Лабораторная работа 8. Массивы и указатели в языке C++

1. ЦЕЛЬ РАБОТЫ

Повторить способы объявления и использования массивов в языке C++; повторить понятие «указатель» и операции над указателями, а также принципы использования указателей при работе с массивами.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Массивы. *Массив* по существу является совокупностью однотипных переменных *(элементов* массива), объединенных под одним именем и различающихся своими *индексами.* Массив объявляется подобно простой переменной, но после имени массива указывается число его элементов в квадратных скобках:

```
int myArray[8];
```

Массив, как и переменную, можно инициализировать при объявлении. Значения для последовательных элементов массива отделяются друг от друга запятыми и заключаются в фигурные скобки:

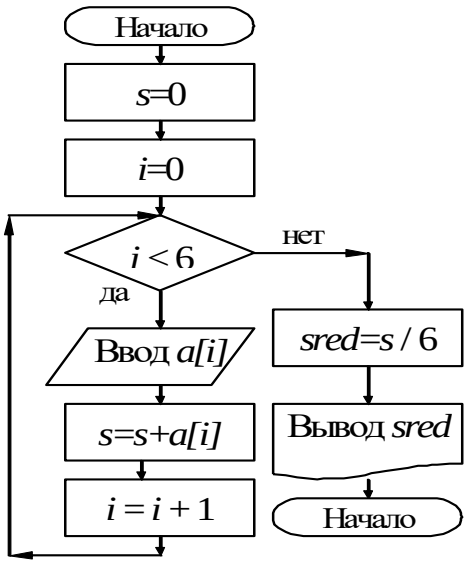
```
int iArray[8] = {7, 4, 3, 5, 0, 1, 2, 6);
```

Обращение к отдельным элементам массива производится путем указания индекса элемента в квадратных скобках, например:

```
myArray[3] = 11;
myArray[i] = iArray[7-i];
```

Индекс должен быть целым выражением, значение которого не выходит за пределы допустимого диапазона. Поскольку индексация массивов начинается в C++ всегда с нуля (т. е. первый элемент имеет индекс 0), то, если массив состоит из N элементов, индекс может принимать значения от 0 до N-1. В языке C++ не предусмотрена автоматическая проверка допустимости значений индекса времени выполнения, поэтому при индексации массивов нужно быть внимательным. Выход индекса за границы массива может приводить к совершенно непредсказуемым результатам. При работе с массивами обычно используют оператор цикла `for`.

Пример программы 1. Ввод с клавиатуры одномерного массива и вычисление среднего арифметического (справа изображена схема алгоритма):




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

int main()
{int i, a[6],s=0; float sred;
for (i=0;i<6;i++)
    {cout <<"Vvedite element N"<<i<< ":";
     cin >> a[i];
```

```
      s+=a[i];
      }
 sred=(float)s/6;
 cout << "Srednee znachenie v massive: " << sred << "\n";
 return 0;
}
```

Указатели. Указатель — это переменная, которая содержит адрес другого объекта. Объявление указателя выглядит так:

```
тип_указываемого_объекта *имя_указателя [= значение];
```

С указателями используются два оператора: «*» и «&». Оператор & возвращает адрес памяти, по которому расположен его операнд. Оператор «*» позволяет обращаться к значению переменной, расположенной по адресу, заданному его операндом.

Например, объявим переменную:

```
int x; // Целое x.
```

Опишем указатель:

```
int *ptr1; // Указатель на целое.
```

После такого описания переменная `ptr1` может принимать значение указателя на величину целого типа. Этой переменной можно присвоить значение так:

```
ptr1=&x;
```

Чтобы получить доступ к объекту, на который указатель ссылается, последний *разыменовывают* (*разадресовывают*), применяя операцию-звездочку.

Единицей изменения значения указателя является размер соответствующего ему типа.

Указатели используются при обработке строк, а также для передачи функциям параметров, значения которых могут ими изменяться (передача по ссылке). Но главное «достоинство» указателей в том, что они позволяют создавать и обрабатывать *динамические структуры данных.* В языке C++ можно выделить память под некоторый объект не только с помощью оператора объявления, но и динамически, во время исполнения программы. Для работы с динамической памятью можно использовать *операции* `new` и `delete`. Выражение с операцией `new` имеет обычно вид:

```
указатель_на_тип = new имя_типа <(инициализатор)>
```

Например:

```
int *ip = new int;
```

При этом создается два объекта: динамический безымянный объект и автоматический или статический указатель с именем `ip`, значением которого является адрес динамического объекта. Можно создать и другой указатель на тот же динамический объект:

```
int *other_p=ip;
```

С другой стороны можно потерять доступ к нашему динамическому объекту, просто присвоив указателю `ip` другое значение:

```
int i; ip = &i;
```

В результате динамический объект будет существовать по-прежнему, но обратиться к нему будет уже невозможно. Другие примеры:

```
int *ip1= new int(1); // дин. объект с инициализацией
int *array = new int[10]; // динамич. массив
double **d = new double*[j]; // дин. массив указателей
            //на double. Число элементов равно j.
```

В случае успешного завершения операция `new` возвращает указатель со значением отличным от нуля. Иначе – `NULL`, нулевой указатель.

Операция `delete` освобождает участок памяти, выделенный ранее операцией `new`.

Указатели и массивы. Между указателями и массивами в C++ существует тесная связь. Имя массива без индекса эквивалентно указателю на его первый элемент. Поэтому можно написать:

```
int iArray[4] ;
int *piArr;
piArr=iArray; //piArr указывает на начальный элемент iArray.
```

Последнее эквивалентно

```
piArr = &iArray[0];
```

И наоборот, указатель можно использовать подобно имени массива, т. е. индексировать его. Например, `piArr[3]` представляет четвертый элемент массива `iArray[]`.

Таким образом, в выражениях с указателями и массивами можно обращаться одинаково. Следует только помнить, что объявление массива выделяет память под соответствующее число элементов, а объявление указателя никакой памяти не выделяет, вернее, выделяет память для хранения значения указателя — некоторого адреса. Компилятор по-разному рассматривает указатели и массивы, хотя внешне они могут выглядеть очень похоже.

Пример программы 2:

```
#include <iostream>

using namespace std;

int main()
{int i, *ip, N, S=0;
 setlocale(LC_ALL,"Russian");
 cin >> N;
 ip=new int[N];
 if (ip != NULL)
    {cout << "Массив создан успешно. \n";
     for (i=0;i<N;i++)
         {cout << "Введите " << i <<"-й элемент: ";
          cin >> *(ip+i);
          S+=ip[i];
          }
     cout << "Сумма элементов: "<< S << "\n";
```

```
      delete ip;
      }
 else cout << "Ошибка выделения динамической памяти! \n";
 return 0;
}
```

Пример программы 3 - пример создания двумерного динамического массива размером N на M:

Вариант 1 (создается один указатель на начало массива):

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;
int main()
{
      int i, j, * ip, N, M, S = 0;
      cin >> N; cin >> M;   // запрашиваем размеры массива
      ip = new int[N * M];        // выделение памяти
      if (ip != NULL)
      {
            cout << "\n********** Massiv ********** \n";
            srand((unsigned)time(NULL));
            for (j = 0; j < M; j++)
            {
                  for (i = 0; i < N; i++)
                  {
                        *(ip + i + j * N) = rand() % 100;
                        cout << *(ip + i + j * N) << "   ";
                        S += *(ip + i + j * N);//или можно так: ip[i+j*N];
                  }
                  cout << "\n";
            }
            cout << "\nSumma: " << S << "\n";
            delete[] ip;
      }
      else cout << "Error ! \n";
      return 0;
}
```

Вариант 2 (создается указатель на массив указателей на одномерные массивы (строки матрицы)):

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

int main()
{
      int i, j, ** ip, N, M, S = 0;
```

```
	cin >> N; cin >> M;   // запрашиваем размеры массива
	ip = new int*[M];     // выделение памяти под указатели на строки
	if (ip != NULL)
	{
		cout << "\n********** Massiv ********** \n";
		srand((unsigned)time(NULL));
		for (j = 0; j < M; j++)
		{
		  ip[j] = new int[N];    // выделение памяти под очередную строку
		  for (i = 0; i < N; i++)
		      {
		       ip[j][i] = rand() % 100;
		       cout << ip[j][i] << "   ";
		       S += ip[j][i];
		      }
		      cout << "\n";
		}
		cout << "\nSumma: " << S << "\n";
		// удаление двумерного динамического массива
		for (j = 0; j < M; j++)
		      delete[]ip[j];
	}
	else cout << "Error ! \n";
	return 0;
}
```

3. *ЗАДАНИЕ НА РАБОТУ*

1. Проверить работу программ, приведенных в кратких теоретических положениях.

2. Разработать самостоятельно приложения для решения задач по своему варианту.

4. *ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ*

Задание 1. Ознакомиться с теоретическим материалом, приведенным в пункте «Краткие теоретические положения» данных методических указаний, а также с конспектом лекций и рекомендуемой литературой по данной теме.

Задание 2. Проверить работу примеров, приведенных в тексте МУ.

Задание 3. Изменить один из вариантов программы из примера 3 так, чтобы в массиве находился столбец с максимальной суммой элементов.

Задание 4. Разработать приложение для создания и обработки одномерного массива целых случайных чисел по своему варианту (см. таблицу 1).

Задание 5. Разработать 2 программы с использованием динамического выделения памяти для хранения массивов данных по вариантам, указанным в таблице 2.

Показать результаты работы преподавателю. Оформить отчет по работе.

5. ОФОРМЛЕНИЕ ОТЧЕТА

Отчет по работе должен содержать:

- название и цель работы;
- номер варианта;
- для задач по своему варианту – текст задачи, текст кода программы, схема алгоритма программы, результаты выполнения разработанной программы для разных наборов исходных данных.

6. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шилдт Г. C++: базовый курс, 3-е издание. : Пер. с англ. – М.: «Издательский дом «Вильямс», 2005. – 624 с.
2. Пахомов Б.И. C/C++ и MS Visual C++ для начинающих. – СПб.: БХВ-Петербург, 2008. – 624 с.
3. Златопольский Д.М. Сборник задач по программированию. - 3-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2011. - 304 с.
4. http://cppstudio.com/