Тема 7. ДИНАМИЧЕСКИЕ МАССИВЫ. УКАЗАТЕЛИ

Цель занятия: овладеть навыками алгоритмизации и программирования структур с использованием указателей; получить практические навыки в составлении программ, в которых используются динамические массивы.

Теоретические сведения

Указатели.

Указатель – переменная, которая хранит адрес какого-то объекта.

Например:

```
char c = 'a';
char* p = &c;      // p содержит адрес c.    & - означает получение адреса операнда
char c2 = *p;      // c2 =  'a'
```

Перед присваиванием указателю адрес переменной, элемента массива, функции или другого объекта может быть получен с помощью оператора взятия адреса &.

Например:

```
char* p = &v[3];      //указатель содержит адрес 4-го элемента массива v
int* q = &inch;       //указатель  содержит адрес переменной inch
double (*fp)(double);
fp = &sqrt;           //указатель содержит адрес функции  sqrt()
```

Над указателями могут выполняться арифметические действия, имеющие смысл для указателей, например:

```
p = p + 2;            //указатель содержит адрес 6-го элемента массива v
```

Для доступа к значению объекта, адресуемого указателем, используется оператор разыменования указателя *. Эта операция также называется косвенным обращением. Оператором разыменования является *.

Например:

```
int i = *q;           // переменная i содержит значение, которое
                      // содержалось в переменной inch
*p = 'c';             // 6-му элементу массива v присвоен код символа 'c'
double sq = (*fp)(4.0); // переменная sq содержит 2.0 - результат
                      // выполнения функции sqrt(4.0)
```

Указатели дают возможность косвенно адресовать программные объекты, благодаря чему появляется возможность динамического создания, модификации и удаления переменных, таблиц функций и других программных структур и объектов.

Указатели на массивы

Имя массива можно использовать в качестве указателя на его первый элемент. Например:

```
int v[] = {1,2,3,4};
int* p1 = v;          //указатель на 1-й элемент
int* p2 = &v[0];      //указатель на 1-й элемент
int* p3 = &v[4];      //указатель на элемент, следующий за последним.
```

Доступ к элементам массива может осуществляться при помощи указателя на массив и индекса либо при помощи указателя на элемент массива.

Пример с использованием индекса:

```
for (int i = 0; i < N; i++)
    cout << arr [i] << " ";
```

Это эквивалентно использованию указателя:

```
int* p = &arr[0];                //адрес первого элемента массива
// вывод значений элементов массива
for (int i=0; i < N; i++, p++)
   cout<< *p<< " ";
```

Префиксный оператор * означает разыменование, поэтому *p есть символ, на который указывает p. Оператор ++ увеличивает значение указателя на размер элемента массива.

Результат применения операторов +, -, ++ или – к указателю увеличивает или уменьшает значение указателя на количество элементов, а не на число байтов.

Доступ к элементам возможен с помощью индексированных переменных ***arr [i]*** (для одномерных массивов), ***matr[i][j]*** (для многомерных массивов) и с помощью указателей ****(arr +i) и *(*(matr+i)+j))*** соответственно для одномерных и двумерных массивов**.**

Для создания динамических переменных используют операцию ***new***, определенную в C++:

указатель = new имя_типа[(*инициализатор)*];

где ***инициализатор*** – выражение в круглых скобках.

Операция **new** позволяет выделить и сделать доступным участок динамической памяти, который соответствует заданному типу данных.

Если задан инициализатор, то в этот участок будет занесено значение, указанное в инициализаторе:

int *x=new int(5);

Для удаления динамических переменных используется операция ***delete***, определенная в C++:

delete указатель;

где **указатель** содержит адрес участка памяти, ранее выделенный с помощью операции ***new***:

delete x;

Динамические массивы

Операция ***new*** при использовании с массивами имеет следующий формат:

new тип_массива

Такая операция выделяет для размещения массива участок динамической памяти соответствующего размера, но не позволяет инициализировать элементы массива. Операция *new* возвращает указатель, значением которого служит адрес первого элемента массива. При выделении динамической памяти размеры массива должны быть полностью определены.

