Funkcije

Funkcije

- Nakon odslušanog bit ćete u stanju:
  - objasniti što je funkcija
  - navesti primjere funkcija koje
    - vraćaju jednu vrijednost (naredba `return`)
    - ne vraćaju nikakvu vrijednost
  - razlikovati definiciju od deklaracije funkcije.

Definicija

- **Funkcija** je programska cjelina koja uzima neke podatke, izvršava određeni niz naredbi i vraća rezultat svog izvršavanja.

- Definicija funkcije ima oblik

```
tip_p ime_funkcije(tip_1 arg_1, ..., tip_n arg_n)
{
    tijelo funkcije
}
```

Definicija (2)

- `tip_p` – tip podatka koji će funkcija vratiti kao rezultat svog izvršavanja.
- Unutar zagrada iza imena funkcije nalazi se deklaracija formalnih argumenata. Prvi argument arg_1 je varijabla tipa `tip_1` itd.
- Unutar vitičastih zagrada nalazi se tijelo funkcije koje se sastoji od deklaracija varijabli i naredbi koje se unutar funkcije izvršavaju.
- Ukoliko `tip_p` nije naveden, prevodilac pretpostavlja da funkcija vraća podatak tipa `int`.

Naredba `return`

- Funkcija vraća rezultat svog izvršavanja pomoću naredbe `return`. Opći oblik te naredbe je

  `return` izraz;

- Izraz se može staviti u oble zagrade, ali to nije nužno.
- Funkcija može vratiti aritmetički tip, strukturu, uniju ili pokazivač, ali ne može vratiti drugu funkciju ili polje.

Primjer: izraz kao stvarni argument

- Funkcija `sqrt` iz datoteke `math.h`

  ```
  double sqrt(double);
  ```

  može biti pozvana na sljedeći način:

  ```
  double x, y;
  ......
  y = sqrt(2 * x - 3.);
  ```

- Tada y poprima vrijednost $\sqrt{2x-3}$.

Primjer:

- Sljedeća funkcija mala slova engleske abecede pretvara u velika.

```
char malo_u_veliko(char z)
{
    char c;
    c = (z >=  'a' && z <= 'z') ? ('A' + z - 'a') : z;
    return c;
}
```

- Poziv funkcije:

```
veliko = malo_u_veliko(slovo);
```

Primjer:

- Što radi sljedeća funkcija?

```
int f(int a, int b)
{
    return a > b ? a : b;
}
```

- Poziv funkcije:

```
rezultat = f(a, b);
```

Funkcija tipa `void`

- Kada funkcija ne vraća nikakvu vrijednost onda se za tip „vraćene” vrijednosti koristi riječ `void`.

Primjer:

```
void maksimum(int x, int y)
{
    int max;
    if(x > y) max = x; else max = y;
    printf("Maksimalna vrijednost je %d.\n", max);
    return;
}
```

Funkcija bez argumenata

```
tip_podatka f(void)
{
        tijelo funkcije
}
```

- Ključna riječ `void` unutar zagrada označava da funkcija ne uzima argumente.
- Poziv funkcije:

  varijabla = f( );

- Zagrade su obavezne. One informiraju prevodilac da je simbol f ime funkcije.

Deklaracija funkcije

- Svaka bi funkcija prije svoga poziva u programu trebala biti deklarirana.
- Deklaracija funkcije govori koliko argumenata funkcija uzima, koji su njihovi tipovi i koji tip podatka funkcija vraća.
- Ako je funkcija definirana u istoj datoteci u kojoj se poziva i to prije svog prvog poziva, onda definicija služi kao deklaracija te posebna deklaracija nije potrebna.

Primjer: deklaracija nije potrebna

```
#include <stdio.h>
void maksimum(double x, double y)
{
    double max;
    if(x > y) max = x; else max = y;
    printf("Maksimalna vrijednost je %g.\n", max);
    return;
}
int main(){
    double x, y;
    printf("Unesite dva realna broja: ");
    scanf("%lg %lg", &x, &y);
    maksimum(x, y);
    return 0;
}
```

Deklaracija funkcije (2)

- Ukoliko je funkcija koju koristimo definirana iza mjesta na moramo ju na početku datoteke deklarirati.

- Deklaracija (prototip) funkcije ima oblik:
  `tip_p` ime_funkcije(`tip_1` arg_1, ..., `tip_n` arg_n);

- Imena argumenata mogu biti izostavljena:
  `tip_p` ime_funkcije(`tip_1`, ..., `tip_n`);

Primjer: deklaracija potrebna

```
#include <stdio.h>
int main(){
    double x, y;
    void maksimum(double x, double y);
    printf("Unesite dva realna broja: ");
    scanf("%lg %lg", &x, &y);
    maksimum(x, y);
    return 0;
}
void maksimum(double x, double y)
{
    double max;
    if(x > y) max = x; else max = y;
    printf("Maksimalna vrijednost je %g.\n", max);
    return;
}
```