Tabele

Tabela je podatkovni tip, ki ga uporabimo, kadar želimo skupaj (pod istim imenom) hraniti več podatkov enakega tipa. Tako lahko imamo tabelo, v kateri hranimo samo cela števila, samo realna števila ... Novo tabelo najpreprosteje ustvarimo tako, da naštejemo vse njene elemente:

```
int tab1[] = {1, 5, 0, -4, 2, -3, 5, 0};
double tab2[] = {3.5, 2.341, 5, -4.2, 4.0, 3};
```

Spremenljivka `tab1` je tabela celih števil. Da gre za tabelo in ne samo za eno celo število, smo določili s praznimi oglatimi oklepaji za imenom spremenljivke. Vrednosti, ki jih hranimo v tej tabeli, naštejemo v zavitih oklepajih. Podobno smo naredili tudi pri tabeli `tab2`, le da v njej hranimo realna števila.

V večini primerov pa vrednosti, ki jih bomo imeli v tabeli, ne poznamo vnaprej, ali pa jih je preveč, da bi jih vse našteli. C nam omogoča, da ustvarimo "prazno" tabelo. Pri taki tabeli moramo vnaprej poznati samo njeno velikost (največje možno število elementov, ki jih bomo hranili v tabeli). Takšno tabelo ustvarimo takole:

```
int t1[8];
double t2[6];
```

Elementi tabele so indeksirani od 0 naprej. Prvi element tabele ima indeks 0, drugi element ima indeks 1, zadnji element pa indeks *n* - 1, če je *n* število elementov v tabeli. Do elementa z indeksom `i` v tabeli `tab` pridemo tako, da indeks napišemo v oglatih oklepajih za imenom tabele, torej `tab[i]`. V našem primeru je `tab1[4]` enako 2, `tab2[1]` pa je enako 2.341. Če je `tab` tabela celih števil, potem je `tab[i]` celo število. Še več, `tab[i]` lahko obravnavamo kot celoštevilsko spremenljivko, kar pomeni, da lahko stoji na levi strani prireditvenega operatorja. To uporabljamo, kadar želimo spremeniti vrednost elementa v tabeli. Podobno velja tudi za druge vrste tabel.

```
t1[0] = 3;
t2[4] = 4.5
```

Programski jezik C ne pozna funkcije, ki bi izračunala dolžino (maksimalno število elementov) dane tabele, zato moramo biti pri uporabi tabel zelo pazljivi. Dolžino poznamo takrat, ko tabelo ustvarimo (če naštejemo vse elemente, jih lahko tudi preštejemo, če samo rezerviramo prostor, pa moramo dolžino sami podati).

Pogosto dolžino tabele omejimo z vnaprej definiranim simbolom (konstanto), ki ga definiramo na začetku datoteke (običajno takoj za vključevanjem knjižnic). Najpogosteje tak simbol poimenujemo `MAX` ali kaj podobnega. Lahko definiramo tudi več simbolov za različne tabele. S tem se izognemu večkratnemu pojavljanju istih konstant skozi program. Če se izkaže, da je potrebno povečati dolžino tabele, potem to lahko storimo samo na enem mestu (s spremembo vrednosti simbola).

Koristen je tudi makro ukaz (običajno ga poimenujemo `SIZEOF`), ki zna izračunati dolžino dane tabele (če ga uporabimo v isti funkciji, kjer je definirana tabela). Z uporabo tega makro ukaza se izognemo ročnemu preštevanju elementov v tabeli. Pri definiciji tega ukaza uporabimo operator `sizeof`, ki vrne velikost pomnilnika v zlogih, kolikor ga zaseda neka spremenljivka.

```
#define MAX 100
#define SIZEOF(tab) (sizeof(tab) / sizeof(tab[0]))
```

Z vnosom podatkov v tabelo imamo nekaj več dela kot z vnosom enega samega celega ali realnega števila. Pred vnosom elementov moramo tabelo ustvariti, kar pomeni, da moramo

najprej oceniti, največ koliko elementov bomo v tabeli hranili. Uporabnik bo nato vpisal dejansko število elementov, ki jih želi shraniti v tabelo, nato pa še vsak element posebej. Vnos podatkov v tabelo celih števil izgleda takole:

```
int n;
int tab[MAX];
printf("Vnesi stevilo elementov (med 0 in %d): ", MAX);
scanf("%d", &n);
for (int i = 0; i < n; ++i) {
   printf("%d. element: ", i + 1);
   scanf("%d", &tab[i]);
}
```

Tudi z izpisom tabele je malo več dela kot z izpisom celega ali realnega števila. Izpisati moramo vsak element posebej, kar pomeni, da potrebujemo zanko, znotraj katere izpisujemo po en element tabele. Če vse elemente izpisujemo v isto vrstico (kot v spodnjem primeru), ne smemo na koncu pozabiti zaključiti vrstice.

```
for (int i = 0; i < n; ++i) printf("%d ", tab[i]);
printf("\n");
```

Primer

Elemente dane tabele preštejemo tako, da definiramo pomožno spremenljivko z začetno vrednostjo 0 (trenutna vsota), nato pa ji v zanki prištevamo posamezne elemente tabele. V spodnji kodi je tabeli ime `tab`, seštevamo pa prvih `n` elementov.

```
int vsota = 0;
for (int i = 0; i < n; ++i) vsota += tab[i];
```

Primer

Najmanjši ali največji element tabele najdemo tako, da prvega v tabeli proglasimo za trenutno največjega in najmanšega. Potem pregledamo vse ostale elemente (zato zanko začnemo pri indeksu 1). Če najdemo manjši element od trenutno najmanjšega, si ga zapomnemo. Podobno velja za večjega od trenutno največjega.

```
int min = tab[0];
int max = tab[0];
for (int i = 1; i < n; ++i) {
   if (tab[i] < min) min = tab[i];
   if (tab[i] > max) max = tab[i];
}
```

S tabelami lahko predstavimo vrsto matematičnih objektov, kot so vektorji, permutacije, polinomi ene spremenljivke ...

Pri predstavitvi vektorja v tabelo zapišemo vse njegove koordinate. Število komponent je enako dimenziji prosta, ki mu pripada vektor.

Permutacijo predstavimo s tabelo tako, da za vsako število od 1 do n v tabelo zapišemo, kam se to število preslika. Sliko števila i zapišemo v tabelo na mesto z indeksom i - 1.

Polinom ene spremenljivke pa predstavimo tako, da v tabelo zapišemo njegove koeficiente. Element na mestu z indeksom i je kar koeficient pri x^i. Za zapis polinoma stopnje n potrebujemo tabelo dolžine n + 1. Vodilni koefocient mora biti različen od 0. Polinom 0 obravnavamo kot polinom stopnje -1 (predstavimo ga s prazno tabelo).