Funkcije

Ciljevi:

- upoznati se s pojmom funkcija
- upoznati se s tipovima funkcija
- upoznati se s pojmom preopterećenja funkcija
- proučiti mogućnosti korištenja funkcija

Pregled lekcije

Definicija funkcije

Funkcije su izdvojeni dijelovi programskog koda koji imaju svoje ime. Osnovna ideja funkcija je da se neki dio koda napiše na jednom mjestu, a zatim se, po potrebi poziva na više mjesta. Time se skraćuje vrijeme pisanja cjelokupnog programa.

Funkcija se sastoji od nekoliko elemenata:

Povratni tip podataka - definira tip podataka koji će funkcija vratiti nakon što se završi njeno izvođenje.

Ime funkcije – naziv funkcije koji je proizvoljan no mora zadovoljavati pravila ispravnog imenovanja varijabli.

Argumenti ili parametri – mogu se a i ne moraju koristiti. Javljaju se u okruglim zagradama nakon imena funkcije i može ih biti jedan ili više. Argumenti predstvljaju ulazne podatke funkcije a ponašaju se i deklariraju isto kao i varijable. Ulazni parametri su vidljivi unutar funkcije ali ne i izvan nje.

Sintaksa funkcije

```
tip ime (parametri) {
   tijelo funkcije
}
```

Kao što vidimo iz sintakse funkcije, ona se sastoji od:

tip - Bilo koji elementarni tip podataka, pokazivač ili slog.
ime - Mora zadovoljavati pravila imenovanja varijabli.
parametri - Parametri se deklariraju isto kao i varijable u programu.
***tip ime* (*parametri*)** – Cijeli ovaj dio se naziva zaglavlje ili glava funkcije.
tijelo funkcije - Ispravan C++ kod koji u svakom slučaju treba završiti naredbom *return* (osim kada je funkcija tipa void) i vratiti podatak tipa koji je definiran povratnim tipom funkcije.

Funkcija treba biti deklarirana prije prvog korištenja u programu.

Napomena: I glavni je program funkcija koja vraća podatak tipa int.

Tipovi funkcija i primjeri

S obzirom na do sada opisane pojmove, izradit ćemo sljedeći zadatak. Treba napisati program koji od korisnika traži unos jednog cijelog broja od 2 do 5. Ukoliko broj nije unutar zadanog raspona ispisati poruku o tome i tražiti ponovni unos broja. Program potom ispisuje uneseni broj uvećan za 7. Rješenje treba izraditi koristeći funkciju.

Primjer 1:
ULAZ: Unijeti cijeli broj N između 2 i 5
IZLAZ: Ispisati brojeve od N do 10

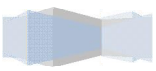
1. Sve dok je N < 2 ili N > 10
2. Učitaj N
3. Ispiši broj N+7

```
#include <iostream>
using namespace std;

int uvecaj(int A)
{
   int rez;
   rez = A + 7;
   return rez;
}

int main () {
    int N;
    do{
        cout << "Unesite broj N: ";
        cin >> N;
    }
    while(N < 2 || N > 10);
    cout << uvecaj(N) << endl;
    system ("PAUSE");
    return 0;
}
```

U navedenom primjeru, odmah prije funkcije main, definirali smo funkciju uvecaj. Funkcija prima kao ulaz jedan argument tipa int. Funkcija ulazni argument uvećava za 7 i taj zbroj vraća kao svoj rezultat pomoću naredbe return. U glavnoj main funkciji pomoću do…while iteracije tražimo unos cijelog broja sve dok on ne bude unutar traženog raspona. Ukoliko je uneseni broj unutar raspona ispisujemo na ekran povratnu vrijednost funkcije uvecaj. Možemo primijetiti da ovom slučaju direktno u ispisu koristimo poziv funkcije. Isto smo mogli riješiti i tako da smo rezultat funkcije prvo pospremili u neku varijablu i potom ispisali samu varijablu na ekran.



```
…
int rez;
```

```
rez = uvecaj(N)
cout << rez << endl;
…
```

Argumenti funkcije se unutar funkcije ponašaju kao normalne varijable. Bilo kakve promjene vrijednosti argumenata tijekom izvođenja funkcije neće se reflektirati na vrijednosti varijable van funkcije. Argument je dakle vidljiv I dohvatljiv samo na razini funkcije tj. unutar funkcije. Promjena vrijednosti ulaznog argument unutar funkcije nema za posljedicu promjenu ulazne varijable koja je proslijeđena kao argument u glavnom programu (main funkciji).

Uzmimo sada sljedeći zadatak kao primjer.

Treba napisati program koji od korisnika traži unos jednog cijelog broja od 2 do 5. Ukoliko broj nije unutar zadanog raspona ispisati poruku o tome i tražiti ponovni unos broja. Program potom ispisuje brojeve od unesenog broja do 10. Rješenje treba izraditi koristeći funkciju.

Primjer 2:

ULAZ: Unijeti cijeli broj N između 2 i 5
IZLAZ: Ispisati brojeve od N do 10

1. Sve dok je N < 2 ili N > 10
2. Učitaj N
3. Ispiši brojeve od N do 10

```
#include <iostream>
using namespace std;

void ispis(int A)
{
   for (int i = A; i <= 10; i++) {
        cout << i << endl;
   }
}

int main () {
    int N;
    do{
        cout << "Unesite broj N: ";
        cin >> N;
    }
    while(N < 2 || N > 10);
    ispis(N);
    system ("PAUSE");
    return 0;
}
```

U ovom primjeru imamo slučaj da funkcija koju koristimo za ispis brojeva ne treba ništa vratiti funkciji iz koje je pozvana. Kako deklarirati funkciju koja ništa ne vraća? Postoji poseban tip void koji se ne smije koristiti kao tip za deklaraciju varijabli, već samo kao povratni tip funkcije i kao tip za deklaraciju pokazivača. Tip void predstavlja izostanak povratnog tipa. Funkcija tipa void u svom tijelu ne treba sadržavati naredbu return.

