka.**
- **Argumenata može biti više (0, 1, 2, 3....).**
- **Povratnih vrijednosti može biti 1 (jedna) ili više (način kada se vraća više vrijednosti kroz složeni tip podataka)**

# Funkcije...

---

- **Lista formalnih argumenata u definiciji funkcije**
  - Odgovara listi argumenata u deklaraciji funkcije.
  - Pojedini argumenti u listi odvajaju se “,”
  - Imena formalnih argumenata u ovoj listi-obavezna.
  - Ako funkcija nema argumenata, umjesto njihove liste se piše **void**.

# Funkcije...

---

- **Tip povratne vrijednosti**
  - **Može biti osnovni ili izvedeni C tip.**
  - **Funkcija može vratiti na mjesto poziva najviše jednu povratnu vrijednost (može biti i adresa na početak niza).**
  - **Ako nema povratne vrijednosti umjesto tipa se piše void.**



# Funkcije

---

- **Tipovi povratnih vrijednosti**

**char f1(...);**

f1 vraća vrijednost tipa char

**int f2(...);**

f2 vraća vrijednost tipa int

**float f3(...);**

f3 vraća vrijednost tipa float

**double f4(...);**

f4 vraća vrijednost tipa double

**void f5(...);**

f5 ne vraća vrijednost na mjesto svog poziva, već rezultate šalje samo izlaznim uređajima (funkcija sa “bočnim efektima”).

# Funkcije



- **Naredbe funkcija**
  - Naredbe deklaracije – na početku funkcije.
  - Izvršne naredbe – poslije naredbi deklaracije.
  - Naredbe povratka – obavezno bar jedna ako funkcija vraća vrijednost.

# Funkcije

---

- **Naredba povratka return**

`return ( vrednost );`       $\leftrightarrow$  `return vrednost ;`  
`return x;`

`return ( (x>=y)? x : y );`

```
if(x>=y)
    return x;
else
    return y;
```

Može biti više naredbi return ali moguće je izvršenje samo jedne od njih (za jednu povratnu vrijednost).

# Funkcije

---

- **Poziv funkcije**

**ime( lista stvarnih argumenata);**

- **Poziv funkcije je operator u C jeziku.**
- **Pojedini argumenti odvajaju se zarezom “,”.**
- **Ako funkcija nema argumenata, piše se samo par zagrada ( ).**

# Funkcije

- Primjer funkcije za množenje dva cijela broja i vraća rezultat u obliku cijelog broja.

```
int proizvod(int x, int y);           // Deklaracija funkcije
main()
{ int a=10, b=5;
    printf( "%d", proizvod(a,b) );    // Poziv funkcije
}
```

```
int proizvod(int x, int y)           // Pocetak definicije funkcije
{
    int z;                            // Naredba funkcije
    z=x*y;                             // Izvrsna naredba
    return z;                          // Naredba povratka
}
```

# Funkcije

---

- Jednostavniji primjer gornje funkcije

```
int proizvod(int x, int y);
main()
{ int a=10, b=5;
    printf( "%d", proizvod(a,b) );
}
int proizvod(int x, int y)
{
    return (x*y);
}
```

# Funkcije

- Deklaracija i definicija
- Kompajler će da prihvati i sledeću situaciju, bez **deklaracije** funkcije, ali **definicija** mora doći ispred funkcije u kojoj se poziva.

```
# include <stdio.h>
// izostavljena deklaracija
int proizvod(int x, int y)
{
    return (x*y);
}
main()
{ int a=10, b=5;
    printf("%d", proizvod(a,b) );
    getchar();
}
```

# Funkcije

---

- Opciono izvršavanje jedne od predvidjenih naredbi **return**

```
float kolicnik(int x, int y);
main()
{ int a=10, b=3;
    printf( "%f", kolicnik(a,b) );
}
float kolicnik(int x, int y)
{    if (y==0)
        return 0;
    else
        return ( (float)x / y );
}
```



# Funkcije

---

- **Rekurzivna funkcija**
- Funkcija koja poziva samu sebe
  - konačan broj puta
  - svaki put sa različitim vrijednostima
- Uvijek postoji:
  - iterativni postupak koji može zamijeniti rekurzivni
- Prednost:
  - jednostavniji algoritam od iterativnog

## Nedostatak:

- može biti veliki broj poziva iste funkcije, a svaki put se pokreće mehanizam poziva.

