Федеральное агентство связи

Федеральное государственное бюджетное образовательное

учреждение высшего образования

Поволжский государственный университет

телекоммуникаций и информатики

Лабораторная работа №1

По дисциплине «Алгоритмы и структуры данных»

Тема: «Массивы автоматические, статические и динамические»

Выполнил студент

группы ИСТ-82 Логинов Ю.

Проверил

Назаренко П. А.

Самара 2021

**Цель работы заключается в изучении:**

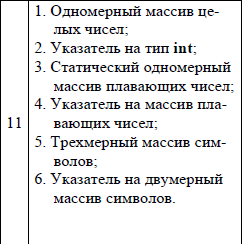
– свойств одномерных и многомерных массивов как линейных и прямоугольных структур данных на примере одного из основных языков программирования;

– свойств автоматических, статических и динамических массивов;

– принципов использования массивов, в том числе динамических;

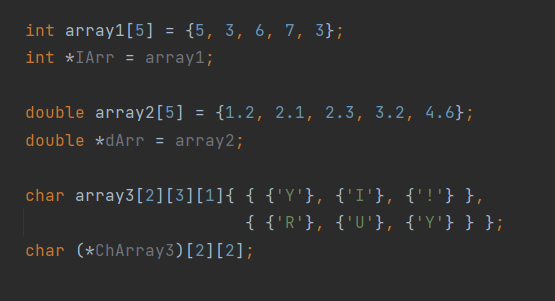
– принципов использования указателей, в том числе для работы с массивами.

Вариант 11



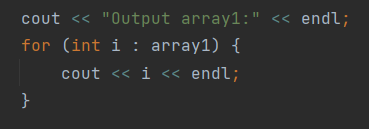
***Порядок выполнения работы***

1. Определить в функции main() переменные и массивы по таблице 1 в соответствии с вариантом (см. таблицу 1).

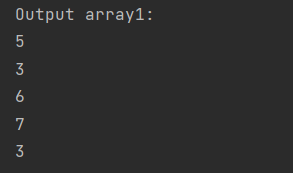


2. В функции main() выполнить следующие действия:

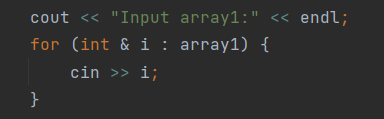
2.1. Проверить содержимое массива №1 (с помощью цикла for и операции вывода cout<<).



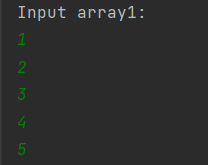
Result:



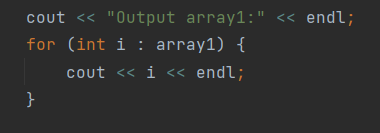
2.2. Ввести данные в массив №1 (с помощью цикла for и операции ввода cin>>).

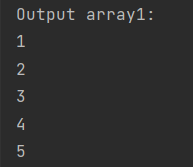


Result:



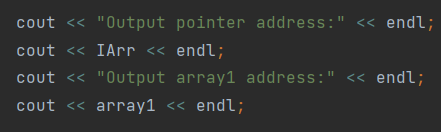
2.3. Еще раз проверить содержимое этого массива, сделать выводы.

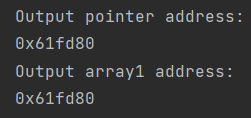




**Вывод значений массива показал введённые в предыдущем задании данные.**

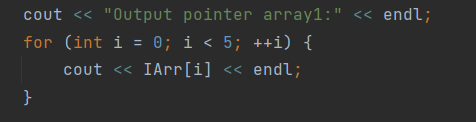
2.4. Присвоить указателю №2 адрес массива №1, вывести на экран адреса массива и указателя и содержимое указателя. Сделать выводы.

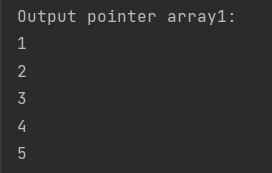




**Массив и указатель имеют одинаковый адрес.**

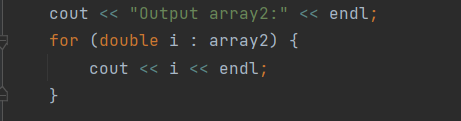
2.5. Повторить пункт 3 для указателя, содержащего адрес массива. Сделать выводы.

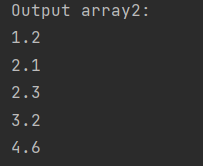


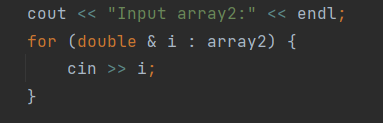


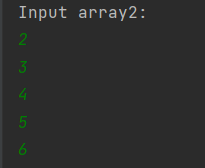
**Указатель содержит те же данные, что и массив.**

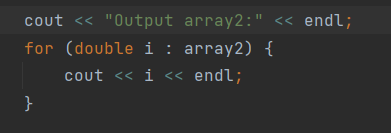
2.6. Повторить пункты 1 – 3 для статического массива №3. Сделать выводы.