Uzmimo sada sljedeći zadatak kao primjer.

Treba napisati program koji od korisnika traži unos 10 cijelih brojeva u polje. Program potom ispisuje sve unesene brojeve. Rješenje treba izraditi koristeći funkciju.

S obzirom da prosljeđivanje polja kao argumenta funkciji nije predmet ove lekcije pokazat ćemo kako možemo riješiti navedeni problem koristeći globalne varijable tj. polje na globalnoj razini. Varijable vidljive iz svih dijelova programa nazivaju se globalne varijable. Dakle, primjerice ukoliko unutar funkcije deklariramo neku varijablu, ona je tada vidljiva samo unutar te funkcije. Ukoliko želimo da varijabla bude dostupna globalno u svim dijelovima programa tada je moramo deklarirati globalno u deklaracijskom dijelu programa, prije definiranja prve funkcije.

Dakle, rješenje našeg problema bi glasilo ovako:

Primjer 3:
ULAZ: Unijeti polje od 10 cijelih brojeva
IZLAZ: Ispisati brojeve polja

1. Učitaj polje A
2. Ispiši polje A

```
#include <iostream>
using namespace std;

int A[10];

void ispis()
{
   for (int i = 0; i <= 9; i++) {
        cout << "A[" << i << "] = " << A[i] << endl;
   }
}

int main () {
    for (int i = 0; i <= 9; i++) {
        cout << "Unesite broj " << i+1 << ".: ";
        cin >> A[i];
    }
    ispis();
    system ("PAUSE");
    return 0;
}
```

Pogledajmo sada sljedeći primjer.

Treba napisati program koji od korisnika traži unos jednog cijelog broja. Program potom ispisuje uneseni broj uvećan za 1. Nakon toga se od korisnika traži unos još jednog cijelog broja te se ispisuje zbroj prvog i drugog unesenog broja. Sva rješenja realizirati pomoću funkcija.

Primjer 4:

ULAZ: Unos dva cijela broja A i B

IZLAZ: Ispisati A+1 i A+B

1. Učitaj A
2. Učitaj B
3. Ispiši A+1
4. Ispiši A+B

```
#include <iostream>
using namespace std;

int A[10];

int ispis(int A)
{
  return ++A;
}

int ispis(int A, int B)
{
  return A+B;
}

int main () {
    int A, B;
    cout << "Unesite cijeli broj A: ";
    cin >> A;
    ispis(A);

    cout << "Unesite cijeli broj B: ";
    cin >> B;

    cout << "A+1 iznosi " << ispis(A) << endl;
    cout << "A+B iznosi " << ispis(A,B) << endl;

    system ("PAUSE");
    return 0;
}
```

Primijetimo da u ovom programu postoje dvije funkcije koje se jednako zovu ali imaju različit broj argumenata. U programu može postojati više istoimenih funkcija (osim funkcije main), ali se one moraju razlikovati po broju ili tipu argumenata. Ovo se svojstvo se naziva preopterećenje funkcija (overloading). Ono što također možemo primijetiti jest da u funkcijama izvodimo tražene operacije istovremeno kada ih vraćamo kao povratnu vrijednost funkcije, umjesto da prvo rezultat pospremimo u neku varijablu pa tu varijablu vraćamo naredbom return.

Zadaci za vježbu

1. Odgovorite na pitanje što je to funkcija i čemu služi.
2. Odgovorite na pitanje kakva je to funkcija tipa void.
3. Mogu li postojati dvije ili više funkcija s istima nazivom? Ako da, pod kojim uvjetima?
4. Izradite program koji će pomoću funkcije vršiti množenje i oduzimanje dva broja.

Programski primjeri za laboratorijske vježbe

Zadatak 1.

Izradite program u kojem:

1. Se unose duljine stranica pravokutnika pa se pomoću funkcija računa i ispisuje površina i opseg.

2. Korisnik unosi stranicu a i dijagonale d1 i d2 romba. Program zatim računa površinu i opseg romba koristeći preopterećenje funkcija. Korisnik potom unosi stranicu a kocke, a program računa pomoću funkcija dijagonalu, površinu i volumen kocke.

3. Korisnik upisuje prirodni broj, a računalo ga ispisuje slovima. Za pretvaranje broja u riječ koristiti funkcije.

4. Učitava riječ s tipkovnice duljine N slova, te pokreće fukciju u kojoj se ispisuju sve riječi koje se mogu napraviti od ovih slova. Pri tome se misli na riječi od 2 pa do N slova. Isto se unešeno slovo ne smije koristiti više puta. No, dva unešena slova mogu biti jednaka.

5. Definiramo funkcije za
   a. Unos točke u ravnini. Točka se sastoji od dvaju koordinata tipa float.
   b. Funkcije za izračun udaljenosti dvaju brojeva u ravnini

   Program treba sa tipkovnice učitati prirodni broj N i N točaka, te vratiti dvije od unesenih točaka koje su najbliže jedna drugoj i njihovu udaljenost.