УПРАЖНЕНИЕ №7

УКАЗАТЕЛИ. СЪЩНОСТ И ДЕКЛАРИРАНЕ НА УКАЗАТЕЛ. ВРЪЗКА МЕЖДУ УКАЗАТЕЛИ И МАСИВИ. ОПЕРАЦИИ С УКАЗАТЕЛИ.

Променлива, чиято стойност е адрес на клетка от паметта се нарича указател. Тя указва мястото, където е съхранена стойността на променливата.

Синтаксисът на дефиниране на указател към променлива от даден тип е следния:

type_var * pname;

Където ***type_var*** е стандартен или дефиниран от програмиста тип данни, ***pname*** е името на указателя, а ****pname*** е стойността на адрес ***pname***.

Примери:

```
int *pi;
float *pfl;
double *pdbl;
```

Операции.

Известни са две операции за работа с указатели:

& Присвояване на адрес на променлива от тип указател. На указателя се задава стойност, която посочва място от паметта, където се съхранява променлива от типа на указателя. Присвояване на адрес на променлива от друг тип е грешка.

*** Извличане на стойност на променливата от паметта, към която сочи указателят.**

Синтаксисът на използуване на операцията от първия вид е:

type_var *pname, var_name;
pname = & var_name;

Където ***pname*** е име на указателя, а ***var_name*** е име на променлива от тип, съвпадащ с този на указателя. При дефинирането на указателите трябва да се спазва условието – дефинирания тип на указателя трябва да съвпада по тип с обекта, към който му се присвоява адрес. Присвояване на адрес на променлива от друг тип е грешка.

Операцията * е точно противоположна по смисъл. Нейният синтаксис е:

type_var * pname, var_name, n_var;
pname = & var_name;
nvar = *pname;

Зад1. Деклариране и инициализиране на указатели.

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

int main ()
{
  int first, second=100;
  int * p;
  p = &first;
  *p = 10;
  p = &second;
  *p = 20;
  cout << "first is " << first << endl;
  cout << "second is " << second << endl;
  return 0;
}

Резултат:
first is 10
second is 20
```

Зад2. Да се декларират два указателя от тип int, да се инициализират и да се извършат действия за присвояване върху тях.

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

int main ()
{
  int first = 5, second = 15;
  int * p1, * p2;

  p1 = &first;       // p1 = address of first
  p2 = &second;      // p2 = address of second
  *p1 = 10;          // value pointed by p1 = 10
  *p2 = *p1;         // value pointed by p2=value pointed by p1
  p1 = p2;           // p1 = p2 (value of pointer is copied)
  *p1 = 20;          // value pointed by p1 = 20

  cout << "first is " << first << endl;
  cout << "second is " << second << endl;
  return 0;
}

Резултат:
first is 10
second is 20
```

Зад3. Операции с указатели.

```
#include<iostream.h>
void main()
{
      int x=20;
```

```
      int *px;
      px = &x;
      cout<<"*px="<<*px<<endl;
      cout<<"&x="<<&x<<endl;
      *px=*px+2;
      cout<<"x="<<x<<endl;
     cout<<"px="<<px<<endl;
      cout<<"*px="<<*px<<endl;
      *px=*px-10;
      cout<<"*px="<<*px<<endl;
      cout<<"x="<<x<<endl;
}
```

Предаване на параметри на функции чрез указатели.

Когато е необходимо функциите да работят с оригиналните параметри, а не с техни копия, параметрите трябва да бъдат предавани или чрез указатели или чрез псевдоними.

Зад4. Да се напише програма, която въвежда стойности на реалните променливи a и b, след което разменя стойностите им.

1начин: Чрез параметри - указатели

```
#include <iostream.h>
#include <iomanip.h>

void swap(double *x, double *y)
{ double work = *x;
      *x = *y;
      *y = work;
      }
int main()
{
double a, b;
cout << "a, b= ";
cin >> a >> b ;
cout << setw(10) << a << setw(10) << b << endl;
swap(&a, &b);
cout << setw(10) << a << setw(10) << b << endl;
return 0;
}
```

В езика C++ има още един начин за предаване **на параметри на функции - по псевдоним (reference).**

2начин: Чрез параметри - псевдоними

```
#include <iostream.h>
#include <iomanip.h>
void swapi(double &x, double &y)
{
      double work = x;
      x = y;
y = work;
}
```

```
int main()
{
      cout << "a, b= ";
      double a, b;
      cin >> a >> b ;
      cout << setw(10) << a << setw(10) << b << endl;
      swapi(a, b);
cout << setw(10) << a << setw(10) << b << endl;
return 0;
}
```

Указатели и масиви.

```
int a[10], x;
int *pa;

pa = &a[0];        // pa ukazatel kam address of a[0]
x = *pa;           // x= стойността на pa или x=a[0]

pa[i]  ≡  *(pa + i);
(pa + i)  ≡ a[i];
```

Зад5. Да се напише програма, в която чрез отделни функции:

- се въвеждат елементи на едномерен целочислен масив;
- сортира се масива във възходящ ред;
- извеждат се сортираните елементи.

//Функция за въвеждане елементи на едномерен целочислен масив

```
void input(int *a, int n)
      {for (int i=0; i<n; i++)
            { cout<<"a ["<<i<<"] =";
              cin>>*(a+i);
        }    }
```

//Функция за извеждане елементи на едномерен целочислен масив

```
void output(int *a, int n)
      {for (int i=0; i<n; i++)
            cout<<"a ["<<i<<"] ="<<*(a+i)<<endl;
       }
```

//Функция за сортиране по метод на мехурчето чрез използване на указатели.

```
void sort (int *array, int num)
 {
      int i, j;
      int temp;  /* Used in swopping array values */
      for (i = 0; i < num; i++)
      for (j = 0; j < num-1; j++)
      if (*(array+j) > *(array+j+1))
       {
            temp = *(array+j);
           *(array+j) = *(array+j+1);
```

```
            *(array+j+1) = temp;
        }
 }

int main()
{     const N=50;  int n;
      int a[N];
  do
      { cout<<"n=";  cin>>n;
      } while (n<1||n>50);

      input(a,n);                    // ПП за въвеждане
      sort(a, n);
      cout<<"Sortiraniat masiv e:"<<endl;
      output (a,n);
}
```

// **Функция за сортиране по метода на пряка селекция.**

```
void izborsort(int a[],int n)
{
    int i,j,ind;
    int xind;
    for(i=0;i<n-1;i++)
   {
            ind=i; xind=a[i];
            for (j=i+1; j<n; j++)
              if (a[j]<xind)
              {
                ind=j ;
                xind=a[j];
               }
        a[ind]=a[i];
        a[i]=xind;
     }
}
```

Динамична памет.

Обекти, които се създават и унищожават по време на изпълнение на програмата се наричат **динамични.**

<Pointer to Type> = new <Type> [<инициализатор>]

```
new[]
int * arr;
arr = new int [5];
delete pointer;
delete [] pointer;
```

Зад6. Пример за динамичен масив.

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

```
int main()
{
int size,i;
cout << "size= ";
cin >>size;
int *p = new int[size];
if (p == 0)
cout << "Error: memory could not be allocated";
else
{
for (i=0; i<size; i++)
{
cout << "Enter number: ";
// cin >> p[i];
cin>>*(p+i);
}
cout << "You have entered: ";

for (i=0; i<size; i++)
cout << *(p+i) << ", ";
delete[] p;
}
return 0;
}
```

Зад7. Да се напише програма за прехвърляне на положителните елементи на даден едномерен масив в друг, като се използва функция с параметри указатели.