Funkcije

Funkcije smo do sedaj samo klicali (`printf`, `scanf`, matematične funkcije, funkcije za delo z nizi, ...), v vsakem programu pa smo po eno tudi definirali (`main`). V programu lahko definiramo tudi več funkcij, ki jih potem kličemo eno iz druge. Funkcija `main` je posebna, saj njo pokliče že sistem (vsak program se prične izvajati v tej funkciji).

Funkcije definiramo eno za drugo, pri tem pa moramo paziti, da je vsaka funkcija definirana (ali vsaj deklarirana) pred prvim klicem. Le tako bo prevajalnik vedel, kakšne vrste parametrov moramo podati pri klicu funkcije in kakšne vrste rezultat funkcija vrača.

Deklaracija funkcije je sestavljena iz tipa rezultata, imena funkcije in morebitnih parametrov. Deklaracijo zaključimo s podpičjem. Posamezne parametre ločimo z vejico, vsakega od njih pa opišemo s tipom in imenom (imena parametrov so pri deklaracijah odveč). Deklaraciji funkcije rečemo tudi *prototip*.

Če funkcija ne vrača ničesar, potem za tip rezultata napišemo `void`. Če funkcija ne dobi nobenih parametrov, potem za parametre napišemo `void`.

Funkcija, ki bi izračunala vrednost binomskega simbola *n* nad *k*, bi za parametra dobila dve celi števili, za rezultat pa bi vrnila spet celo število:

```
int binom(int n, int k);
```

Definicija funkcije se prične podobno kot deklaracija, le da ne zaključimo s podpičjem, pač pa nadaljujemo s telesom funkcije, zapisanim v zavitih oklepajih. Telo funkcije je poljubno zaporedje stavkov, kot smo jih do sedaj pisali v funkciji `main`. Zadnji stavek v telesu funkcije je običajno stavek `return`, ki vrne vrednost funkcije. Če funkcija ne vrača ničesar, tega stavka na koncu ne pišemo.

```
int binom(int n, int k)
{
   int b = 1;
   for (int i = 1; i <= k; ++i) {
      b *= n - i + 1;
      b /= i;
   }
   return b;
}
```

Parametri se v telesu funkcije obnašajo kot ostale spremenljivke, definirane v funkciji (rečemo jim lokalne spremenljivke). Od njih se razlikujejo samo v tem, da začetnih vrednosti ne dobijo v funkciji, pač pa pri klicu funkcije. Pri klicu funkcije moramo namreč predpisati vrednosti vseh njenih parametrov:

```
int rezultat = binom(10, 6);
int vrednost = binom(8, 3) + 5 * binom(14, 5);
```

Stavek `return` lahko v funkciji uporabimo tudi večkrat. Ko se ta stavek izvede, prekine delovanje funkcije, izvajanje programa pa se nadaljuje na mestu, od koder je bila funkcija poklicana. V spodnji funkciji izračunamo najmanjšo izmed treh podanih vrednosti. Če je `a` manjši od ostalih dveh, ga vrnemo (in končamo). Sicer vemo, da `a` ni najmanjši, in primerjamo samo še ostala dva. Če je `b` manjši od `c`, ga vrnemo (in končamo), sicer pa vemo, da tudi `b` ni najmanjši. Ostane samo še možnost, da je `c` najmanjši.

```
int najmanjsi(int a, int b, int c)
{
   if (a < b && a < c) return a;
   if (b < c) return b;
```

```
   return c;
}
```

Vrednost, ki jo dobi parameter pri klicu funkcije, lahko v funkciji tudi spremenimo. Ker pa se parameter znotraj funkcije obnaša kot lokalna spremenljivka, taka sprememba navzven ni vidna. Poglejmo si primer:

```
int povecaj(int a, int k)
{
   a += k;
   return a;
}
```

Zgornja funkcija poveča vrednost parametra a in vrne povečano vrednost. Funkcijo bi lahko poklicali takole:

```
int main(void)
{
   int x = 5;
   int y = povecaj(x, 3);
   ...
```