# Funkcije

- Iterativni postupak u funkciji faktorijel

```
long ifakt(int n)
{ int i, fakt=1; /*na pocetku fakt=1*/
  for(i=2; i<=n; i++)
    fakt *=i; /*mnozenje u svakoj iteraciji sa i*/
  return fakt;
}
```

n-1 puta izvršava operaciju  $\text{fakt}=\text{fakt} * i$

$\text{ifakt}(n) = 1 * 2 * 3 * \dots * (n-2) * (n-1) * n$

U jednom pozivu funkcije (n-1) iteracija

# Funkcije

- **Rekurzivni algoritam funkcije faktorijel**

```
long rfakt(int n)
{ if(n==1)
  return 1; /*kraj izvršavanja funkcije*/
else
  return( n*rfakt(n-1) ); /*funkcija poziva samu sebe*/
}
```

$rfakt(n) = n * rfakt(n-1)$   
 $(n-1) * rfakt(n-2)$   
 $(n-2) * rfakt(n-3)$   
.....  
 $2 * rfakt(1)$   
 $1$

Funkcija poziva samu sebe (n-1) puta

# Funkcije

---

- **Glavna funkcija**

```
main()
{ int x,y;
.
.
.
return 0;
}
```

Po pokretanju programa operativni sistem poziva njegovu glavnu funkciju.

Poslednja funkcija glavne funkcije vraća status operativnom sistemu.

# Funkcije

- **Identičnost, ugradejnje konstante**

```
main()
{ .
.
.
return 0;
}
```



```
main()
{ .
.
.
return EXIT_SUCCESS;
}
```

U operativnom sistemu  
Rezervisana konstanta

# Funkcije

---

- **Funkcija za prevereni kraj izvršenja programa**

Funkcija **exit()** iz **stdlib.h**

```
void exit( int status ); // Deklaracija u biblioteci  
exit( status ); // Poziv
```

Pozivom funkcije **exit()** prosledjuje se status operativnom sistemu: 0/1

Primjeri poziva :

-U slučaju da nema greški

```
exit(0); exit(EXIT_SUCCESS);
```

- U slučaju greške

```
exit(1); exit(EXIT_FAILURE);
```

# Funkcije...

- Funkcija za poziv drugog programa u okviru postojećeg operativnog sistema, a iz bilo koje funkcije C koda

Funkcija **system()** iz **stdlib.h**

```
int system(const char *ime);
```

```
system("ime_programa");
```

(Vraća 0 ako dobije u argumentu ime programa, u suprotnom -1)

**system()** prekida tok izvršavanja jednog programa, poziva se i izvršava drugi program i nakon toga nastavlja izvršavanje prekinutog programa.

Primjeri poziva DOS komandi:

```
system("cls");
```

```
system("dir");
```

```
system("ime_drugog_programa");
```

# Funkcije vs Makroi

## Makroi sa argumentima

- **Makro** je definicija simbola u kojoj se javljaju jedan ili više argumenata.
- Na primer:

```
#define KVADRAT(x) ((x) * (x))
```

definiše makro koji izračunava drugi stepen argumenta. Argument x se u programu zamenjuje stvarnim argumentom proizvoljnog tipa.

Vidimo da je parametrizirani makro vrlo sličan funkciji no u njemu nema funkcijskog poziva i prenošenja argumenata, pa je stoga efikasniji. Makro je brzi, a funkcija zauzima manje memorije.

ISTO!!!!

```
//Ilustracija funkcija i MAKROA
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14
// MAKRO za površinu
#define POV(x) ((x/2)*(x/2)*PI) //MAKRO
// funkcija za površinu
float površina_kruga(float d)
{
    float r,p;
    r=d/2;
    p=r*r*PI;
    return p;
}

// main funkcija
int main()
{
    float pr, pov;
    printf("\nUnesi prečnik kruga: ");
    scanf("%f", &pr);
    pov=površina_kruga(pr);
    printf("\nPovršina kruga je %f\n", pov);
    printf("\nPovršina kruga je %f\n", POV(pr));
    getchar();
    getchar();
    return 0;
}
```



# Memorijske klase promenljivih

- **4 memorijske klase promjenljivih:**

- globalne

definišu se izvan svih funkcija

- staticke

definišu se korišćenjem ključne riječi static  
static tip ime=vrijednost

- automatske

definišu se korišćenjem ključne riječi auto  
auto tip ime=vrijednost  
podrazumijeva se da je promjenljiva automatska, ako se drugacije  
ne navede

- registarske

definišu se korišćenjem  
ključne riječi register  
register tip ime=vrijednost

# Memorijske klase promenljivih

Automatske ili lokalne promjenljive definišu se na početku tijela funkcije (glavnog programa ili potprograma).

