Slogovi i unije

Ciljevi:

- upoznati se sa slogovima i unijama
- shvatiti način rada slogova i unija
- uvidjeti zašto i kako koristiti slogove i unije
- naučiti na koji način organizirati podatke kroz slogove i unije

Pregled lekcije

Definicija sloga

Slog (struktura) je mehanizam agregacije u kojem se povezuje više podataka koji mogu imati međusobno različite tipove, pri čemu se svaki podatak unutar samoga sloga naziva komponenta sloga a komponentama sloga se pristupa preko identifikatora sloga odnosno imena sloga.

Sintaksa:

```
struct ime_strukture {
       opis_komponenti
} deklaracije varijabli;
```

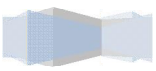
Komponente strukture se deklariraju po istim pravilima po kojima se deklariraju varijable. Iza zatvorene vitičaste zagrade navode se identifikatori (imena) varijabli koje će kao tip podataka imati složeni tip definiran strukturom. Ime strukture koristi se kao ime tipa podataka kojim će se kasnije u programu moći deklarirati varijable.

Napomena: Ime sloga je opcionalno. Ako se ne navede, reći ćemo da je struktura neimenovana i neće se moći koristiti za kasniju deklaraciju varijabli. Isto tako, deklaracije varijabli su opcionalne. Varijable se mogu deklarirati i kasnije u programu, no tada slog mora biti imenovan.

Primjena sloga

Pretpostavimo da želimo izraditi algoritam za slijedeći zadatak:

- ULAZ: N osoba (najviše 50). Za svaku osobu se unosi šifra, prezime i ime, spol, težina i visina.
- IZLAZ: Za svaku osobu treba izračunati omjer težine i visine (relativnu težinu), te ispisati sve osobe koje su iznad prosjeka. Pri tome treba odvojiti osobe prema spolu.



Naime, razmislimo li malo o ovom primjeru vidjet ćemo da nailazimo na određene probleme s kojima se još nismo susretali.

Problem koji ovdje imamo jest u tome što su podaci koji se pohranjuju za pojedinu osobu različitih tipova podataka. Stoga nije moguće, kao u nekim ranijim slučajevima riješiti ovo pomoću dodavanja dodatnih dimenzija u polje. Mogli bismo generirati više polja, po jedno za svaki podatak, no to bi nam znatno otežalo rad i zakompliciralo obradu.

Dakle, u ovom slučaju da bismo što efikasnije riješili problem uporabom koristit čemo slog pri čemu je potrebno organizirati podatke unutar sloga na svrsishodan način.

Primjer 1:

ULAZ: N osoba (najviše 50). Za svaku osobu se unosi šifra, prezime i ime, spol, težina i visina
IZLAZ: Za svaku osobu treba izračunati omjer težine i visine (relativnu težinu), te ispisati sve osobe koje su iznad prosjeka. Pri tome treba odvojiti osobe prema spolu.

1. Upiši N
2. SZ = 0, SM = 0, NZ = 0, NM = 0
3. Za I = 0..N-1
4. Upiši prezime i ime i-te osobe
5. Upiši spol i-te osobe
6. Upiši visinu i-te osobe
7. Upiši težinu i-te osobe
8. Ako je spol ženski
9. SZ = SZ + težina/visina
10. NZ = NZ + 1
11. inače
12. SM = SM + težina/visina
13. NM = NM + 1
14. SZ = SZ / NZ
15. SM = SM / NM
16. Za i = 0..N-1 radi
17. Ako je spol ženski i ako je težina/visina > SZ
18. ispiši prezime i ime, visinu, težinu
19. inače
20. Ako je težina/visina > SM
21. ispiši prezime i ime, visinu, težinu

Dakle, prvo što možemo odmah pretvoriti u C++ kod bez dodatnih razmatranje je uključivanje standardnih biblioteka zaglavlja te prostora funkcijskih imena. Isto tako pseudo kod stavljamo u **`main`** funkciju i sada nadalje razrađujemo kod.

```
#include <iostream>
using namespace std;
int main () {
```

