Лабораторная №5. Одномерные массивы.

Темы, рассматриваемые в лабораторной работе:

Основные алгоритмы обработки массивов. Статические массивы. Массивы - параметры функций.

Некоторые вспомогательные материалы по теме. Студентам рекомендуется рассмотреть самостоятельно и, возможно, выполнить некоторые из них для проверки и понимания.

Статические массивы в языке C++ могут быть объявлены следующим образом:

```
int arr[10];
```

Длину статического массива может задавать только константа или выражение, вычисляемое во время компиляции программы.

```
#define MAX_LEN 256
int arr[MAX_LEN]; // можно задать длину массива макросом

const int n = 10;
int arr2[n]; // также можно использовать константные переменные
```

Массивам можно присваивать значения при их инициализации:

```
const int n = 5;
int arr[n] = {1, 2, 3, 4, 5};

int arr2[] = {1, 2, 3, 4, 5}; // или же можно без указания длины массива
```

К элементам массива можно обращаться через индекс или указатель:

```
int a[] = {1, 2, 3, 4, 5};
cout << a[2] << endl;
cout << *(a + 2) << endl;
```

Перенос значений из одного массива в другой необходимо производить поэлементно или копируя память целиком:

```
int arr[] = {1, 2, 3, 4, 5};
int arr_copy[5];

for (int i = 0; i < 5; ++i)// цикл по индексам элементов
    arr_copy[i] = arr[i]; // поэлементный перенос
for (int i = 0; i < 5; ++i)
    cout << arr_copy[i]  << " "; // вывод элементов на консоль
cout << endl;

int arr_copy_mem[5];

//копия непрерывного участка памяти,
//занимаемого массивом (необходимо подключить библиотеку
#include <cstring>)
memcpy(arr_copy_mem, arr, sizeof(int) * 5);
```

```
for (auto element : arr_copy_mem) // цикл по самим элементам
    cout << element << " "; // вывод элементов на консоль
cout << endl;
```

В качестве аргумента функций массивы передаются в виде указателя:

```
void copy_arr(int* dst, const int* src, int size) {
// функция копирования массива
    for (int i = 0; i < size; ++i)
        dst[i] = src[i];
}

int arr[] = {1, 2, 3, 4, 5};
int arr_copy[5];

copy_arr(arr_copy, arr, 5);
```

Пример функции поиска максимума в одномерном массиве целых чисел:

```
int max_of_arr(const int* a, int size) {
    int max_elem = a[0];
    for (int i = 1; i < size; ++i)
        if (a[i] > max_elem)
            max_elem = a[i];
    return max_elem;
}

int a[] = {1, 2, 3, 9, 0, 4, 5};
cout << "Max: " << max_of_arr(a, sizeof(a) / sizeof(int)) << endl;
// зная размер занимаемой памяти и размер одного элемента,
// можно рассчитать длину статического массива
```

При написании тестов к функциям, меняющим значения элементов массива, следует задавать массив до выполнения функции и массив после, используя утверждение Assert в цикле для каждой пары значений. Например, тест для функции сортировки buble_sort целочисленного массива:

```
TEST_METHOD(TestSort){
 int u_sort[]={7,3,-1,0,2};
 int sort[]={-1,0,2,3,7};
 buble_sort(u_sort,5);
 for (int i=0;i<5;++i)
  Assert::AreEqual(u_sort[i],sort[i]);
}
```

Задания:

Решение каждой задачи должно сопровождаться представительным набором тестов.

1. Реализовать функцию для вычисления суммы элементов одномерного массива.
2. Реализовать функцию поэлементного умножения значений двух массивов одинакового размера (результат записать в третий массив, передаваемый в функцию).
3. Реализовать функцию сортировки массива методом вставки.

4. Реализовать функцию проверки свойства массива – «в массиве нет нулевых элементов»
5. Реализовать функцию проверки свойства массива – «в массиве нет стоящих рядом одинаковых элементов».

Дополнительные задания:

6. Реализовать функции проверки свойства целочисленного массива, если условие проверки передается как логическая функция с одним или двумя параметрами (одноместный и двуместный предикат). Примеры проверяемых свойств: все элементы положительные, все элементы четные, все элементы содержат в десятичной записи цифру 5, два элемента имеют разные/одинаковые знаки, в паре элементов всегда первый меньше или равен второго.