- Pocetna vrijednost im se ne podrazumijeva!!! (To znaci da nije nula!!!)
- Mogu da se koriste samo u funkciji u kojoj su definisane. To znaci da isto ime možemo da koristimo nezavisno u više razlicitih funkcija. Cak te promjenljive mogu da budu i razlicitih tipova. Automatski nastaju prilikom ulaska u funkciju, a automatski nestaju nakon izlaska iz funkcije.
- Podrazumijeva se da je promjenljiva automatska ako se drugacije ne navede Ne mora se eksplicitno navoditi da se radi o automatskoj promjenljivoj, ali može pomocu kljucne rijeci auto

***auto tip ime=vrijednost;***

# Memorijske klase promenljivih

Definišu se navodenjem ključne riječi `static`

***static tip ime=vrijednost***

- Vrijednost im se određuje prije početka izvršavanja programa uzima se eksplicitna vrijednost kojom je inicijalizovana promjenljiva podrazumijeva se početna vrijednost nula ako se ne navede
- Imaju trajan karakter, postoje od početka do kraja izvršavanja programa vrijednost ostaje sacuvana do sljedeceg poziva date funkcije

# Memorijske klase promjenljivih

- ***Globalne promjenljive definišu se izvan svih funkcija. Ako se u definiciji ne navede početna vrijednost podrazumijeva se nula!!!***
- ***Globalna promjenljiva dostupna je u svim funkcijama koje su definisane nakon definicije date promjenljive.***
- ***Globalne promjenljive omogućavaju da više funkcija manipuliše istim skupom podataka, pa omogućavaju efikasnu razmjenu podataka između funkcija (jer sve funkcije pristupaju istim memorijskim lokacijama).***
- ***Treba ih izbjegavati jer se gubi univerzalnost funkcije i fleksibilnost primjene na različite setove podataka.***
- ***Ako se u nekoj funkciji definiše promjenljiva sa istim imenom kao i neka globalna, tada ta lokalna promjenljiva maskira globalnu i globalnoj ne može da se pristupi.***
- ***Ako se u nekoj funkciji koristi neka globalna promjenljiva to može da se eksplicitno naglasi deklaracijom korišćenjem ključne riječi extern, iako ne postoji obaveza da se globalne promjenljive deklariraju u funkciji extern tip promjenljiva:***

# Memorijske klase promenljivih

## Primjer lokalne promenljive

```
#include <stdio.h>
void f1 ()
{ int i=1;
printf("f1: %d\n", i);
}

void f2 ()
{ int i=2;
printf("f2: %d\n", i);
i=10;
}

main()
{
int i=0;
printf("main: %d\n", i);
f1();
printf("main: %d\n", i);
i=5;
f2();
printf("main: %d\n", i);
}
```

```
main: 0
f1: 1
main: 0
f2: 2
main: 0
```

# Memorijske klase promenljivih

## Primjer globalne promenljive

```
#include <stdio.h>
void f1 ()
{ int i=1;
  printf("f1: %d\n", i);
}
int i;

void f2 ()
{ extern int i;
  printf("f2: %d\n", i);
  i=10;
}

main()
{
  printf("main: %d\n", i);
  f1();
  printf("main: %d\n", i);
  i=5;
  f2();
  printf("main: %d\n", i);
}
```

```
main: 0
f1: 1
main: 0
f2: 5
main: 10
```

## Primjer staticke promenljive

```
#include <stdio.h>
void f ()
{
  static int brs=1;
  auto int bra=1;
  printf("static brs=%d auto
  bra=%d\n", brs, bra);
  brs++;
  bra++;
}

main()
{
  int i;
  for (i=1; i<=3; i++)
  f();
}
```

```
static brs=1 auto bra=1
static brs=2 auto bra=1
static brs=3 auto bra=1
```

# Funkcije

- Primjer 1

```
/*Odredjivanje veceg od dva broja pomocu odgovarajuce funkcije*/
#include <stdio.h>
int max(int x, int y); /*deklaracija funkcije*/
main( )
{
    int a, b, c;
    printf("\nZadati dva cela broja: "); scanf("%d%d", &a, &b);
    c = max(a, b); /*poziv funkcije*/
        if(c == 0) printf("\nZadati brojevi su isti!!!\n");
        else printf("\nVeci od dva zadata broja je broj %d\n\n", c);
    return 0;
}
```

```
int max(int x, int y)
{ if(x == y) return 0;
  else return ((x>y)?x:y);}
```

# Funkcije

- Primjer 2

```
/*Odredjivanje najveceg od 4 broja pomocu odgovarajucih funkcija*/
```

```
#include <stdio.h>
```

```
int max(int x, int y);
```

```
int max4(int x, int y, int z, int w);
```

```
main( )
```

```
{
```

```
int a, b, c, d;
```

```
printf("\nZadati cetiri cela broja: ");
```

```
scanf("%d%d%d%d", &a, &b, &c, &d);
```

```
printf("\nNajveci od cetiri zadata broja je %d\n\n", max4(a,b,c,d));
```

```
return 0; }
```

```
int max(int x, int y)
```

```
{ return ((x>=y)?x:y); }
```

```
int max4 (int x, int y, int z, int w)
```

```
{ return (max(max(x,y), max(z,w))); }
```



