Лабораторно упражнение N 9
Указатели. Адресна аритметика

I. Теоретична обосновка

Всеки елемент от програма (константа, променлива) се съхранява в определени клетки от паметта или, както е прието да се казва: на определен адрес от паметта.

Променлива, чиято стойност е адрес от паметта се нарича ***указател***. Стойността на указателя посочва (указва) местоположението на дадена променлива и косвено служи за достъп до нея, за разлика от познатия, директен достъп до променливата, чрез името й.

Като обобщение на казаното, ***указателят е променлива, чиято стойност е адрес на променлива.***

Използването на указатели е начин за писане на ефективни и компактни програми. От друга страна, ако не бъдат правилно използвани, указателите могат да доведат до разрушаване на данните.

Както всяка променлива и указателят трябва да се дефинира. Общият вид на дефиницията на променлива-указател е:

тип *име_указател [=стойност];

Тип определя типа на съдържанието на указателя т.е. типа на променливата, чийто адрес ще сочи указателя.

Име_указател е идентификатора на променливата указател. Символ * пред името на променливата определя, че променливата е указател. Обикновено, имената на променливите указатели започват с **p**, от английският термин **pointer** (указател).

Както всяка друга променлива, така и указателят може да бъде инициализиран с дадена начална стойност. Тази стойност трябва да бъде адрес на променлива.

Примери:

```
int *px;          // px е указател към променлива от тип int
float *p1;        // p1 е указател към променлива от тип float
double *p2;       // p2 е указател към променлива от тип double

int x;
int *px=&x;       // указателят px сочи адреса на променливата x
```

Ако указателят не е инициализиран, неговата стойност е NULL. Тази стойност може да бъде присвоена на указател от всеки тип.

2. Операции с указатели. Адресна аритметика

Съществуват две специални операции с указатели, които позволяват тяхното ефективно използване:

операция ***&*** определя адреса на операнда

операция ***** определя стойността от посочения адрес.

Форматът на операциите & и * е следния:

&променлива

Освен променлива, операндът може да е и елемент на масив. Като резултат операцията връща адреса на променливата.

****указател***

Резултат от операцията е стойността на променливата, чийто адрес съдържа указателя.

Върху указателите могат да се прилагат по-малко на брой аритметични операции, отколкото върху обикновените променливи. Освен това, изпълнението на аритметичните операции върху указатели е свързано с някои особености, поради което тези операции са известни още като ***адресна аритметика***.

Особеност на адресната аритметика е, че при изпълнението на операциите *събиране, изваждане, инкрементиране и декрементиране* автоматично се извършва ***мащабиране***, което отчита типа на обектите, към които сочат указателите.

4. Указатели и масиви

В C/C++ съществува пряка връзка между указатели и масиви - ***имената на масивите се явяват указатели***, като съдържат адреса на първия елемент от масива т.е. елемент с индекс 0.

По този начин, достъпът до елемент на масив може да се осъществи и чрез указател.

Пример:

```
int array[5];
int *p;
int x,y,i;

x=array[0];      // променливите x, y осъществяват достъп до един и същ елемент
y=*array;

x=array[i];      // променливите x, y осъществяват достъп до елемент с индекс i
y=*(array+i);

p=array+i;       // също, достъп до елемент с индекс i
y=*p
```

Основната разлика между името на масива и променливата-указател е, че името на масива се явява константа и стойността му не може да бъде променена. Например:

```
p=array;
p++;             // изразите са допустими

array=p;
array++;
array=&x;        // изразите са недопустими
```

Достъпът до елементите на масив може да се осъществи както чрез индекс, така и чрез указател. Достъпът чрез указател е много по-бърз, отколкото този чрез индекс и поради тази причина често е предпочитан в програмите на C/C++.

Пример за достъп до елементите на масив чрез указател е даден с програмната реализация на задачата за сумиране на елементите на едномерен масив mas от 10 целочислени елемента. Указателят p сочи текущият елемент от масива.

```
// намиране ма сумата от елементите на едномерен масив
#include <stdio.h>
main()
{ int mas[10],*p;
  int i,sum;
   p=mas;
  for(i=0;i<10;i++)
   { printf ("insert the elements of the masive=");
```

```
      scanf ("%d",p);            // достъпът де текущия елемент е чрез указател
      p++;
   }
  p=mas;
  sum=0;
    for (i=0;i<10;i++)
      { sum+=*p; // достъпът де текущия елемент е чрез указател
        p++;
       }
  printf ("sum=%d",sum);
  return 0;
}
```

5. Указатели към указатели

Тъй като указателят също е променлива, която се разполага на определен адрес, възможно е, друг указател да сочи неговият адрес. такъв тип указател се нарича ***двоен указател***.

Дефинирането на двоен указател се извършва по следният начин:

тип **име[=стойност];

Например, нека x е променлива; px - указател към променливата и нека ppx да е указател, съдържащ адреса на px. Дефинирането на тези променливи е по следният начин:

```
int x;
int *px=&x;
int **ppx=&px;
```

Приложението на двойните указатели е при работата със различни структури от данни като дървета, списъци, графи, както и при работа с многомерни масиви.

II. Контролни въпроси

1. Какво се дефинира със следния програмен ред?

```
int* px, py, pz;
```

2. Има ли разлика ако дефиницията е записана `int *px, py, pz;` ?

3. Напишете програмен ред, с който дефинирате 3 указателя към тип int.

4. Какво ще се изведе на екрана след изпълнението на програмен код:

```
int* px, py;
py=10;
px=&py;
++*px;
printf("%d\n", py);
```

5. Какво ще се изведе на екрана, ако програмен ред 4 се замени с програмен ред: `*++px;`

III. Задачи за изпълнение

1. Да се за размяна на две числа, като достъпът се осъществява чрез указатели.

2. Да се създаде конзолно приложение, с което се дефинира едномерен масив от 10 целочислени елемента. Да се въведат стойностите на елементите и да се изведат като се използват три начина: достъп чрез индекс посредством оператор [], достъп чрез индекс посредством адресен оператор *, достъп чрез указател

3. Да се състави конзолно приложение за търсене на най-голям и най-малък елемент в едномерен масив от 10 реални елемента. Достъпът до елементите на масива да се осъществи чрез указатели.