Податоци од тип структура

Во сите програми кои ги разгледавме досега, работевме со променливи од само еден податочен тип. Тука ги вклучуваме и низите, кои претставуваат множества на податоци од еден ист тип. Но, често се случува да имаме потреба од работа со групи податоци кои не се од ист тип - на пример, сакаме да чуваме податоци кои ќе опишуваат една, или повеќе, книги. Ваквата структура (книга) треба да содржи информации за името на книгата, авторот, бројот на страници, датумот на издавање, итн.

Во C++, доколку програмата треба да работи со само една книга и сите информации се процесираат во само една функција, податоците може да ги чуваме во неколку различни променливи - секоја со свое име: imeKniga, imeAvtor, brojStranici, datumIzdavanje, итн. Но, доколку овие податоци треба да ги предадеме на друга функција, тогаш треба секоја од овие променливи да ја наведеме како аргумент на функцијата. Слично, доколку сакаме да работиме со множества (низи) од книги - тогаш треба да имаме огромен број на променливи, секоја со сопствено име и вредност.

Структури на податоци претставуваат групи од податоци дефинирани под едно име. Секој податок во структурата се нарекува нејзин член. Во програмскиот јазик C++, структури на податоци се дефинираат на следниот начин:

```
struct ime
{
   tip1 clen1;
   tip2 clen2;
   tip3 clen3;
   ....
   tipN clenN;
};
```

На пример, со следниот код може да креираме структура која содржи 4 члена (imeKniga, imeAvtor, brojStranici, datumIzdavanje) и има име Kniga:

Пример:

```
struct Kniga
{
string imeKniga;
string imeAvtor;
int brojStranici;
string datumIzdavanje;
};
```

знакот ';' мора да биде на крајот од дефиницијата на структурата.

Дефинициите на структурите најдобро е да се пишуваат надвор од функциите - иако тоа не е задолжително. Дефиницијата на податочната структура Kniga (и на која било структура која ќе се креира на начинот прикажан погоре) не зазема меморија и не креира променливи од тип Kniga. Таквата дефиниција само опишува какви податоци ќе содржат променливите од тип Kniga кои подоцна ќе ги декларираме. Всушност, со горната дефиниција креиравме податочен тип кој понатаму може да го користиме како и сите останати податочни типови (int, char, long, итн.). Во една програма, променливи од тип Kniga декларираме на истиот начин како што декларираме променливи од кој било друг тип:

```
Kniga imeNaPromenliva;
```

Со наредбата дадена погоре креирана е променлива од тип Kniga, со име imeNaPromenliva. Како и секоја друга променлива, така и овие променливи зафаќаат одреден податочен простор (меморија): по правило, овие променливи зафаќаат онолку простор колку што е потребно за да се чуваат податоците - што пак, претставува збир од меморијата потребна за чување на секој од членовите на структурата. Како и кај

сите други променливи, може да го искористиме операторот sizeof() за да одредиме колку точно бајти зафаќаат овие податоци.

Во C++, до податоците (членовите) на една структура се пристапува со помош на операторот '.'. На пример, со наредбата "imeNaPromenliva.imeKniga" се пристапува до членот imeKniga на променливата imeNaPromenliva.

```
#include <iostream>
#include <string>
using namespace std;
struct Kniga
{
string imeKniga;
string imeAvtor;
int brojStranici;
string datumIzdavanje;
};
int main()
{
Kniga prva;
prva.imeKniga = "C++ Primer Plus ";
prva.imeAvtor = "Stephen Prata";
prva.brojStranici = 1224;
prva.datumIzdavanje = "25.11.2004";
Kniga vtora;
vtora.imeKniga = "C++ how to program ";
vtora.imeAvtor = "Paul Deitel";
vtora.brojStranici = 1303;
vtora.datumIzdavanje = "20.08.2012";
cout << prva.imeKniga << " - " << prva.imeAvtor << endl;
cout << vtora.imeKniga << " - " << vtora.imeAvtor << endl;
//chlenovite na strukturata se odnesuvaat kako obichni promenlivi
//i vrz sekoja od niv mozhe da izvrshuvame najrazlichni operacii
cout << vtora.imeKniga << " ima " << (vtora.brojStranici - prva.brojStranici)<< " povekje stranici od " <<
prva.imeKniga << endl;
return 0;
  }
```

Во C++, **променливите** кои се креираат **врз база на дефиниција на одреден податочен тип се нарекуваат објекти од тој тип.** На пример, во програмата дадена погоре, **prva и vtora се објекти од типот Kniga**. Многу е важно да ги разграничиме поимите тип и објект: **тип** претставува дефиниција на податок (што е тоа Kniga и кои податоци треба да се чуваат за една книга), додека **објектите** ги содржат вистинските податоци - книга со име "C++ Primer Plus (5th Edition)".