1. Upiši N
2. SZ = 0, SM = 0, NZ = 0, NM = 0
3. Za I = 0..N-1
4. Upiši prezime i ime i-te osobe
5. Upiši spol i-te osobe
6. Upiši visinu i-te osobe

```
7.          Upiši težinu i-te osobe
8.          Ako je spol ženski
9.                  SZ = SZ + težina/visina
10.                 NZ = NZ + 1
11.         inače
12.                 SM = SM + težina/visina
13.                 NM = NM  + 1
14. . . .
```

Prije nego li počnemo s daljnjom pretvorbom pseudo koda u C++ kod definirajmo strukturu podataka nad kojom će algoritam raditi.

Organizacija podataka će biti takva da će osobe biti smještene u polje a svaki element polja mora sadržavati:

- prezime i ime
- spol
- visinu
- težinu

Dakle element polja bit će tipa sloga

```
struct {
        char preime[40];
        char spol;
        int visina;
        int tezina;
};
```

a nad ovom strukturom ćemo definirati polje od 50 osoba i to na sljedeći način:

```
struct {
        char preime[40];
        char spol;
        int visina;
        int tezina;
} osoba[50];
```

Uz do sada definirano bit će nam potrebno i:

- Cjelobrojna varijabla N koja će sadržavati broj osoba koje treba unijeti,
- Cjelobrojne varijable NZ i NM u kojima ćemo pobrojavati muškarce i žene, kako bismo mogli izračunati prosjek relativnih težina oba spola,
- Decimalne varijable SZ i SM u kojima ćemo izračunavati sumu kvocijenta težina i visina i na kraju izračunati prosječnu relativnu težinu oba spola.

Ukomponirajmo da sada učinjeno u C++ kod:

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

```
int main () {
    struct {
            char preime[40];
            char spol;
            int visina, tezina;
    } osobe [50];

    int N;
    float SM = 0, SZ = 0;
    int NZ = 0, NM = 0;
}
```

Nakon toga prelazimo na prvi dio jezgre algoritma koji ćemo izraditi na sljedeći način.

```
    do {
       cout << "N = ";
       cin >> N;
    } while (N < 1 || N > 50);

    for (int i = 0; i < N; i++) {
        cout << "Prezime i ime: ";
        cin.ignore();
        cin.getline(osobe[i].preime,40);
        do {
           cout << "Spol: ";
           osobe[i].spol = cin.get();
           cin.ignore();
        } while (osobe[i].spol != 'Z' &&
                 osobe[i].spol != 'z' &&
                 osobe[i].spol != 'M' &&
                 osobe[i].spol != 'm');
```

Primijetite da smo prije korištenja `cin.getline` funkcije koristili `cin.ignore` kako bismo ispraznili spremnik od prethodne `cin` naredbe. Isto tako kod unošenja spola smo trebali samo jedan znak te smo s obzirom na to koristili `cin.get` ali nismo brisali spremnik pošto nije bilo potrebno jer se prije ove funkcije koristila funkcija `cin.getline`. No imajte na umu da će nam za sljedeći unos biti potrebno obrisati spremnik tako da smo opet nakon ove linije koda koristili `cin.ignore` za brisanje spremnika kako bi spremnik bio prazan kod sljedećeg unosa.

```
        cout << "Visina: ";
        cin >> osobe[i].visina;
        cout << "Tezina: ";
        cin >> osobe[i].tezina;
             if (osobe[i].spol == 'z' || osobe[i].spol == 'Z')
{
           SZ += ((float)osobe[i].tezina/osobe[i].visina);
           NZ++;
        }
        else {
```

```
            SM += ((float)osobe[i].tezina/osobe[i].visina);
            NM++;
        }
    }
```