Например: выделение динамической памяти:

1. ***int *a=new int[100];*** /*выделение динамической памяти размером ***100*sizeof(int)*** байтов*/

double *b=new double[12]; /* выделение динамической памяти размером 12*sizeof(double) байтов */

2. ***long(*la)[4];*** /*указатель на массив из 4 элементов типа long*/

la=new[2][4]; /*выделение динамической памяти размером 2*4*sizeof(long) байтов*/

3. ***int**matr=(int**)new int[5][12];*** /*еще один способ выделения памяти под двумерный массив*/

4. ***int **matr;***

matr=new int*[4]; /*выделяем память под массив указателей ***int**** их ***n*** элементов*/

for(int I=0;I<4;I++) matr[I]=new int[6];/*выделяем память под строки массива*/

Указатель на динамический массив затем используется при освобождении памяти с помощью операции ***delete***.

Например: освобождение динамической памяти.

delete[] a //освобождает память, выделенную под массив, если a адресует его начало

delete[]b;

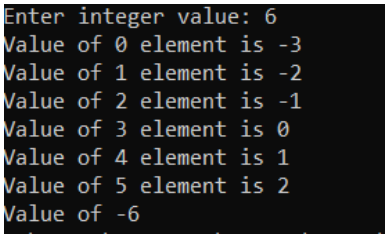
delete[] la;

for(I=0;I<4;I++) delete matr[I]; //удаляем строки

delete [] matr;//удаляем массив указателей.

Пример 7.1. Вычислить сумму отрицательных элементов массива

```
#include <iostream>
using namespace std;
int main()
{
   int num;                // размер массива
   cout << "Enter integer value: ";
   cin >> num;                    // получение от пользователя размера массива
   int* p_darr = new int[num]; // Выделение памяти для массива
   for (int i = 0; i < num; i++)
   {
      // Заполнение массива и вывод значений его элементов
      *(p_darr + i) = i - 3;// p_darr[i] = i - 3;

      cout << "Value of " << i << " element is " << p_darr[i] << endl;
   }
   int* p = p_darr;       //адрес первого элемента массива
   int  s = 0;
   for (int i = 0; i < num; i++, p++)
      if (*p < 0)
         s += *p;
   cout << "Value of " << s;
   delete[] p_darr; // очистка памяти
   return 0;
}
```



Задача 7.1. Вычислить сумму четных элементов массива

Пример 7.2. Проверить, упорядочен ли массив по возрастанию.

```
#include <windows.h>
#include <iostream>
using namespace std;
int main()
{
  SetConsoleOutputCP(1251);
    int N;                              // размер массива
    cout << "Введите размерность массива: ";
    cin >> N;                           // получение от пользователя размера массива
    int *arr = new int[N];              // выделение памяти для массива
     cout << "Введите элементы массива: ";       // заполнение массива
    for (int i = 0; i < N; i++)
    cin >>arr[i];                        // cin >>*( arr +i);

    int* p = arr;               //адрес первого элемента массива
    cout<< "Массив:\n";         // вывод значений элементов массива
    for (int i=0; i < N; i++, p++)
       cout<< *p<< " ";              //  cout<< arr[i];
    cout<<"\n";
  //после предыдущего цикла p указывает на следующий за последним
  //элемент массива, надо снова присвоить p адрес начала массива
  p = arr;
  bool b = true;
  for (int i=0; i< N-1; i++, p++)
            if(*p > *(p+1))
            {
                    b = false;
                    break;
            }
  if(b)
            cout<<"Массив отсортирован по возрастанию\n";
  else
            cout<<"Массив не упорядочен по возрастанию\n";
  delete [] arr; // очистка памяти
  return 0;
}
```

Задача 7.2. Проверить, упорядочен ли массив по убыванию.
Пример 7.3. Найти максимальный элемент среди четных элементов массива.

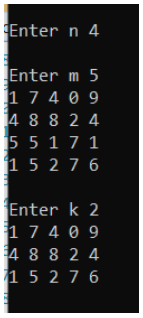
```
#include <windows.h>
#include <iostream>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int N;            // размер массива
    cout << "Введите размерность массива: ";
    cin >> N;       // получение от пользователя размера массива

    int *arr = new int[N];          // Выделение памяти для массива
    cout << "Введите элементы массива: ";   // Заполнение массива
    for (int i = 0; i < N; i++)
        cin >>arr[i];
    int* p = arr;             //адрес первого элемента массива
    cout<< "Массив:\n";          // вывод значений его элементов
    for (int i=0; i < N; i++, p++)
       cout<< *p<< " ";
    cout<<"\n";
    int max,i=0;
    p = arr;                          //адрес первого элемента массива
    while (*p % 2!=0 && i < N) {p++; i++;}//поиск первого четного элемента массива
    if (i< N-1)                       //если четный элемент найден,
//то ищем максимальный среди оставшихся четных элементов массива
  {   max=*p;
      p++;
     for (int j=i+1; j < N; j++, p++)
           if(*p % 2==0 && *p >max)  max=*p;
  }
   cout<<"Максимальный среди четных элементов массива = "<<max<< "\n";
   delete [] arr;                     // очистка памяти
   return 0;
}
```

Задача 7.3. Найти минимальный элемент среди нечетных элементов массива.