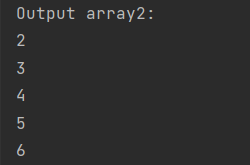




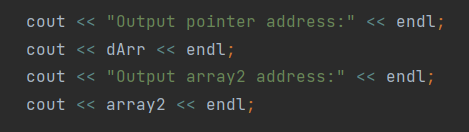


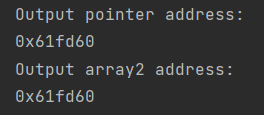




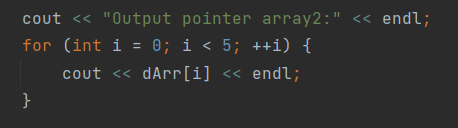


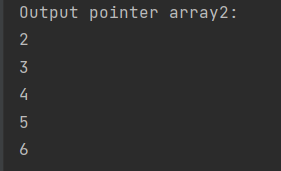
**Вывод значений массива показал введённые в предыдущем задании данные.**

****



**Массив и указатель имеют одинаковый адрес.**

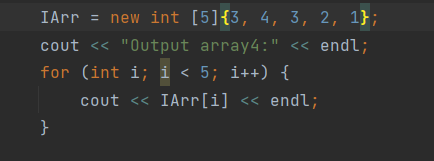
****

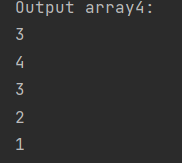


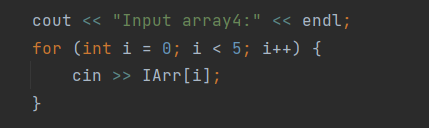
**Указатель содержит те же данные, что и массив.**

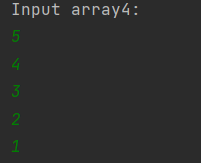
2.7. Используя имеющийся указатель №2, создать динамический массив и повторить для него пункты 1 – 3. Сделать выводы.

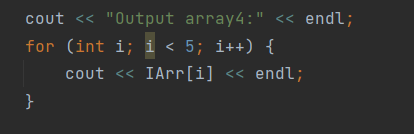
Объявление динамического массива:

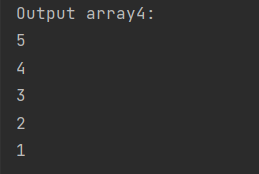






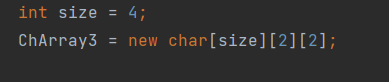


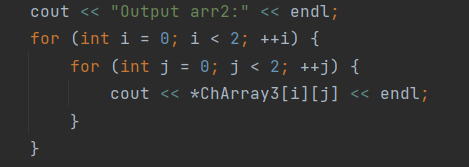


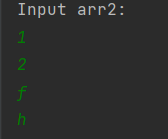


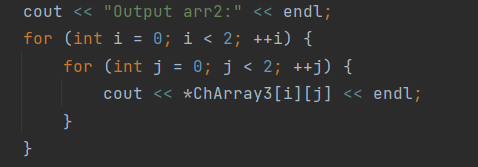
**Вывод значений массива показал введённые в предыдущем задании данные.**

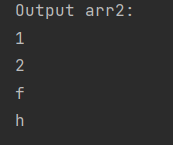
2.8. Удалить динамический массив. Используя указатель №4, создать двумерный массив и повторить для него пункты 2, 3. Сделать выводы. Удалить двумерный динамический массив.





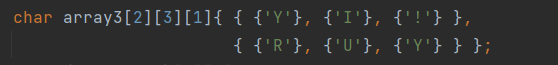


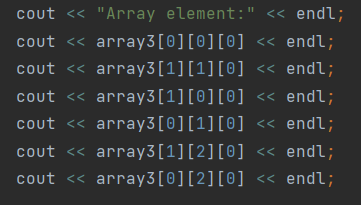


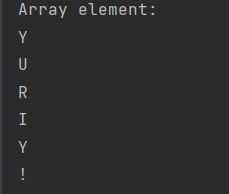


**Вывод значений массива показал введённые в предыдущем задании данные.**

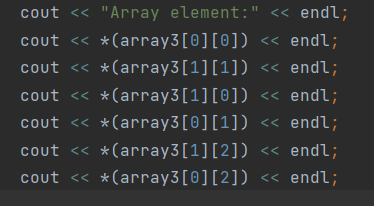
2.9. Вывести на экран любой из элементов трехмерного массива №5, используя операцию индексации.