I za kraj još radimo provjeru spola te ispis potrebnih podataka.

```
    SZ /= NZ;
    SM /= NM;
    for (int i=0; i<N; i++)
        if (osobe[i].spol == 'z' || osobe[i].spol == 'Z' ?
            (float)osobe[i].tezina/osobe[i].visina > SZ :
            (float)osobe[i].tezina/osobe[i].visina > SM)
            cout << osobe[i].preime << " "
                 << osobe[i].tezina << " "
                 << osobe[i].visina << endl;

    system("pause");

    return 0;
}
```

Jedino što je ovdje možda nejasno je ternarni kondicional koji radi na principu provjere istinitosti tvrdnje prije znaka '?' i s obzirom na ishod skače na tvrdnju lijevo odnosno desno od znaka ':'. Nakon toga se još provjerava istinitost IF selekcije imajući na umu ishod ternarnog kondicionala.

Definicija unije

U programskom jeziku C++ postoji mehanizam agregacije koji omogućuje da se alternativni podaci čuvaju u istom memorijskom prostoru. Taj se mehanizam naziva unija. Deklaracija unije u potpunosti je jednaka po sintaksi i svojstvima kao i deklaracija strukture, osim u ključnoj riječi union koja se koristi umjesto ključne riječi struct.

Sintaksa:

```
union ime unije {
          alternativni podaci
} identifikatori varijabli;
```

Ilustrirajmo uporabu unije na sljedećem primjeru evidencije nastavnika i studenata.

Primjer 2:

ULAZ: N osoba (najviše 1000). Osobe koje su unesene su nastavnici i studenti. Sve osobe imaju sljedeće podatke

- Identifikator (različit za nastavnike i studente, ali u svakom slučaju broj)
- Prezime
- Ime

Studenti imaju još i sljedeće podatke:

- Godina studija
- Broj položenih ispita
- Prosječna ocjena

Nastavnici imaju sljedeće podatke:

- Nastavno zvanje
- Broj sati nastave

IZLAZ: Program mora moći raditi dvije stvari:

- ispisivati studente koji imaju prosjek veći od zadanog
- Ispisivati sve nastavnike i njihovu razliku od norme. Pri tome su norme sljedeće:
  - Asistent i viši asistent 150 sati
  - Docent, Izvanredni profesor i Redoviti profesor 300 sati
  - Stručni suradnik 600 sati
  - Predavač i viši predavač 450 sati

Jasno je da postoje dvije vrste osoba i da svaka vrsta ima drugačije podatke stoga definirajmo najprije strukture koje opisuju posebne podatke koje imaju samo studenti i samo nastavnici a nakon toga ćemo dati cjelokupnu organizaciju podataka.

```
struct nastavnik {
        char zvanje[20];
        int brsati;
};

struct student {
        int godstud;
        float prosjek;
};
```

Dakle, za početak smo definirali dva sloga, po jedan za svaki tip osobe s pripadajućim komponentama. Slog nastavnik se sastoji od varijable tipa **`int`** u koju će biti spreman broj sati te znakovnog niza od dvadeset elemenata dok je slog student definiran kroz dvije varijable u koje se sprema godina studija (**`int`**) i prosjek (**`float`**).

U glavnu strukturu potrebno je dodati podatak koji će određivati je li osoba nastavnik ili student. To ćemo definirati kao podatak tipa bool. Prema tome ovisno o tome da li je osoba student ili nastavnik, potrebno je, osim općih podataka pohraniti i posebne podatke. Mogli bismo svakoj osobi dodati svaki od podataka, no time bismo bespotrebno trošili memorijski prostor. Stoga ćemo spojiti posebne podatke u uniju.

```
union pod {
      nastavnik pod_nas;
      student pod_stud;
};
```

Sada ćemo još deklarirati strukturu za jednu osobu.

```
struct osoba {
        int id;
        char ime[20], prezime[20];
        short god_rod;
        bool student;
        pod podaci;
};
```

I na kraju ćemo još deklarirati glavnu strukturu koja će sadržavati polje osoba te broj osoba u polju.

```
struct osobe {
        osoba p[1000];
        int broj;
};
```

Sada kada smo definirali na koji će način podaci biti organizirani razradimo pseudo kod našeg algoritamskog rješenja. Globalno bi situacija bila sljedeća:

ULAZ: N osoba (najviše 1000). Osobe koje su unesene su nastavnici i studenti.
IZLAZ: Program ispisuje studente koji imaju prosjek veći od zadanog te sve nastavnike i njihovu razliku od norme.

