|  |  |
| --- | --- |
| Отчет за лабораторную работу № 7 | Выполнил:  Студент группы ВИС13  Скибицкий Артём Сергеевич |

**Тема:** Одномерные массивы как аргументы функций.

**Цель работы:** Освоить технику передачи в функции и возврата из функций массивов значений. Использование директивы define и статических переменных для управления работой программы.

**Задание 1**

**Код программы:**

#include <iostream>

using namespace std;

void get\_arr(int arr[], int len);

void print\_arr(int arr[], int len);

#define ARR\_SIZE 7

int main()

{

    int arr[ARR\_SIZE];

    get\_arr(arr, ARR\_SIZE);

    cout << "Elements have got their values\n";

    print\_arr(arr,ARR\_SIZE);

}

void get\_arr(int arr[], int len)

{

    cout << "Enter array values:\n";

    for (int i = 0; i < len; i++)

    {

        printf("arr[%i] = ", i);

        cin >> arr[i];

    }

}

void print\_arr(int arr[], int len)

{

    for (int i = 0; i < len; i++)

    {

        printf("arr[%i] = %i\n", i, arr[i]);

    }

}

**Вывод:**

Enter array values:

arr[0] = 1

arr[1] = 2

arr[2] = 3

arr[3] = 4

arr[4] = 5

arr[5] = 6

arr[6] = 7

Elements have got their values

arr[0] = 1

arr[1] = 2

arr[2] = 3

arr[3] = 4

arr[4] = 5

arr[5] = 6

arr[6] = 7

**Пояснение:** Для передачи массива как аргумента используется передача указателя на него. Для этого применятся конструкция int arr[] в прототипе функций. Аргумент имеет для этого параметра имеет вид arr, так при условии, что arr – имя массива, тогда arr сам по себе – указатель. Так как при передаче указателя на массив невозможно получить количество элементов самого массива, добавлен параметр – len, отвечающий за количество элементов.

**Задание 2**

**Задание 2.1**

**Код программы:**

#include <iostream>

using namespace std;

void get\_arr(int arr[], int len);

bool sorted(int arr[], int len);

#define ARR\_LEN 10

int main()

{

    int x[ARR\_LEN] = {};

    get\_arr(x, ARR\_LEN);

    cout << sorted(x, ARR\_LEN);

}

void get\_arr(int arr[], int len)

{

    for (int i = 0; i < len; i++)

    {

        printf("arr[%i] = ", i);

        cin >> arr[i];

    }

}

bool sorted(int arr[], int len)

{

    for (int i = 1; i < len; i++)

    {

        if (arr[i] < arr[i - 1])

            return false; // досрочный выход если текущий меньше предыдущего

    }

    return true;

}

**Вывод:**

arr[0] = 1

arr[1] = 2

arr[2] = 3

arr[3] = 4

arr[4] = 5

arr[5] = 6

arr[6] = 7

arr[7] = 8

arr[8] = 9

arr[9] = 10

1

**Вывод при несортированном вводе:**

arr[0] = 0

arr[1] = 1

arr[2] = 2

arr[3] = 34

arr[4] = 5

arr[5] = 4334

arr[6] = 7784

arr[7] = 842

arr[8] = 11

arr[9] = 80

0

**Пояснение:** Получаем от пользователя массив, проходимся по нему, сравнивая каждый текущий элемент с предыдущим, в случае если он (текущий) меньше предыдущего – досрочно возвращаем ложь, иначе – доходим до конца – возвращаем истину.

**Задание 2.2**

**Код программы:**

#include <iostream>

using namespace std;

int find(int arr[], int len, int x);

#define ARR\_LEN 10

int main()

{

int arr[ARR\_LEN] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int x = 0;

cout << "x = ";

cin >> x;

cout << find(arr, ARR\_LEN, x);

}

int find(int arr[], int len, int x)

{

for (int i = 0; i < len; i++)

{

if (arr[i] == x)

return i;

}

return -1;

}

**Вывод при x = 10:**

x = 10

-1

**При x = 3:**

x = 3

3

**Пояснение:** Досрочный выход при совпадении, возвращаем индекс искомого числа, иначе возвращаем -1.

**Примечание:** Здесь и далее функции get\_arr(), print\_arr() использоваться не будут, заменены на заранее определенные массивы. (Если явно не указана необходимость).

