

Министерство образования и науки РФ  
ФГБОУ ВО «ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
Кафедра «Информационная безопасность систем и технологий»

Отчёт  
о практической работе №1  
Функции и массивы

Дисциплина: Языки программирования

Группа: 18ПИ1

Выполнил: Нестеров И.С.

Количество баллов:

Дата сдачи:

Принял: к.т.н., доцент М.Ю. Лупанов

Пенза 2019

1 Цель работы:

Освоить работу с массивами и указателями в программах на языке Си++.

Освоить создание и использование функций в программах на языке Си++.

2 Задание:

2.1 Создайте проект и поместите в него программу из приложения Г. Внесите изменения в код в соответствии с вашей группой и номером зачетной книжки. Постройте проект. Запустите программу. Выполните проверку на ошибки работы с памятью с помощью дополнения MemCheck. Проанализируйте результаты и исправьте ошибки в программе. Постройте проект. Запустите программу. Сравните результат ее работы с результатом, полученным до проверки памяти.

2.2 Разработайте и реализуйте функцию нормировки значений массива чисел с плавающей точкой на диапазон от 0 до 1 по формуле  $y(x) = (x - x_{\min}) / (x_{\max} - x_{\min})$ . Нормировку производить «in place», т. е. в исходном массиве. Разработайте и реализуйте программу для проверки корректности работы функции нормировки на различных наборах данных.

2.3 Измените функцию нормировки значений массива чисел с плавающей точкой, разработанную при выполнении задания 2, так, чтобы нормированные значения помещались в новый массив. Функция должна возвращать указатель на этот массив. Разработайте и реализуйте программу для проверки корректности работы функции нормировки на различных наборах данных.

2.4 Выполните проверку программ, реализованных при выполнении заданий 2 и 3, на корректность работы с памятью с помощью дополнения MemCheck. В случае обнаружения ошибок проведите их анализ и внесите исправления в программы.

2.5 Задание повышенной сложности. Разработайте и реализуйте программу для шифрования перестановочным шифром Скитала [4]. Зашифрование и расшифрование реализовать в виде отдельных функций. В

качестве ключа шифрования использовать длину окружности Скитала (в символах). Для функции зашифрования входные данные: строка текста, ключ. Строка текста должна состоять только из прописных символов английского алфавита, при несоответствии сигнализировать об ошибке. Ключ — целое положительное число, не превышающее половины длины текста, при несоответствии сигнализировать об ошибке. Выходные данные: строка шифротекста. Для функции зашифрования входные данные: строка шифротекста, ключ. Строка шифротекста должна состоять только из прописных символов английского алфавита, при несоответствии сигнализировать об ошибке. Ключ — целое положительное число, не превышающее половины длины шифротекста, при несоответствии сигнализировать об ошибке. Выходные данные: строка текста.

2.6 Задание повышенной сложности. Выполните проверку программ, реализованную при выполнении задания 5, на корректность работы с памятью с помощью дополнения MemCheck. В случае обнаружения ошибок проведите их анализ и внесите исправления в программу.

2.7 . Задание повышенной сложности. Разработайте тестовые наборы входных данных. Проверьте программу на корректность выполняемых преобразований с помощью этих наборов данных.

### 3 Результаты работы:

3.1 Был создан проект, в него был помещён код программы из приложения Г. В код были внесены изменения в соответствии с личными данными. После построения проекта была выполнена проверка программы на ошибки работы с памятью с помощью дополнения MemCheck. Ошибки, обнаруженные MemCheck показаны на рисунке 1.

