Funkcije

predavač: Nadežda Jakšić

Programiranje programski jezik C++

Zašto se koriste funkcije?

- do sada su korišćene "gotove" funkcije iz standardnih biblioteka (cin, cout...) – one su pozivane iz main funkcije koja je glavna funkcija u programu jer izvršavanje programa počinje od nje
- korisničke funkcije – imenovani i samostalni delovi kôda koji izršavaju neki zadatak - pišu se nezavisno i pozivaju se iz main funkcije
- koriste se kada postoji više ponavljanja istih blokova naredbi (operacija) u različitim delovima programa (time se skraćuje tekst programa
- upotrebom funkcija, program se deli na odvojene blokove, svaki blok radi određeni posao, ovim se olakšava čitljivost programa - osmišljavanje, razumevanje i održavanje manjih blokova koda je lakše
- deo programa se može izvršiti sa različitim parametrima, ovi parametri se prosleđuju funkciji kao argumenti funkcije

Definicija

sintaksa:
tipRezultata imeFunkcije (formalniArgumenti)
{
telo funkcije (definicije, deklaracije, naredbe)
}

- tipRezultata je tip koji ima rezultat funkcije; rezultat je ono što funkcija vraća kada se pozove; ako se ne navede podrazumeva se int; kod funkcija koje ne vraćaju vrednost navodi se void
- imeFunkcije je identifikator
- formalniArgumenti su argumenti pomoću kojih se unose početni podaci u funkciju; oni rezervišu mesta za veličine sa kojima funkcija operiše posle njenog poziva; za svaki argument mora posebno da se navede tip (int a, int b, float c...)
- vrednosti koje se dodeljuju formalnim argumentima nazivaju se stvarni argumenti

Definicija

```
//funkcija vraća vrednost
float Kvadriraj (float x)
{
        float rezultat;
        rezultat = x * x; // ili return (x * x);
        return rezultat;
 }
poziv funkcije: a=Kvadriraj (b);
//b mora da bude float

//funkcija ne vraća vrednost
void ispisUpozorenja()
 {
        cout << "Upozorenje!";
 }
poziv funkcije: ispisUpozorenja();
```

naredba return se koristi za povratak iz funkcije; rezultat koji ova funkcija vraća može da bude promenljiva, pokazivač ili aritmetički izraz
return (povratnaVrednost)

tip funkcije koja ne vraća vrednost je void

poziv funkcije:
ako funkcija vraća vrednost
imeFunkcije (stvarniParametri)

ako ne vraća vrednost
imeFunkcije ();

Primer

```
//funkcija vraća vrednost – zbir dva broja
.....
int zbir (int a, int b)    //funkcija će vratiti rezultat tipa int, formalni
{      //argumenti su int a, int b, za svaki mora posebno da se navede tip
            int c;
            c=a+b;
return c; //može i return a+b; naredba return vraća zbir koji je tipa int
}
int main ()
{
            int broj1, broj2;
            cout<<"Unesite dva broja"<<endl;
            cin>>broj1>>broj2;
            cout<<"Zbir ova dva broja je"<<zbir (broj1, broj2);
return 0;}
```

poziv funkcije zbir, broj1 i broj2 su stvarni argumenti

Primer

```
//funkcija ne vraća vrednost
.....
void pozdrav ()
{
        cout<<"Zdravo"<<endl;
}
int main ()
{
        int i;
        for (i=1;i<=10;i++)
        {
                pozdrav (); //poziv funkcije koja ne vraća vrednost
        }
return 0;
}
```

Primer

```
//funkcija ne vraća vrednost – zbir dva broja
.....
void zbir (int a, int b) //funkcija ima tip void
{
        int c;
        c=a+b;
        cout<<"Zbir ova dva broja je"<<c;
}
int main ()
{
        int broj1, broj2;
        cout<<"Unesite dva broja"<<endl;
        cin>>broj1>>broj2;
        zbir (broj1, broj2); //poziv funkcije zbir
        return 0;}
```