**Задание 2.3**

**Код программы:**

#include <iostream>

using namespace std;

double twoThree(double arr[], int len); // 2\_3

int main()

{

    double x[10] = {3, -5, 7, 0, -2, 9, -8, 4, -1, 6};

    cout << twoThree(x, sizeof(x) / sizeof(x[0]));

}

double twoThree(double arr[], int len)

{

    int m = 0; // По умолчанию 0 (если нет отрицательных элементов)

    // Поиск первого отрицательного элемента

    for (int i = 0; i < len; ++i)

    {

        if (arr[i] < 0)

        {

            m = i;

            break;

        }

    }

    double sum = 0.0;     // сумма

    double product = 1.0; // текущее произведение

    for (int i = 0; i <= m; ++i)

    {

        product \*= arr[i];

        sum += product;

    }

    return sum;

}

**Вывод:**

-12

**Пояснение:** Функция проходит по массиву чтобы найти первый отрицательный элемент. Если такой элемент найден, m устанавливается равным его индексу. Если отрицательных элементов нет, m остаётся 0. Переменная product хранит текущее произведение элементов (начинается с 1.0). На каждом шаге цикла (от 0 до m включительно) произведение умножается на элемент x[i], а результат добавляется к сумме sum.

**Задание 2.4**

**Код программы:**

#include <iostream>

using namespace std;

double twoFour(double arr[], int len);

int main()

{

    double y[] = {3, -5, 7, 0, -2, 9, -8, 4, -1, 6, -10, 2, 5, -3, 1, -7, 8, -4, 10, -6};

    cout << twoFour(y, sizeof(y) / sizeof(y[0]));

}

double twoFour(double arr[], int len)

{

    double maxSum = arr[0] + arr[len - 1];

    // Проходим по всем симметричным парам массива

    for (int i = 1; i < len / 2; i++)

    {

        double currentSum = arr[i] + arr[len - 1 - i];

        if (currentSum > maxSum)

        {

            maxSum = currentSum;

        }

    }

    // Если количество элементов нечетное, сравниваем средний элемент \* 2 с максимальной парой

    if (len % 2 != 0 && 2 \* arr[len / 2] > maxSum)

    {

        maxSum = 2 \* arr[len / 2];

    }

    return maxSum;

}

**Вывод:**

10

**Пояснение:** Функция проходит по всем симметричным парам массива и ищет максимальную. Если число элементов нечетное, то центральный элемент умножаем на 2 и сравниваем с максимальной парой. Результат – максимальная сумма пары.

**Задание 2.5**

**Код программы:**

#include <iostream>

using namespace std;

double twoFive(double arr[], int len);

int main()

{

    double y[] = {3, -5, 7, 0, -2, 9, -8, 4, -1, 6, -10, 2, 5, -3, 1, -7, 8, -4, 10, -6};

    cout << twoFive(y, sizeof(y) / sizeof(y[0]));

}

double twoFive(double arr[], int len)

{

    double min\_ = \_\_DBL\_MAX\_\_;

    for (int i = 1; i < len; ++i)

    {

        if (min\_ > arr[i - 1] \* arr[i])

        {

            min\_ = arr[i - 1] \* arr[i];

        }

    }

    return min\_;

}

**Вывод:**

-72

**Пояснение:** Функция проходит по парам чисел и ищет минимальную пару. Макрос \_\_DBL\_MAX\_\_ - максимальное число для типа double.

**Задание 2.6**

**Код программы:**

#include <iostream>

using namespace std;

double twoSix(double arr[], int len);

int main()

{

double y[] = {3, -5, 7, 0, -2, 9, -8, 4, -1, 6, -10, 2, 5, -3, 1, -7, 8, -4, 10, -6};

cout << twoSix(y, sizeof(y) / sizeof(y[0]));

}

double twoSix(double arr[], int len)

{

double negatives[len] = {};

double positives[len] = {};

int q = 0, p = 0;

// Разделение элементов на отрицательные и положительные

for (int i = 0; i < len; ++i)

{

if (arr[i] < 0)

{

negatives[q] = arr[i];

q++;

}

else if (arr[i] > 0)

{

positives[p] = arr[i];

p++;

}

}

int k = min(p, q);

// Вычисление суммы произведений

double sum = 0.0;

for (int i = 0; i < k; ++i)

{

sum += negatives[i] \* positives[p - 1 - i];

}

return sum;

}

**Вывод:**

-223

**Пояснение:**  В функции создается два временных массива – один для положительных другой для отрицательных чисел. Затем передаваемый массив разделяется на 2, находится k = min(p,q) и вычисляется сумма. Для разворота массива положительных чисел используется конструкция [p - 1 - i].