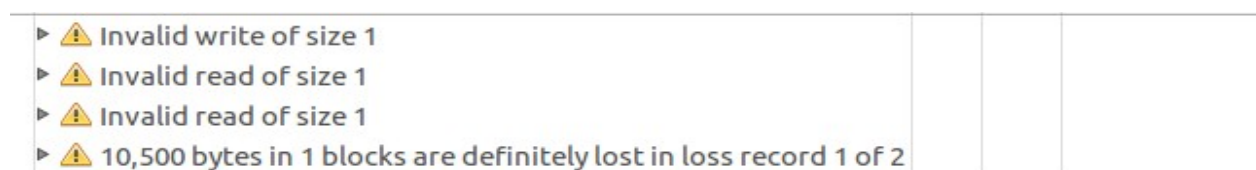


Рисунок 1 — Обнаруженные ошибки

Ошибка «Invalid write of size 1» возникает из-за того, что программа выходит за границу массива. Для ее исправления был уменьшен индекс массива. Для исправления ошибки «Invalid read of size 1» было изменен порядок строк в программе и в строке 29 уменьшен индекс на 1. Строки 28 и 30 поменялись местами, так как программа должна сначала прочитать значение, а только потом очистить память. Для исправления последней ошибки была очищена память, выделяющиеся под массив pT. Исправленный код представлен ниже:

```
#include <iostream>
#include <new>
using namespace std;
// раскомментируй в зависимости от твоей группы
#define PI1
// #define PT1
// #define PT2
// вместо нуля поставь последние три цифры из номера
зачетки
#define NUM 112
#define SIZE (NUM*100)
int main()
{
    #if NUM==0
    #error NUM not properly defined
    #endif
    #ifndef PI1
        typedef char T;
    #elif defined PT1
        typedef int T;
    #elif defined PT2
        typedef double T;
    #else
    #error Group not selected
    #endif
    T * pT = new T [SIZE];
    T * pT2 = new T [100];
    pT[SIZE-1] = 65 + SIZE % 26;
    pT2[10] = 65 + NUM / 10.0;
    cout<<pT[SIZE-1]<<endl;
    cout<<pT2[10]<<endl;
```

```

        delete[] pT;
        delete[] pT2;
        return 0;
    }

```

3.2 Была разработана программа для нормировке массива в диапазон от 0 до 1. Параметрами функции являются указатель на массив и размер массива. Сначала выполняется проверка и нахождение максимального значения и минимального с помощью ветвления если и иначе. После запускается цикл, в котором происходит непосредственно сама нормировка. Алгоритм работы представлен на рисунке 2.

На основе разработанного алгоритма был реализован код программы:

```

void normaliz(double * m, int razmer)
{
    double max = m[0];
    double min = m[0];
    for (int i = 1; i<razmer; i++) {
        if (m[i]>max)
            max=m[i];
        if (m[i]<min)
            min=m[i];
    }
    cout<<"Minimum = ";
    cout<<min<<endl;
    cout<<"Maximum = ";
    cout<<max<<endl;
    for (int i = 0; i<razmer; i++) {
        m[i]= (m[i]-min)/(max-min);
    }
}

```

Далее была разработана программа для проверки полученной функции. Для этого программа на вход получает целочисленный массив с известным количеством элементов, после чего происходит нормализация и в конце были выведены полученные результаты. Алгоритм представлен на рисунке 3.



