Лабораторная №5. Одномерные массивы.

Темы, рассматриваемые в лабораторной работе:

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

Вспомогательные материалы по теме:

Студентам рекомендуется рассмотреть данные примеры самостоятельно и, возможно, выполнить некоторые из них для проверки и понимания.

Статические массивы (их размер не меняется в процессе выполнения программы) 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 arr1[n]; // также можно использовать константные переменные
int arr2[2*n];
```

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}; // можно без указания длины массива
```

В последнем случае размер массива может быть определен следующим образом

```
// зная размер занимаемой памяти и размер одного элемента,
// можно рассчитать длину статического массива
int size_of_arr= sizeof(arr2) / sizeof(int);
```

К элементам массива можно обращаться через индекс или указатель:

```
int a[] = {1, 2, 3, 4, 5};
cout << a[2] << endl;
cout << a[0] << endl;
cout << *(a + 2) << endl;
cout << *a << endl;
```

Ввод и вывод элементов массива осуществляется поэлементно

```
int arr[5];
for(int i=0; i<5; ++i) {
  cout<<i<<"-й элемент массива";
  cin>>arr[i];
}
```

```
for (int i = 0; i < 5; ++i)
    cout << arr[i]  << " ";
cout << endl;
```

Рекомендуется оформить ввод и вывод в виде вспомогательных функций. Параметры массивы передаются в функции указателем.

```
void read_arr(int *a, int n){
 for(int i=0; i<n; ++i) {
   cout<<i<<"-й элемент массива";
   cin>>a[i];
 }
}
void print_arr(const int *a,int n){
     cout<<"a[";
     for (int i = 0; i < n; ++i)
             cout << a[i]  << ", ";
     cout <<"]"<< endl;
}
```

Перенос значений из одного массива в другой необходимо производить поэлементно:

```
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;
}
```

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 поиска элемента с заданным значением в массиве. Результат поиска – индекс первого вхождения элемента в массив или «-1».
4. Реализовать функцию сортировки массива методом вставки.
5. Реализовать функцию проверки свойства массива – «в массиве нет нулевых элементов»
6. Реализовать функцию проверки свойства массива – «в массиве нет стоящих рядом одинаковых элементов».

Дополнительные задания:

7. 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.