# Funkcije

- Primjer 3

```
/* Odredjivanje faktorijela pomocu iterativne funkcije */
#include <stdio.h>
#define OPSEG 12 /*da faktorijel broja bude u opsegu long*/
long ifakt(int x);
main()
{ int broj;
  do
  { printf("\nUnesite ceo broj u opsegu od 1 do %d: ", OPSEG);
    scanf("%d", &broj);
  }while(broj<1 || broj>OPSEG);
printf("\nFaktorijel broja %d je %ld\n\n", broj, ifakt(broj));
return 0; }
long ifakt(int x)
{ long rezultat = 1;
  while(x>1)
  { rezultat *=x; x--; }
return rezultat; }
```

# Funkcije

- Primjer 4

```
/*Izracunavanje faktoriijela zadatog broja pomocu rekurzivne funkcije*/
#include <stdio.h>
#define OPSEG 12 /*da faktoriijel broja bude u opsegu long*/
long rfakt(int x);
main()
{ int broj;
  do
  { printf("\nUnesite ceo broj u opsegu od 1 do %d: ", OPSEG);
    scanf("%d", &broj);
  }while(broj<1 || broj>OPSEG);
printf("\nFaktoriijel broja %d je %ld\n\n", broj, rfakt(broj));
return 0; }
long rfakt(int x)
{ return ((x>1)? (x*rfakt(x-1)) : 1); }
```


# Funkcije

---

- Primjer 5

```
//Crtanje na ekranu okvira zadatih dimenzija
#include <stdio.h>
#define HLIN '-' //znak horizontalna crta okvira
#define VLIN '|' //znak vertikalna crta okvira
#define UGAO '+' //znak za ugao okvira
#define PRAZNO ' ' //prazan znak
#define MAX1 50 //dozvoljena duzina okvira
#define MAX2 20 //dozvoljena sirina okvira
void hokvir(int l);
void vokvir(int l);
void znakovi(int n, char c);
```

# Funkcije



---

```
main()                                // NASTAVAK
{ int i, duzina, sirina;
do
{ printf("\nUnesite duzinu okvira (od 1 do %d):
",MAX1);
scanf("%d", &duzina);
}while(duzina<1 || duzina>MAX1);
do
{ printf("\nUnesite sirinu okvira (od 1 do %d): ",MAX2);
scanf("%d", &sirina);
}while(sirina<1 || sirina>MAX2);
//Stampanje gornjeg dela okvira
hokvir(duzina);
//Stampanje srednjeg dela okvira
for(i=0; i<sirina; i++)
vokvir(duzina);
//Stampanje donjeg dela okvira
hokvir(duzina);
getchar();
getchar();
return 0; }
```

# Funkcije

---

```
void hokvir(int l)
{ putchar(UGAO);
  znakovi(l, HLIN);
  putchar(UGAO);
  putchar('\n');
}
void vokvir(int l)
{ putchar(VLIN);
  znakovi(l, PRAZNO);
  putchar(VLIN);
  putchar('\n'); }
void znakovi(int n, char c)
{ while(n>0)
  { n--; putchar(c); }
}
```

// NASTAVAK

# Funkcije

● Rata za kredit:  $r = (A * p * k) / (A - 1)$

p – pozajmica (iznos kredita)

k – mesečna kamata

m – broj mesečnih rata

$$A = (1 + k)^m$$

● Primer izvršavanja:

Iznos kredita? 100

Mesečna kamata? 0.01

Broj mesečnih rata? 10

Mesečna rata: 10.56

# Funkcije

```
int main()
{
    double p;    // pozajmica (iznos kredita)
    double k;    // kamata (8% = 0.08)
    int m;       // broj mesecnih rata
    double r;    // rata za kredit
    double A;    // pomocna promenljiva

    printf("Iznos kredita? ");
    scanf("%lf", &p);
    printf("Mesecna kamata (npr. 8%%=0.08)? ");
    scanf("%lf", &k);
    printf("Broj mesecnih rata? ");
    scanf("%d", &m);

    A = stepen(1+k, m);
    r = (A*p*k)/(A-1);
    printf("Mesecna rata: %8.2f\n", r);
    return 0;
}
```

# Funkcije

---

- Izracunavanje stepena broja

```
double stepen(double x, int n)
{
    double y = 1.0;
    int i;

    for(i = 0; i < n; i++)
        y = y * x;
    return y;
}
```