C++ овозможува креирање на објекти (променливи) од структурата веднаш по нејзиното дефинирање. Така, помеѓу знаците '}' и ';' може да ги запишеме имињата на тие променливите од типот што е креиран. На пример, следниот код дефинира структура Kniga и креира две променливи prva и vtora:

```
struct Kniga
{
string imeKniga;
string imeAvtor;
int brojStranici;
```

```
string datumIzdavanje;
} prva, vtora;
```

Во C++ дозволено е вгнездување на една структура во друга - на пример дефинирање на структура Avtor (imeAvtor, prezimeAvtor, godinaRaganje, итн) и, потоа, дефинирање на структура Kniga која содржи член од тип Avtor. Ова овозможува креирање на најразлични хиерархии на податоци.

Како и кај сите останати променливи, по креирањето на објекти од одреден тип, потребно е, на истите, да им се додели почетна вредност. Во програмата со книгите, ова го направивме со неколку наредби - наведени веднаш по креирањето на променливите prva и vtora. Бидејќи, многу често, структурите на податоци содржат голем број на членови и потребни се повеќе наредби за нивно иницијализирање, C++ овозможува доделување на вредности уште при самото креирање на променливите (слично како кај низите од податоци). Доколку има вгнездување на структури, иницијализацијата се прави со вгнездување на самата листа од вредности:

```
#include <iostream>
#include <string>
using namespace std;
struct Avtor
{
string ime;
string prezime;
};
struct Kniga
{
string imeKniga;
Avtor avtor;
int brojStranici;
};
int main()
{
Avtor avtor = {"Stephen", "Prata"};
Kniga prva = {"C++ Primer Plus", avtor, 1224};
cout << prva.imeKniga << " - " << prva.avtor.ime <<" "<< prva.avtor.prezime<<endl;
//inicijalizacija (vgnezduvanje)
Kniga vtora = {"C++ how to program ",{" Paul ", "Deitel "}, 1303};
cout << vtora.imeKniga << " - " << vtora.avtor.ime <<" "<< vtora.avtor.prezime<<endl;
return 0;
}
```

Од примерот даден погоре може да се забележи дека, при иницијализацијата на променливите avtor, prva и vtora, вредностите мора да ги наведеме во истиот редослед како што се наведени членовите во дефиницијата на соодветната структура на податоци (кај avtor прво го наведуваме неговото име, па презиме). Бидејќи, при процесот на програмирање, често сакаме да го промениме бројот на членови кај одредена структура на податоци (со додавање или бришење на податоци), користењето на ваквите листи за иницијализација е опасно - може да доведе до грешки при извршување на програмите кои тешко се откриваат.

Пр. да се напише програма со која се составува список од n студенти. Студентите да се внесат како структура составена од име, презиме и број на бодови. Да се подреди списокот според бројот на

бодови во опаѓачки редослед.

```
#include <iostream>
#include <string>
using namespace std;
struct student
{
   char ime[50];
   char prezime[50];
   int bodovi;
};
int main()
{
   int i,j,n;
   struct student spisok[1000];
   struct student p;
   cout << "vnesi broj na studenti" << endl;
   cin>>n;
   for(i=0;i<n;i++)
   {
      cout<<"vnesi ime, prezime i bodovi"<<endl;
      cin>>spisok[i].ime>>spisok[i].prezime>>spisok[i].bodovi;
   }
   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(spisok[i].bodovi<spisok[j].bodovi)
         {
            p=spisok[i];
            spisok[i]=spisok[j];
            spisok[j]=p;
         }
      }
   }
    cout<<"podreden spisok "<<endl;
   for(i=0;i<n;i++)
   {
      cout<<spisok[i].ime<<" "<<spisok[i].prezime<<" "<<spisok[i].bodovi<<endl;
   }
   return 0;
}
```

Пр. да се напише програма со која се составува список од n ученици. Учениците да се внесат како структура составена од име, презиме, оценка и просек, а оценка е структура во која се внесуваат оценките на учениците добиени од 3 теста по програмски јазици. Да се подреди и испечати список на ученици подреден по просечната оценка во опаѓачки редослед.

```
#include <iostream>
```

```
#include <string>
using namespace std;
struct ocenka
{
   int t1;
   int t2;
   int t3;
};
struct ucenik
{
   char ime[50];
   char prezime[50];
   ocenka ocena;
   float pros;
};

int main()
{
   int i,j,n;
   struct ucenik spisok[1000];
   struct ucenik p;
   cout << "vnesi broj na ucenici" << endl;
   cin>>n;
   for(i=0;i<n;i++)
   {
      cout<<"vnesi ime, prezime i 3 ocenki"<<endl;
      cin>>spisok[i].ime>>spisok[i].prezime>>spisok[i].ocena.t1>>spisok[i].ocena.t2>>spisok[i].ocena.t3;
      spisok[i].pros=(float)(spisok[i].ocena.t1+spisok[i].ocena.t2+spisok[i].ocena.t3)/3;
   }
   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(spisok[i].pros<spisok[j].pros)
         {
            p=spisok[i];
            spisok[i]=spisok[j];
            spisok[j]=p;
         }
      }
   }
     cout<<"podreden spisok "<<endl;
   for(i=0;i<n;i++)
   {
      cout<<spisok[i].ime<<" "<<spisok[i].prezime<<" "<<spisok[i].pros<<endl;
   }
     return 0;
}
```