1. Učitaj osobe
2. radi
3. Ispiši izbornik
4. Učitaj izbor
5. Ako su izabrani nastavnici
6. ispiši razliku od norme
7. inače
8. Učitaj prosjek
9. Ispiši studente kojima je prosjek veći ili jednak zadanom.
10. dok nije unesena 0

Razradimo li ovaj pseudo kod detaljnije dobit ćemo sljedeću situaciju.

1. Učitaj N
2. Za i=0..N-1
3. Učitaj identifikator
4. Učitaj ime
5. Učitaj prezime
6. Učitaj student
7. Ako je student = true
8. Učitaj godinu
9. Učitaj prosjek
10. inače
11. Učitaj zvanje

12. Učitaj satnicu
13. Radi
14. Ispiši izbornik
15. Učitaj izbor
16. Ako su izabrani nastavnici
17. ispiši razliku od norme
18. inače
19. Učitaj prosjek
20. Ispiši studente kojima je prosjek veći ili jednak zadanom.
21. dok nije unešena 0

Sada možemo prevesti algoritma u C++ kod. Kao i uvijek krećemo s uključivanje biblioteka zaglavlja, prostora funkcijskih imena, definiranjem struktura i svega ostalog potrebnog za adekvatno funkcioniranje našeg algoritma i nakon toga slijedi jezgra programa.

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

struct nastavnik {
        char zvanje[20];
        int brsati;
};

struct student {
        int godstud;
        float prosjek;
};

union pod {
       nastavnik pod_nas;
       student pod_stud;
};

struct osoba {
        int id;
        char ime[20], prezime[20];
        short god_rod;
        bool student;
        pod podaci;
};

struct osobe {
        osoba p[1000];
        int broj;
};

int main () {

    int N;
    osobe o;
```

```
do {
   cout << "N = ";
   cin >> N;
} while (N<1 || N >1000);

o.broj = N;

for (int i=0; i<o.broj; i++) {

    cout << "Indentifikator: ";
    cin >> o.p[i].id;
    cout << "Prezime: ";
    cin.ignore();
    cin.getline(o.p[i].prezime,20);
    cout << "Ime: ";
    cin.getline(o.p[i].ime,20);
    cout << "Student (d/n): ";

    char s;

    do {
       s = cin.get();
    } while (s != 'd' && s != 'D' &&
             s != 'n' && s != 'N');

    if (s == 'd' || s == 'D')
       o.p[i].student = true;
    else
        o.p[i].student = false;

    if (o.p[i].student) {
       cout << "Godina studija: ";
       cin >> o.p[i].podaci.pod_stud.godstud;
       cout << "Prosjek: ";
       cin >> o.p[i].podaci.pod_stud.prosjek;
    }
    else {
         cout << "Zvanje: ";
         cin >> o.p[i].podaci.pod_nas.zvanje;
         cout << "Satnica: ";
         cin >> o.p[i].podaci.pod_nas.brsati;
    }
}

int izbor;
do {
   cout << "1. Prosjek" << endl;
   cout << "2. Norme" << endl;
   cout << "0. Izlaz" << endl;
   cout << endl << "Izbor: ";
```