Primer

```
//funkcija vraća vrednost – kub nekog broja
int kub (int x)
{
        int rez;
        rez=x*x*x;
        return rez;
}
int main ()
{
        int a,k;
        cout<<"Unesite broj"<<endl;
        cin>>a;
        k=kub (a);
        cout<<"Kub unetog broja je"<< k;
return 0; }
```

Primer

```
//funkcija računa prosek dva broja

float nadjiProsek (float a, float b)
{
        float prosek;
        prosek=(a+b)/2;
        return (prosek);
}
int main ()
{
        float a=5, b=15, rezultat;
        rezultat=nadjiProsek (a,b);
        cout<<"prosek je " << rezultat;
}
```

Primer

```
//funkcija prikazuje kvadrate brojeva od 1 do 9

void kvadrati ()
{
        int i;
        for (i=1;i<=9;i++)
        cout<< i*i;
}
int main ()
{
        kvadrati ();
        return 0;
}
```

Primer

```
// funkcija može da se poziva više puta u okviru programa

int oduzimanje (int a, int b)
{
int r;
r=a-b;
return r;
}
int main ()
{
int x=5, y=3, z;
z = oduzimanje (7,2);
cout << " Prvi rezultat je " << z << '\n'; //5
cout << " Drugi rezultat je " << oduzimanje(7,2) << '\n'; //5
cout << " Treci rezultat je " << oduzimanje(x,y) << '\n'; //2
z= 4 + oduzimanje(x,y);
cout << " Cetvrti rezultat je" << z << '\n'; //6
}
```

Prost broj

```
//funkcija ispituje da li je broj prost u intervalu od m do n
int prostBroj (int broj);
int main()
{  int m,n,i;
  cout<<"Unesite dva broja " <<endl;
  cin>>m>>n;
  cout<<"Prosti brojevi u intervalu od " <<m<< " do " <<n<< " su " <<endl;
  for (i=m; i<=n; i++)
  {          if (prostBroj (i))
             cout<< i<<endl; }
  return 0;}
int prostBroj (int broj) {
int i, prost=1;
for( i=2;i<=broj/2;i++)
 {      if (broj%i==0) {
           prost=0;
           break; }
  }  return prost; } //obezbediti da početak intervala bude broj koji je veći od
                                  jedan
```

vrednost promenljive prost se vraća u glavnu funkciju i ako promenljiva ima vrednost jedan prikazuje se broj

Prosleđivanje po vrednosti

dva načina za prosleđivanje vrednosti formalnim argumentima:

- prosleđivanje (pozivanje) po vrednosti (formalnim argumentima se dodeljuje vrednost stvarnog argumenta, nema izmene stvarnog argumenta po izlasku iz funkcije)

```
void zameniMesta (int a, int b)
 {
int pomocna;
pomocna = a;
a = b;
b = pomocna;
 }
```

- funkcija pravi svoje privatne kopije parametara i koristi te kopije pri izvršavanju - po završetku, prostor koji je funkcija zauzela za kopije se oslobađa i vrednosti kopija se pri tome gube, a originalne promenljive ostaju nepromenjene

Prosleđivanje po referenci

- prosleđivanje po referenci (ako se umesto vrednosti stvarnih argumenata prosleđuju adrese tih argumenata onda se menja vrednost stvarnim argumentima kada se izađe iz funkcije)

```
void zamenaMesta (int& a, int& b)
 {
int pomocna;
pomocna = a;
a = b;
b = pomocna;
}
```

- promene koje funkcija napravi na parametrima su trajne, to jest događaju se na originalnim promenljivama sa mesta poziva funkcije

Primer

```
//zamena mesta
#include <iostream>

using namespace std;

void zamenaMesta (int &a, int &b)
    {
    int pomocna;
    pomocna = a;
    a = b;
    b = pomocna; }
int main()  {
int broj1,broj2;
cout<<"Unesite dva broja"<<endl;
cin>>broj1>>broj2;
zamenaMesta (broj1,broj2);
cout<<"Posle zamene: "<<broj1 << broj2<<endl;
return 0;}
```

