УПОТРЕБА НА ПОКАЖУВАЧИ КАЈ СТРУКТУРИ

Покажувачи и структури се јавуваат заедно во следниве случаи: покажувачот е елемент на структурата, покажувачи на структури и покажувач на структурата е елемент на структурата

*ime_na_strukturata.ime_na_pokazuvac.

Слично како кај покажувачите на основните типови податоци, може да се декларираат покажувачи на структури. При иницијализацијата на покажувачот во овој случај му се доделува адреса на структурата. Пристап до елементите на структурата се врши со операторот "->" бидејќи про користање на операторот точка ". " иако е точно, се користи повеќе пати операторот "*" и загради со што програмата е понепрегледна.

Синтаксата е:

ime_na_pokazuvac_na_struktura->ime_na_element_na_struktura

или:

(*ime_na_pokazuvac_na_struktura). ime_na_element_na_struktura

Ако во структурата во која се пристапува преку покажувач се наоѓа покажувач до тој елемент се пристапува на следниов начин:

* ime_na_pokazuvac_na_struktura-> ime_na_pokazuvac_vo_struktura

или:

*(*ime_na_pokazuvac_na_struktura). ime_na_pokazuvac_vo_struktura

Пр.

```
struct tocka
{
int x;
int y;
} p1, *pp1;
```

p1 е променлива од тип struct tocka, а pp1 е покажувач на структура од тип tocka. Покажувач може да се иницијализира со адресата на променливата:

pp1= &p1;

До член на структурата може да се пристапи со (*pp1).x и (*pp1).y. Заградите се неопходни бидејќи операторот „.“ има повисок приоритет од операторот на дереференцирање.
Изразот *pp1.x е еквивалентен на *(pp1.x), што не е точно формиран израз. Поедноставно е да се користи операторот „->“ . Така:

p1.x е исто со pp1->x.

Ако членот на структурата и самиот е структура, може да се комбинираат операторите -> и . за да се дојде до елемент од подструктурата:

ptvar->clen.podclen

Ако некој член на структурата е поле, тогаш се пристапува со:

ptvar->clen[izraz]

Структурата може да е составена од покажувачи. Бидејќи операторите „ . “и „ ->“ имаат повисок приоритет од операторот *, пристапот е:

*var.clen
ili
*ptvar->clen

Слично, бидејќи операторите „ . “и „ ->“ имаат највисок приоритет, изразите како:

++ptvar->clen и ++ptvar->clen.podclen
Се еквивалентни на:

++(ptvar->clen) и ++(ptvar->clen.podclen).

Изразот: (++ptvar)->clen ја зголемува вредноста на покажувачот на структурата пред да пристапи до член на структурата. Адресата која ја содржи ptvar се зголемува за онолку колку е големината на структурата.

Пр.
struct pravoagolnik r, *pr=&r;
Тогаш следниве изрази се еквивалентни:
r.pt1.x
pr->pt1.x
(r.pt1).x
(pr->pt1).x

Кога имамае покажувач на структура со член покажувач, се јавуваат посложени изрази.

Пр.
Struct k
{
int n;
char *ch;
} *pt;

Тогаш мозе да имаме изрази:
(++pt)->ch
(pt++)->ch

При што (++pt)->ch го зголемува покажувачот pt а потоа го враќа членот на структурата, а (pt++)->ch прво го враќа членот на структурата а потоа го зголемува покажувачот pt. Ова е исто со
pt++->ch

Изразот:
*pt->ch++
Го зголемува членот ch откако ќе ја земи вредноста на која покажува pt->ch.
Операторот -> има највисок приоритет и прв се применува. Унарните оператори имаат ист приоритет и асоцијативност. Значи прво се применува операторот за зголемување на покажувачот pt->ch, а потоа операторот за дереференцирање. Бидејќи операторот за зголемување е во постфикс нотација, зголемување ќе се изврши откога ќе се земи вредноста на која pt->ch покажува.
За да се зголеми објектот на кој pt->ch покажува, треба (*pt->ch)++ или *pt++->ch.
Ќе го зголеми pt откако ќе пристапи до објектот на кој ch покажува.

Пр. struct student
```
    {
        int index;
        char imeprezime[30];
        int godstudii;
    }
student *a1, *a2;
```

со:
a1=&ana;
a2=&mile;

на покажувачот a1 му се доделува адресата на структурата ana, т.е. се поставува да покажува на структурата ana. Покажувачот a2 покажува на структурата mile.
Пристап до членовите на структурите ana и mile може да се добие:

a2-> imeprezime
е исто со:
mile. imeprezime
или со:
(*a2). imeprezime

a1-> godstudii;
е исто со:
ana.godstudii;
или со:
(*a1).godstudii;

Пр. програма со која се внесува и печати број на индекс, име, презиме и година на студии на студент
```
#include <iostream>
using namespace std;
```

```
struct student
{
   int index;
   string ime;
   string prezime;
   int godstudii;
};

int main()
{
 student s1;
 student *sp=&s1;
   cout << "vnesi broj na indeks, ime, prezime i god na studii na student" << endl;
   cin>>sp->index>>sp->ime>>sp->prezime>>sp->godstudii;
   cout<<"vneseno e "<<endl;
   cout<< sp->index<<" "<<sp->ime<<" "<<sp->prezime<<" "<<sp->godstudii;
   return 0;
}
```

Пр. програма со која се внесуваат податоци за n студенти. Студентот е структура составен од име, презиме , број на индекс и просек. Да се подреди списокот на студенти по просек во опаѓачки редослед. Да се користат покажувачи кон членовите на структурата.

```
#include <iostream>
using namespace std;
typedef struct student
{
   int index;
   string ime;
   string prezime;
   float prosek;
};

int main()
{
  student s1[1000];
 student *sp=s1;
 student p1;
 student *p=s1;
 int n,j,i;
 cout<<"vnesi broj na studenti"<<endl;
 cin>>n;
 for (i=0;i<n;i++)
 {
   cout << "vnesi broj na indeks, ime, prezime i prosek na student" << endl;
   cin>>(sp+i)->index>>(sp+i)->ime>>(sp+i)->prezime>>(sp+i)->prosek;
 }
 for (i=0;i<n-1;i++)
```

```
  {
     for(j=i+1;j<n;j++)
     {
        if(((sp+i)->prosek)<((sp+j)->prosek))
        {
            p1=*(sp+i);
           *(sp+i)=*(sp+j);
           *(sp+j)=p1;
        }
     }
  }
  cout<<"spisokot e "<<endl;
    for (i=0;i<n;i++)
  {
     cout<< (sp+i)->index<<" "<<(sp+i)->ime<<" "<<(sp+i)->prezime<<" "<<(sp+i)->prosek<<endl;
  }
    return 0;
}
```

Забелешка: со користење на typedef при дефинирање на структурате се овозможува податок од тип на структура да се користи било каде во програмата.

Прашања поврзани со наставните единици може да се испраќаат на email: anetastojceska@gmail.com