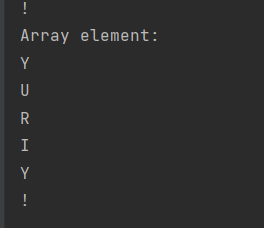




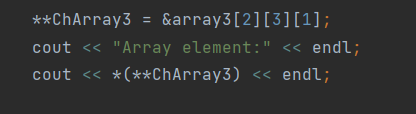


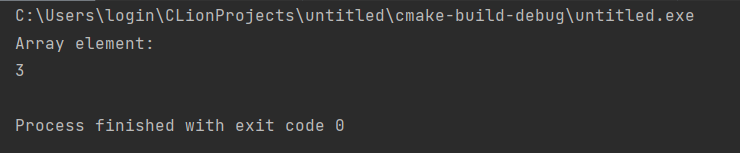
2.10. Повторить пункт 9, используя имя массива как указатель и операцию доступа по указателю.





2.11. Присвоить указателю №6 на двумерный массив адрес трехмерного массива №5. Повторить для этого указателя пункт 10. Сделать выводы.





**Вывод: в данной работе мы изучили свойства: одномерных и многомерных массивов, статических и динамических массивов.**

**Вывод**: в данной работе мы изучили свойства одномерных и двумерных массивов, свойства автоматических, статических и динамических массивов, принципы использования массивов и указателей.

Листинг программы:

#include <iostream>  
  
int main() {  
 using namespace std;  
  
 int array1[5] = {5, 3, 6, 7, 3};  
 int \*IArr = array1;  
  
 double array2[5] = {1.2, 2.1, 2.3, 3.2, 4.6};  
 double \*dArr[5];  
  
 char array3[2][3][1]{{{'Y'}, {'I'}, {'!'}},  
 {{'R'}, {'U'}, {'Y'}}};  
 char \*(ChArray3[2][2]);  
  
  
 // массив целых чисел  
 cout << "Output array1:" << endl;  
 for (int i: array1) {  
 cout << i << endl;  
 }  
  
 cout << "Input array1:" << endl;  
 for (int &i: array1) {  
 cin >> i;  
 }  
  
 cout << "Output array1:" << endl;  
 for (int i: array1) {  
 cout << i << endl;  
 }  
  
 cout << "Output pointer address:" << endl;  
 cout << IArr << endl;  
 cout << "Output array1 address:" << endl;  
 cout << array1 << endl;  
  
 cout << "Output pointer array1:" << endl;  
 for (int i = 0; i < 5; ++i) {  
 cout << IArr[i] << endl;  
 }  
  
 // массив чисел с плавующей точкой  
 cout << "Output array2:" << endl;  
 for (double i: array2) {  
 cout << i << endl;  
 }  
  
 cout << "Input array2:" << endl;  
 for (double &i: array2) {  
 cin >> i;  
 }  
  
 cout << "Output array2:" << endl;  
 for (double i: array2) {  
 cout << i << endl;  
 }  
  
 cout << "Output pointer address:" << endl;  
 cout << dArr << endl;  
 cout << "Output array2 address:" << endl;  
 cout << array2 << endl;  
  
 cout << "Output pointer array2:" << endl;  
 for (int i = 0; i < 5; ++i) {  
 cout << dArr[i] << endl;  
 }  
  
  
 IArr = new int[5]{3, 4, 3, 2, 1};  
 cout << "Output array4:" << endl;  
 for (int i = 0; i < 5; i++) {  
 cout << IArr[i] << endl;  
 }  
  
 cout << "Input array4:" << endl;  
 for (int i = 0; i < 5; i++) {  
 cin >> IArr[i];  
 }  
  
 cout << "Output array4:" << endl;  
 for (int i = 0; i < 5; i++) {  
 cout << IArr[i] << endl;  
 }  
 delete[] IArr;  
  
 cout << "Array element:" << endl;  
 cout << array3[0][0][0] << endl;  
 cout << array3[1][1][0] << endl;  
 cout << array3[1][0][0] << endl;  
 cout << array3[0][1][0] << endl;  
 cout << array3[1][2][0] << endl;  
 cout << array3[0][2][0] << endl;  
  
 cout << "Array element:" << endl;  
 cout << \*(array3[0][0]) << endl;  
 cout << \*(array3[1][1]) << endl;  
 cout << \*(array3[1][0]) << endl;  
 cout << \*(array3[0][1]) << endl;  
 cout << \*(array3[1][2]) << endl;  
 cout << \*(array3[0][2]) << endl;  
  
  
 \*\*ChArray3 = &array3[2][3][1];  
 cout << "Array element:" << endl;  
 cout << \*(\*\*ChArray3) << endl;  
  
 return 0;  
}