Spremenljivka `x` dobi začetno vrednost 5, ob klicu funkcije `povecaj` dobi parameter `a` vrednost 5, parameter `k` pa vrednost 3. V funkciji spremenljivki `a` prištejemo `k`, tako da dobi vrednost 8. To vrednost vrnemo, tako da tudi spremenljivka `y` dobi vrednost 8. Ker sta `x` in `a` dve različni spremenljivki, se sprememba spremenljivke `a` ne pozna na spremenljivki `x`. Cela zgodba ostane enaka tudi v primeru, če spremenljivko `x` preimenujemo v `a`. To lahko storimo, saj sta spremenljivki `x` in `a` definirani v dveh različnih funkcijah, torej sta različni spremenljivki, ki pa imata enako ime.

```
int main(void)
{
   int a = 5;
   int y = povecaj(a, 3);
   ...
```

Tabela kot parameter funkciji

Parametri funkcije so lahko katerihkoli tipov, torej lahko poleg celih in realnih števil za parametre dobimo tudi tabele in nize. Ker so nizi pravzaprav tabele znakov, bomo v nadaljevanju govorili samo o tabelah.

Tabelo kot parameter definiramo podobno kot običajno število, le da ne smemo pozabiti na oglate oklepaje za imenom parametra. Prototip funkcije, ki v tabeli realnih števil poišče dano vrednost, vrne pa indeks v tabeli, kjer se iskana vrednost nahaja, bi lahko izgledal takole:

```
int poisci(double tab[], double x);           // kakšna je dolžina tabele?
```

Vendar pa pri izvedbi te funkcije naletimo na težave, saj ne vemo, koliko elementov je v tabeli (če gre za niz, potem z dolžino ni težav, saj je na koncu niza vedno znak s kodo 0). Rešitvi sta dve. Če vemo, da bodo vse tabele, v katerih bomo iskali, vsebovale enako mnogo elementov, potem lahko (ni pa obvezno) to konstanto zapišemo med oglate oklepaje. Če pa so lahko tabele različnih dolžin, nam ne preostane drugega, kot da število elementov v tabeli podamo kot dodaten parameter:

```
int poisci(double tab[MAX], double x);        // prva možnost
int poisci(double tab[], int n, double x);    // druga možnost
```

Tabelo, ki jo bomo dali za parameter pri klicu funkcije, moramo sestaviti že pred klicem funkcije. Za vrednost parametra potem napišemo samo ime tabele. Izjema so le nizi, ki jih v

funkciji ne spreminjamo (takšnega niza ni treba shraniti v spremenljivko, če ga damo za parameter).

```
double t[] = {1, 5.3, 6.342, -3.5, 8.2};
int p = poisci(t, SIZEOF(t), -3.5);           // SIZEOF izračuna dolžino
tabele
```

Če v funkciji spremenimo katerega od elementov tabele, ki smo jo dobili za parameter, bo ta sprememba vidna tudi navzven. Za primer vzemimo funkcijo, ki zamenja dva elementa v tabeli. Za parameter bo funkcija dobila tabelo in indeksa elementov, ki ju je treba zamenjati, vrnila pa ne bo ničesar, saj do sprememb pride samo v podani tabeli. Dolžine tabele kot dodaten parameter ne potrebujemo.

```
void zamenjaj(int tab[], int i, int j)
{
   int t = tab[i];
   tab[i] = tab[j];
   tab[j] = t;
}
```

Sestavimo še funkcijo za izpis tabele:

```
void izpis(char opis[], int tab[], int n)
{
   printf("%s:", opis);
   for (int i = 0; i < n; ++i) printf(" %d", tab[i]);
   printf("\n");
}
```

Zdaj lahko preverimo delovanje funkcije `zamenjaj`

```
int main(void)
{
   int t[] = {2, 6, 3, 8, 4, 3, 7, 4};
   int n = SIZEOF(t);

   izpis("Prvotna tabela", t, n);
   zamenjaj(t, 1, 6);
   izpis("Po zamenjavi", t, n);
   return 0;
}
```