Primer

```
//dupliranje unetih vrednosti

void dupliranje(int& a, int& b, int& c)
{
  a*=2;
  b*=2;
  c*=2;
}

int main ()
{
  int x=1, y=3, z=7;
  dupliranje (x, y, z);
  cout << "x=" << x << ", y=" << y << ", z=" << z; //2,6,14
  return 0;
}
```

Programi

1. funkcija koja n puta ispisuje neki tekst, broj n se unosi u glavnom programu
2. u glavnoj funkciji se unose tri broja, napisti funkciju koja ispisuje najveći od tri broja
3. u glavnoj funkciji učitati n brojeva, napisti funkciju koja računa aritmetičku sredinu unetih brojeva
4. napisti funkciju koja prihvata dva broja i vraća veći od ta dva broja
5. u glavnoj funkciji učitati dva broja, napisti funkciju koja računa zbir brojeva u intervalu od prvog do drugog broja
6. u glavnoj funkciji učitati broj n, napisati dve funkcije, prva treba da računa sumu parnih brojeva do n, a druga proizvod neparnih brojeva do n
7. napisati funkciju koja zamenjuje vrednosti dvema promenljivima
8. učitavati brojeve dok su pozitivni, za svaki od pozitivnih brojeva u funkciji izračunati i ispisati njegov koren

Rekurzivne funkcije

rekurzivne funkcije su funkcije koje pozivaju same sebe

primer: računanje faktorijela unetog broja (npr. 3!=3*2*1) bez rekurzije

```
int f1 (int n) //n je 3
{
if (n==0)
        return 1;
else
        return (n*f2(n-1));
}
int f2 (int n) //n je 2
{
if (n==0)
        return 1;
else
        return (n*f3(n-1));}
```

```
int f3 (int n)
{
if (n==0)
        return 1;
else
        return (n*f4(n-1));// n je 1
}
int f4 (int n) //n je nula
{
if (n==0)
        return 1;
else
        return (n*f5(n-1));
}
```

Faktorijel broja 3 - rekurzivno

```
1. int f1(int n)
2. {
3. if (n==0)
4.         return 1;
5. else
6. return (n*f1(n-1));
7. }
8. int main ()
9. {
10.int r:
11.r=f1(3);
12.cout<<"tri faktorijel je " <<r;
13.return 0;
14.}
```

dok main izvršava liniju 11 poziva f1(3)

dok f1(3) izvršava liniju 6 poziva f1(2)

dok f1(2) izvršava liniju 6 poziva f1(1)

dok f1(1) izvršava liniju 6 poziva f1(0)

dok f1(0) izvršava liniju 4 (n==0), vraća vrednost 1 funkciji f1(1)

f1(1) vraća vrednost 1*1=1 funkciji f1(2)

f1(2) vraća vrednost 2*1=2 funkciji f1(3)

f1(3) vraća vrednost 3*2=6 funkciji main

funkcija main u liniji 12 prikazuje broj 6

Rekurzivne funkcije

1. suma kvadrata od jedan do n
2. suma kvadrata od m do n
   - uzlazna rekurzija
   - silazna rekurzija
3. funkcija računa sumu neparnih brojeva od jedan do zadatog (u main funkciji) broja n; u main funkciji ispisati rezulatat
   - funkcija main proverava da li je broj neparan
   - rekurzivna funkcija proverava da li je broj neparan
4. suma brojeva deljivih sa tri od jedan do zadatog u (u main funkciji) broja n; u main funkciji ispisati rezulatat
   - funkcija main proverava da li je broj deljiv sa tri
   - rekurzivna funkcija proverava da li je broj deljiv sa tri
5. naći n–ti Fibonačijev broj (Fibonačijev niz je niz brojeva u kome zbir prethodna dva broja daju vrednost narednog člana)
   0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…