Пример 7.4. Удалить из матрицы строку с номером K.

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;
void main()
{
  int n, m;                 //размерность матрицы
  int i, j;
```

```
cout << "\nEnter n";
cin >> n;                    //ввод количества строк
cout << "\nEnter m";
cin >> m;                    // ввод количества столбцов
//выделение памяти
int** matr = new int* [n];      /* массив указателей на строки*/
for (i = 0; i < n; i++)
        matr[i] = new int[m];           /*память  под  элементы матрицы*/
        //заполнение матрицы
for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
                *(*(matr + i) + j) = rand() % 10;       //заполнение матрицы
                //печать сформированной матрицы
for (i = 0; i < n; i++)
{
        for (j = 0; j < m; j++)
                cout << *(*(matr + i) + j) << " ";
        cout << "\n";
}
//удаление строки с номером к
int k;
cout << "\nEnter k";
cin >> k;
int** temp = new   int* [n - 1];        /*формирование   новой матрицы*/
for (i = 0; i < n - 1; i++)
        temp[i] = new int[m];
//заполнение новой матрицы
int t;
for (i = 0, t = 0; i < n; i++)
        if (i != k)
        {
                for (j = 0; j < m; j++)
                        *(*(temp + t) + j) = *(*(matr + i) + j);
                t++;
        }
//удаление старой матрицы
for (i = 0; i < n; i++)
        delete matr[i];                 //удаляем строки
delete[]matr;           //удаляем массив указателей
n--;
//печать новой матрицы
for (i = 0; i < n; i++)
{
        for (j = 0; j < m; j++)
                cout << *(*(temp + i) + j) << " ";
        cout << "\n";
}
}
```



Задача 7.4. Удалить из матрицы столбец с номером K.

Методические указания

При подготовке к занятию необходимо изучить:
- способы инициализации указателя;
- операции с указателями;
- правила использования приемов программирования с использованием указателей;
- операции для работы с динамическими данными, особенности их использования.

Аудиторные и домашние задания

1. Найти номера первого нечетного и последнего четного чисел массива.
2. Найти произведение последних N отрицательных элементов массива.
3. Определить величину и индексы максимального по модулю элемента массива.
4. Переставить в массиве элементы с максимальным и минимальным значениями.
5. В массиве все элементы, следующие за минимальным элементом возвести в квадрат.
6. Определить упорядочен ли массив по убыванию.
7. Вычислить произведение положительных элементов массива до появления первого отрицательного элемента.
8. Проверить массив на симметрию.
9. В массиве *A(n)* определить число соседств двух чисел одного знака, причем модуль первого числа должен быть больше модуля второго числа.
10. Для массива *B(18)* вычислить произведение элементов, предшествующих первому наименьшему.
11. Переписать в новый массив элементы, расположенные между максимальным и минимальным элементами массива.
12. Даны целые числа *A(30)*. Пусть *max* – наибольшее, а *min* – наименьшее значения. Получить в порядке убывания все целые числа из интервала (*max*, *min*), которые не входят в исходную последовательность.
13. Даны целые числа X(n). Переписать в новый массив все числа, лежащие в диапазоне [-3,7].
14. Дан массив M(n). Не вводя дополнительного массива, осуществить циклический сдвиг на K элементов влево/
15. Дана квадратная матрица C порядка n. Элементы главной и побочной диагоналей поместить в массив T.
16. Дана квадратная матрица D порядка n. Найти наибольший по модулю элемент верхней треугольной части матрицы.

17. Дана квадратная матрица E порядка n. Выяснить, верно ли, что наибольшее из значений элементов главной диагонали больше, чем наименьшее из значений элементов побочной диагонали.
18. Задана матрица A(n,m). Минимальный элемент каждого столбца заменить суммой положительных элементов этого же столбца.

Контрольные вопросы

1. Назначение указателей?
2. Способы инициализации указателя?
3. Перечислите операции с указателями?
4. Для чего используется оператор разыменования указателя *?
5. Можно ли имя массива использовать в качестве указателя на его первый элемент?
6. С помощью каких операций производится выделение и освобождение динамической памяти?