Покажувачи

Според операциите што можат да се извршат над одредено множество податоци, разликуваме повеќе типови на податоци. Под тип на податок се подразбира множество на податоци T={t1, t2,….,tn} врз кои можат да се применат кои било операции од множеството O={o1,o2,…om}.За секој податок што се користи во програмата мора да се дефинира неговиот тип.
Типовите податоци можат да бидат:

- статички
- динамички

Статички типови податоци се оние чија големина е однапред дефинирана (најчесто на почетокот на програмата). Тие се сместени на фиксни локации во меморијата.

Динамичките типови на податоци се оние чија големина и/или структура се менува во текот на извршувањето на програмата. Тие не се сместуваат на фиксна локација во меморијата, туку на локација која во моментот на нивното креирање е слободна.Динамичките типови на податоци можат да бидат:

- со променлива големина (низа битови, низа знаци, општа низа, множество, куп, ред и датотека)
- со променлива структура (листа, дрво, покажувач и објект).

При преведувањето на програмата потребно е преведувачот да ги знае сите променливи и нивниот тип. Променливите кои се создаваат за време на извршување на програмата, а потоа се бришат се нарекуваат **динамички променливи.**

Динамичките променливи кои добиваат податоци од тип покажувач се наречени **променливи од тип покажувач или само покажувачи**. При креирањето, динамичките променливи се сместуваат во слободната внатрешна (динамичка) меморија, наречена heap-меморија.

Користењето на променливи од тип покажувач нуди две битни погодности во однос на меморијата:

1. Се проширува меморискиот простор што може да се користи за податоци во една програма
2. Со користење на покажувачките променливи во динамичката меморија, програмата може да се извршува со помала количина неопходна меморија.

На пример, програмата може да поседува две многу сложени структури на податоци што не се користат истовремено. Ако овие две структури се декларираат како глобални променливи, тогаш тие се наоѓаат во сегментот за податоци и завземаат меморија за цело време на извршувањето на програмата, без оглед дали се користат во моментот или не. Но, ако се дефинирани преку покажувачи (динамички), тие се наоѓаат во динамичката меморија и ќе бидат избришани од неа по престанокот на нивното користење

Декларација на покажувачи

Покажувачките променливи не содржат податоци на ист начин како и другите променливи.

Овој тип променливи наместо податоци содржи локација т.е. адреса на податокот што се наоѓа во динамичката меморија.

Покажувачите се декларираат на следниот начин:

*тип *име_на_покажувач;*

каде што:

тип- е типот на променлива на кои ќе покажува покажувачот

*- знак за декларација на покажувач т.е покажувачка променлива

Пример: int *pok;

се декларира покажувачот (покажувачката променлива) pok, кој може да се користи само за да покажува на променливи од тип int.

Адресен оператор &

Операторот & се нарекува адресен оператор. Тој е унарен оператор, кој ја враќа адресата на операндот по него.

Пример:

int y=5;

int *yPok;

со наредбата

yPok=&y;

на покажувачот (покажувачката променлива) yPok му се доделува адресата на променливата y. Затоа за покажувачот yPok се вели дека покажува на y.

Оператор за дереференцирање *

Променливата yPok може да се означи и преку покажувачот yPok со *yPok.

Операторот * се нарекува **индиректен оператор** или **оператор за дереференцирање.** Тој ја враќа вредноста на променливата на која покажува неговиот операнд. Односно *yPok ја враќа вредноста на променливата на која покажува покажувачот yPok.

Пример:

cout<< *yPok;

ја печати вредноста на променливата y, т.е 5.

Пример:

int ip;*

int a=47;

*int *ipa=&a;*

На овој начин a и ipa се иницијализирани и покажувачот кон цел број ipa ја содржи адресата на a, а за да се пристапи кон променливата a преку покажувачот, покажувачот се дереференцира на следниот начин:

**ipa=110;*

сега a содржи вредност 110 наместо 47.

Иницијализација на покажувачи

Покажувачите треба да се иницијализираат или при декларација или во некоја наредба за доделување. Покажувачот исто така може да биде иницијализиран на 0, NULL. Покажувачот со вреднот NULL не покажува на ниту една променлива. NULL покажувач не е исто што и неиницијализиран покажувач!

*Пример за користење ба операторите & и **

```
#include
using namespace std;
void main()
{
int a;
int *aPok;
a=7;
aPok=&a;
cout<<" Adresata na a e" << &a << "\n Vrednosta na aPok e " << aPok;
cout<<"Vrednosta na a e "<< a << "\n Vrednosta na *aPok e "<< *aPok;
cout<<"Pokazuvame deka * i & se komplementni &*aPok="<< &*aPok << " *&aPok=" << *&aPok << endl;
}
```

Покажувачи и низи

Постои тесна врска меѓу низите и покажувачите.Доколку е декларирана и иницијализирана низата:

int a[]={1,2,4,8,16};

Нејзиниот идентификатор a има вредност на адресата на првиот елемент од низата и е од *тип покажувач на променливи од типот на елементите на низата.* Во овој случај a е покажувач на целобројни променливи затоа:

a;

&a[0];

ја соджат адресата на првиот елемент од низата.Исто така покажувачите често се користат за пристап до елементите на низа.Тоа е можно бидеќи елементите на низите во меморијата се сместуваат во последователни адреси.

Пример:

int a[10], *p;

со наредбата:

p=a;

на покажувачот p му се доделува вредноста на адресата на првиот елемент на низата a[]. А додека вредноста на првиот елемт од низата може да се добие со *a или a[0].

Наредбите new и delete

Наредбата new има форма

покажувач= new тип;

Со неа во меморијата се креира променлива од соодветен тип и на покажувач му се доделува адресата на креираната променлива. Во спротивно покажувачот добива вредност NULL.

Пример, ако е деклариран покажувач

```
int *ip;
```

тогаш со наредбата

```
ip=new int;
```

во меморијата се креира неименувана променлива и нејзината адреса му се доделува на покажувачот ip.

Наредбата delete се користи за бришење (ослободување) на мемориската локација.

Пример, со:

```
delete ip;
```

се ослободува меморијата на променливата на која покажувал покажувачот ip, при што тој останува недефиниран.

Внимателно со покажувачите!

„Играњето“ со меморијата може да биде ризично и да причинува некои програмски грешки, меѓутоа истовремено и некои програмски задачи не можат (или е потешко возможно) да се реализираат без употреба на покажувачите!