Пр. да се напише програма со која се составува список од n вработени. Вработените да се внесат како структура составена од име, презиме, работен стаж и плата.
Платата е структура во која се внесуваат основата и коефициентот на сложеност на работното место.
Платата се пресметува како основата помножена со коефициентот на сложеност и се додава работниот стаж помножен со 0.5.
Работниот стаж се пресметува 2020-godinanavrabotuvanje.
Да се направи мени за избор дали да се печати список на сите вработени подредени висина на плата во опаѓачки редослед или по години на стаж во растечки редослед.

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

struct plata
{
   float osnova;
   float koef;
} ;
struct vraboten
{
   char ime[50];
   char prezime[50];
   int s;
   plata prim;
   float plati;
};

int main()
{
   int i,j,n,m,g;
   struct vraboten spisok[1000];
   struct vraboten p;
   cout << "vnesi broj na vraboteni" << endl;
   cin>>n;
   for(i=0;i<n;i++)
   {
      cout<<"vnesi ime, prezime, god. na vrabotuvanje, osnova na plata slozenost na rab. mesto "<<endl;
      cin>>spisok[i].ime>>spisok[i].prezime>>g>>spisok[i].prim.osnova>>spisok[i].prim.koef;
      spisok[i].s=2020-g;
      spisok[i].plati=spisok[i].prim.osnova*spisok[i].prim.koef+spisok[i].s*0.5;
   }
   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(spisok[i].plati<spisok[j].plati)
         {
            p=spisok[i];
            spisok[i]=spisok[j];
```

```
            spisok[j]=p;
          }
       }
    }
    cout<<"vnesi 1 ako podreduvas po plata, 2 ako podreduvas po raboten staz"<<endl;
    cin>>m;
    switch(m)
    {
       case 1:
{
     cout<<"podreden spisok po plata"<<endl;

    for(i=0;i<n-1;i++)
    {
       for(j=i+1;j<n;j++)
       {
          if(spisok[i].plati<spisok[j].plati)
          {
             p=spisok[i];
             spisok[i]=spisok[j];
             spisok[j]=p;
          }
       }
    }
    for(i=0;i<n;i++)
    {
       cout<<spisok[i].ime<<" "<<spisok[i].prezime<<" "<<spisok[i].plati<<endl;
    }
}
break;
       case 2:
          {
     cout<<"podreden spisok po godini na raboten staz"<<endl;
     for(i=0;i<n-1;i++)
    {
       for(j=i+1;j<n;j++)
       {
          if(spisok[i].s>spisok[j].s)
          {
             p=spisok[i];
             spisok[i]=spisok[j];
             spisok[j]=p;
          }
       }
    }
    for(i=0;i<n;i++)
    {
       cout<<spisok[i].ime<<" "<<spisok[i].prezime<<" "<<spisok[i].s<<endl;
    }
```

```
        }
    }
      return 0;
}
```

Пр. Да се напише програма која од стандарден влез ќе чита податоци за n држави и на екран ќе го отпечати името и презимето на претседателот на државата чиј што главен град има најмногу жители. Податоци за држава:
• име
• претседател
• главен град
• број на жители.

Податоци за град:
• име
• број на жители.

Податоци за претседател:
• име
• презиме
• политичка партија.

```
#include <iostream>
#include <string>
using namespace std;
struct grad
{
   string ime;
   long int broj;
};
struct pretsedatel
{
   string ime;
   string prezime;
   string polpart;
};
struct drzava
{
   string ime;
   pretsedatel pret;
   grad glaven;
   long int broj;
};
int main()
{
   int i,s,m,n;
   struct drzava l[1000];
   struct drzava p;
   cout << "vnesi broj na drzavi" << endl;
```

```
   cin>>n;
   for(i=0;i<n;i++)
   {
      cout<<"vnesi ime na drzava, ime na pretsedatel prezime, pol.partija, glaven grad i broj na ziteli na gl.
grad"<<endl;
      cin>>l[i].ime>>l[i].pret.ime>>l[i].pret.prezime>>l[i].pret.polpart>>l[i].glaven.ime>>l[i].glaven.broj;
   }
   m=0; s=0;
   for(i=0;i<n;i++)
   {
     if(l[i].glaven.broj>m)
         {
            m=l[i].glaven.broj;
            s=i;
         }
      }
    cout<<"pretsedatel e  "<<l[s].pret.ime<<"  "<<l[s].pret.prezime<<endl;
   return 0;
}
```

ВЕЖБИ:

Претходните задачи да се решат со помош на структури и датотеки!.