```
        cin >> izbor;

    if(izbor == 0)
    break;

     if (izbor == 2) {
                for (int i = 0; i< o.broj; i++) {
                int norma;
                char zv[20];
                strcpy(zv, o.p[i].podaci.pod_nas.zvanje);
                int bs = o.p[i].podaci.pod_nas.brsati;
                if (!o.p[i].student)
                {
                   if (!strcmp(zv,"prof") ||
                       !strcmp(zv, "doc"))
                      norma = 300;
                   if (!strcmp(zv,"asistent"))
                      norma = 150;
                   if (!strcmp(zv,"predavac"))
                      norma = 450;
                   if (!strcmp(zv,"suradnik"))
                      norma = 600;
                   cout << o.p[i].prezime << " "
                        << o.p[i].ime << " "
                        << norma - bs << endl;
                        }
            }
       }
       else {
       float prosjek;
       cout << "Prosjek: ";
       cin >> prosjek;
       for (int i=0; i<o.broj; i++)
           if (o.p[i].student &&
               o.p[i].podaci.pod_stud.prosjek >=
               prosjek)
              cout << o.p[i].ime << " "
                   << o.p[i].prezime << " " <<
                   o.p[i].podaci.pod_stud.prosjek
                   << endl;

       }

    } while (izbor != 0);

    system("pause");

    return 0;

}
```

Zadaci za vježbu

- Objasnite što je to slog a što je unija.
- Koja je sintaksa sloga a koje unije?
- Koja je uloga unije?
- Izradite slijedeći algoritam uz pomoć slogova:
  ULAZ: N studenata. Za svakog studenta upisuju se sljedeći podaci:
  - Broj indeksa
  - Prezime
  - Ime
  - Godina studija
  - Polje s ocjenama

  IZLAZ: Poredati studente od prve do četvrte godine, a unutar godine silazno po prosječnoj ocjeni.

Programski primjeri za laboratorijske vježbe

Zadatak 1.

Izradite program u kojem:

1. Se računa površina plašta valjka s time da su i ulazni ali i izlazni podaci spremljeni u proizvoljan slog. Površina plašta valjka se računa po formuli $P = 2r\pi v$. Nakon izračuna rezultat se treba ispisati na zaslon.

2. Je potrebno pronaći brojeve čiji umnožak sa svakim od ASCII kodova znakova 'z','A','g','h','N' daje rezultat iz prethodne točke. Svaki dobiveni broj ali i sve ostale podatke spremati u isti slog te brojeve uzlazno sortirati preko sortiranja umetanjem. Dobivene brojeve kao i sortirano jednodimenzionalno polje ispisati na zaslon.

3. Se unose podaci za: ime, prezime, matični broj, datum upisa na studij i predmeti za sve studente jedne godine (najviše 50). Nakon unosa svih podataka potrebno je izračunati prosjek ocjena za svakog studenta te ih ispisati na zaslon, zajedno s pripadajućim prosjekom te ostalim poznatim podacima, studenta s najmanjim i najvećim prosjekom. Algoritam realizirati pomoću sloga. Koristiti metodu sortiranja pomoću umetanja.

4. Se filtriraju svi studenti iz prethodne točke čija imena su duža od 3 a prezimena kraća od 7 znakova. Nakon toga se ti studenti sortiraju s obzirom na ime od manjeg znakovnog niza k većem. Pri tome je potrebno koristiti metodu mjehuričastog sortiranja. Sada se broj znakova u svakom drugom zapisu imena zbraja a u ostalim oduzima. Prethodna suma daje pomak s obzirom na koji je potrebno kriptirati znakove iz 2. točke Cezarovom šifrom s pomakom koji se unosi s tipkovnice. Na zaslon je potrebno ispisati filtrirane studente, sortirani niz studenata, kriptografski pomak te originalne i kriptirane znakove.

5. Je algoritam iz 3. točke modificiran da radi i za učenike i za studente s time da su zajednički atributi ime, prezime i matični broj. Uz ove atribute studenti još imaju broj indeksa, predstavnik studenata (da/ne), smještaj u studentskom domu (da/ne) te potpora ministarstva (da/ne) dok učenici imaju dužinu školovanja (3 ili 4 godine), grad u kojem se školuju te mjesto završene osnovne škole. Rješenje mora biti realizirano sa slogom i unijom. Uz obradu koja se dešava u 3. točki mora se ponuditi i uzlazno i silazno sortiranje kod ispisa svih studenata odnosno učenika. Algoritam mora biti neovisan s obzirom na točku 3.