Zapiski s priprav

Seznami

ACM priprave na tekmovanja

Različica z dne 12. december 2024

Kazalo

1 Sintaksa

Ko si želimo podatke shraniti tako, da jih bomo lahko spreminjali in na koncu nekaj z njimi naredili (npr. da z njimi računamo, jih izpišemo, itd.), za to uporabimo spremenljivke. Vsaka spremenljivka hrani en podatek – vse spremenljivke do sedaj so hranile le eno številko. Pogosto pa si želimo števila shraniti tako, da bomo kasneje lahko dostopali do njih, ampak med pisanjem programa ne vemo točno, s koliko števili bo program moral delati. Problem rešimo s seznami.

Da ustvarimo seznam (angl. *array*), zapišemo tip spremenljivke, ki ga bodo imeli vsi elementi seznama (trenutno poznamo le `int`, a bomo kmalu spoznali tudi druge tipe spremenljivk), nato seznamu damo ime, in na koncu v oglatih oklepajih zapišemo dolžino seznama, takole:

```
int seznam_stevil[300];
```

Tukaj smo ustvarili seznam z imenom `seznam_stevil`, ki hrani 300 števil. Če želimo dostopati do elementov seznama, ali nastaviti njihove vrednosti, tudi uporabimo oglate oklepaje, kot v spodnjem primeru.

```
#include <stdio.h>

int seznam_stevil[300];

int main() {
    seznam_stevil[3] = 7;
    seznam_stevil[4] = 9;
    seznam_stevil[5] = seznam_stevil[3] + seznam_stevil[4];
    printf("%d\n", seznam_stevil[5]);  // izpiše 16
    return 0;
}
```

Tu smo prvo nastavili tretji element seznama na 7, nato smo nastavili četrti element seznama na 9, in za tem nastavili peti element seznama na njuno vsoto. Ostalih elementov seznama se nismo dotaknili. Tako kot pri spremenljivkah nismo smeli uporabiti vrednosti spremenljivke, preden smo ji vrednost nastavili, moramo paziti, da vrednosti posamičnega elementa seznama ne uporabljamo, preden je ne nastavimo. Narobe bi bilo na primer izpisati `seznam_stevil[2]` ali to vrednost uporabiti v računu, saj je nismo nikoli nastavili.

Pri dostopanju do elementov seznama moramo paziti tudi na naslednje dejstvo: če imamo seznam dolžine N, potem so elementi tega seznama oštevilčeni s številkami od 0 do $N - 1$ vključno, ne pa s številkami od 1 do N, kot bi morda pričakovali.

V zgornjem primeru tako lahko zapišemo `seznam_stevil[0]`, `seznam_stevil[1]`, …, `seznam_stevil[299]`, ne pa tudi `seznam_stevil[300]`. Pravimo, da so seznami *indeksirani* od 0 naprej.

Poglejmo si primer preproste naloge. Na vhodu je podano število N, ki mu sledi N števil. Program mora izpisati ta števila (razen prvega, N), v obratnem vrstnem redu. V ta namen ustvarimo seznam z imenom `seznam`, ki lahko shrani največ 1000 (predpostavimo, da uporabnik ne bo vnesel več kot 1000 števil). Potem s `for` zanko preberemo N števil in vsako shranimo v svoj element seznama, začenši z 0. Na koncu se s še eno `for` zanko sprehodimo skozi $i = 0, 1, \ldots, N-1$. Na vsakem koraku izračunamo indeks (položaj) elementa, ki je i-ti od konca seznama, in ga shranimo v spremenljivko `obratni`. Ta indeks je natanko `N-i-1`, saj mora za `i=0` biti `obratni=N-1`, za `i=1` mora biti `obratni=N-2`, itd. Da se seznam na izhodu izpiše v obratnem vrstnem redu, preprosto izpišemo `seznam[obratni]`.

```c
#include <stdio.h>

int seznam[1000];

int main() {
    int N;
    scanf("%d", &N);

    for (int i = 0; i < N; i++) {
        scanf("%d", &seznam[i]);
    }

    for (int i = 0; i < N; i++) {
        int obratni = N-i-1;
        printf("%d ", seznam[obratni]);
    }
    printf("\n");

    return 0;
}
```

Poglejmo si še malo bolj zanimiv primer. Naša naloga sedaj je, da uredimo seznam N števil ($N \leq 10^6$) po velikosti od najmanjšega do največjega. Podano imamo tudi informacijo, da bodo ta števila velika med vključno 0 in 100. Naloge se lahko lotimo tako, da preštejemo, kolikokrat se neko število pojavi v danem seznamu, nato pa bomo seznam rekonstruirali tako, da bo urejen. Da preštejemo, kolikokrat se kakšno število pojavi, uporabimo nov seznam, kjer indeks pomeni številko, ki jo štejemo, shranjena vrednost pa kolikokrat smo to številko že prešteli.

```c
#include <stdio.h>

// seznam iz vhoda
int seznam[1000003];
// na indeksu j je zapisano, kolikokrat smo že videli
// število j v seznamu
int stetje[101];
// nov, urejen seznam
int novseznam[1000003];

int main() {
    int N;  // velikost seznama
    scanf("%d", &N);
    for (int i = 0; i < N; i++)
        scanf("%d", &seznam[i]);

    // število na mestu i v seznamu je seznam[i]
    // v eni iteraciji zanke vidimo eno število; zato prištejemo
    // 1 na pravo mesto seznama za štetje
    for (int i = 0; i < N; i++)
        stetje[ seznam[i] ] += 1;

    // sedaj rekonstruiramo seznam
    // shranjujemo si indeks prvega elementa v novem seznamu,
    // ki ga še nismo nastavili
    int indeks = 0;
    for (int j = 0; j <= 100; j++) {
        // stetje[j]-krat moramo zapisati j v nov seznam
        for (int k = 0; k < stetje[j]; k++) {
            novseznam[indeks] = j;
            // povečamo indeks, ker smo ga ravno nastavili
            indeks++;
        }
    }
    // izpišemo nov seznam
    for (int i = 0; i < N; i++)
        printf("%d ", novseznam[i]);

    printf("\n");
    return 0;
}
```

Ta algoritem za urejanje je zelo znan; imenuje se urejanje s preštevanjem (angl. *counting sort*). Primeren je, kadar imamo zelo majhen razpon možnih vrednosti števil, kakor smo imeli tu $(0-100)$. Obstajajo tudi drugi algoritmi za urejanje. Nekatere bomo spoznali kasneje.

1.1 Večdimenzionalni seznami

Videli smo, kako ustvariti seznam števil, kaj pa seznam seznamov? Takemu seznamu pravimo *dvodimenzionalen seznam*, ustvarimo pa ga tako, da napišemo dva zaporedna oglata oklepaja z velikostjo, kot spodaj:

```
1 int seznam_seznamov_stevil[velikost1][velikost2];
```

Dvodimenzionalni seznami so uporabni, kadar moramo podatke predstaviti v tabeli. Do posamičnih elementov dostopamo z dvojnimi oglatimi oklepaji, tako kot pri inicializaciji spremenljivke; `tabela[i][j]`. Tabele si običajno predstavljamo tako, da nam prvi indeks poda zaporedno številko vrstice, drugi pa zaporedno številko stolpca. Sicer pa z njimi delamo enako kot z običajnimi seznami.