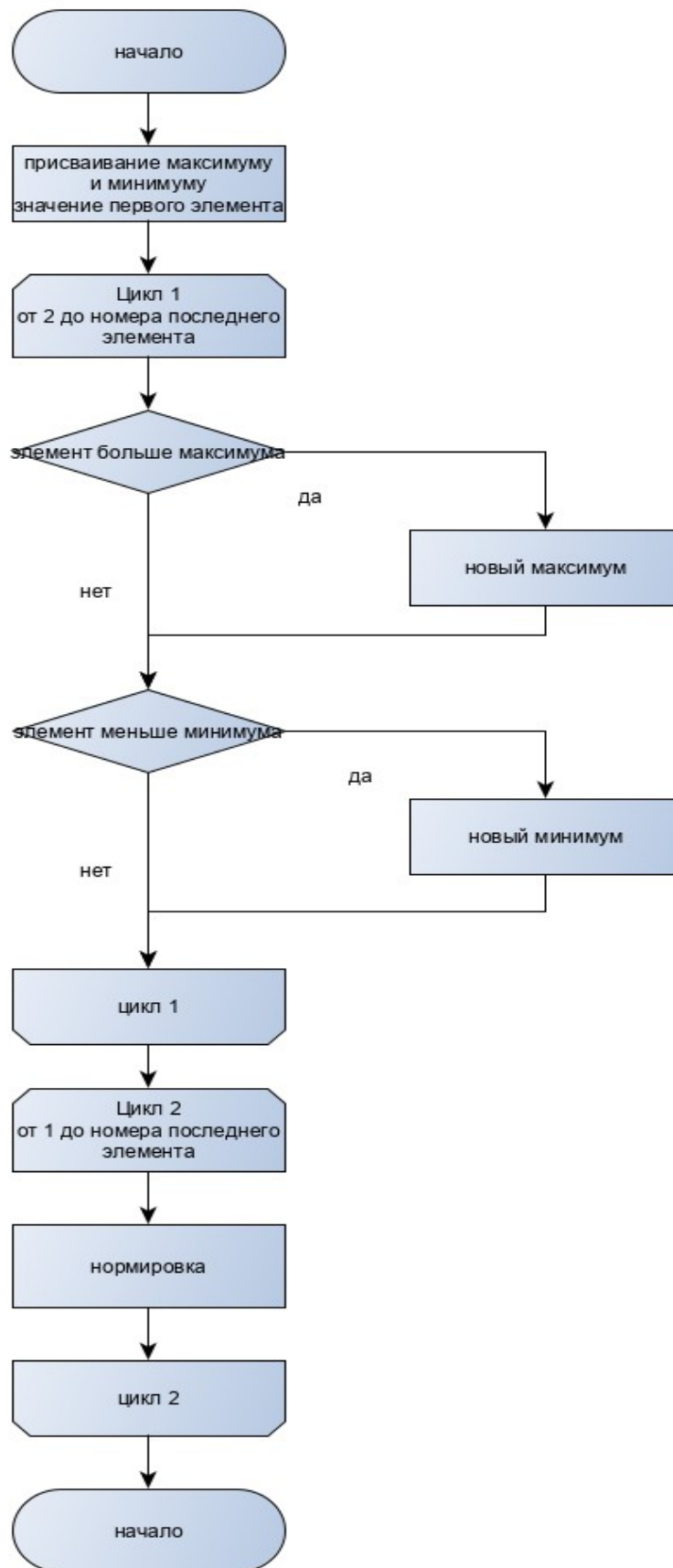


Рисунок 2 — Алгоритм функции нормализации

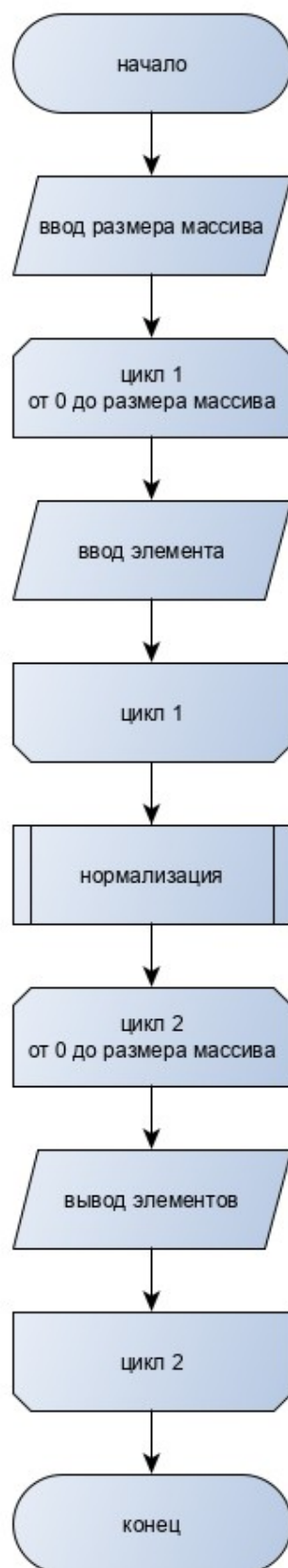


Рисунок 3 — Алгоритм проверки функции нормализации  
Код программы проверки:

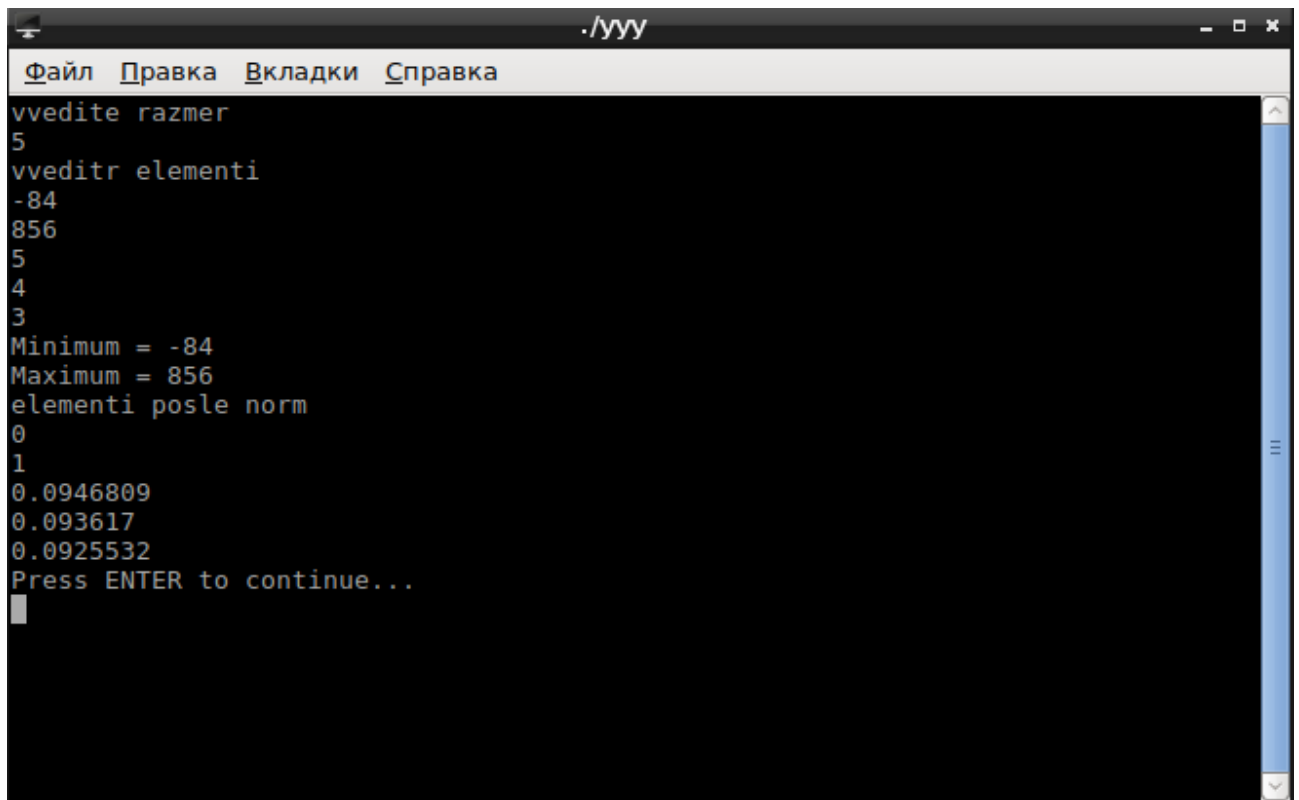


```

#include <iostream>
using namespace std;
void normaliz(double * m, int razmer);
void normaliz(double * m, int razmer)
{
    double max = m[0];
    double min = m[0];
    for (int i = 1; i<razmer; i++) {
        if (m[i]>max)
            max=m[i];
        if (m[i]<min)
            min=m[i];
    }
    cout<<"Minimum = ";
    cout<<min<<endl;
    cout<<"Maximum = ";
    cout<<max<<endl;
    for (int i = 0; i<razmer; i++) {
        m[i]= (m[i]-min)/(max-min);
    }
}
int main()
{
    int a;
    cout<<"vvedite razmer"<<endl;
    cin>>a;
    double * mas = new double[a];
    cout<<"vveditr elementi"<<endl;
    for (int i = 0; i < a; i++) {
        cin>>mas[i];
    }
    normaliz(mas, a);
    cout<<"elementi posle norm"<<endl;
    for (int i = 0; i < a; i++) {
        cout<<mas[i]<<endl;
    }
    return 0;
}

```

Результат работы программы представлен на рисунке 4.



```
./yyy
Файл  Правка  Вкладки  Справка
vvedite razmer
5
vveditr elementi
-84
856
5
4
3
Minimum = -84
Maximum = 856
elementi posle norm
0
1
0.0946809
0.093617
0.0925532
Press ENTER to continue...
█
```

Рисунок 4 - Результат работы программы нормализации

3.3 Алгоритм нормализации представлен в предыдущем пункте, отличие в том, что элементы после проделанной операции вносятся в новый массив.

Код программы:

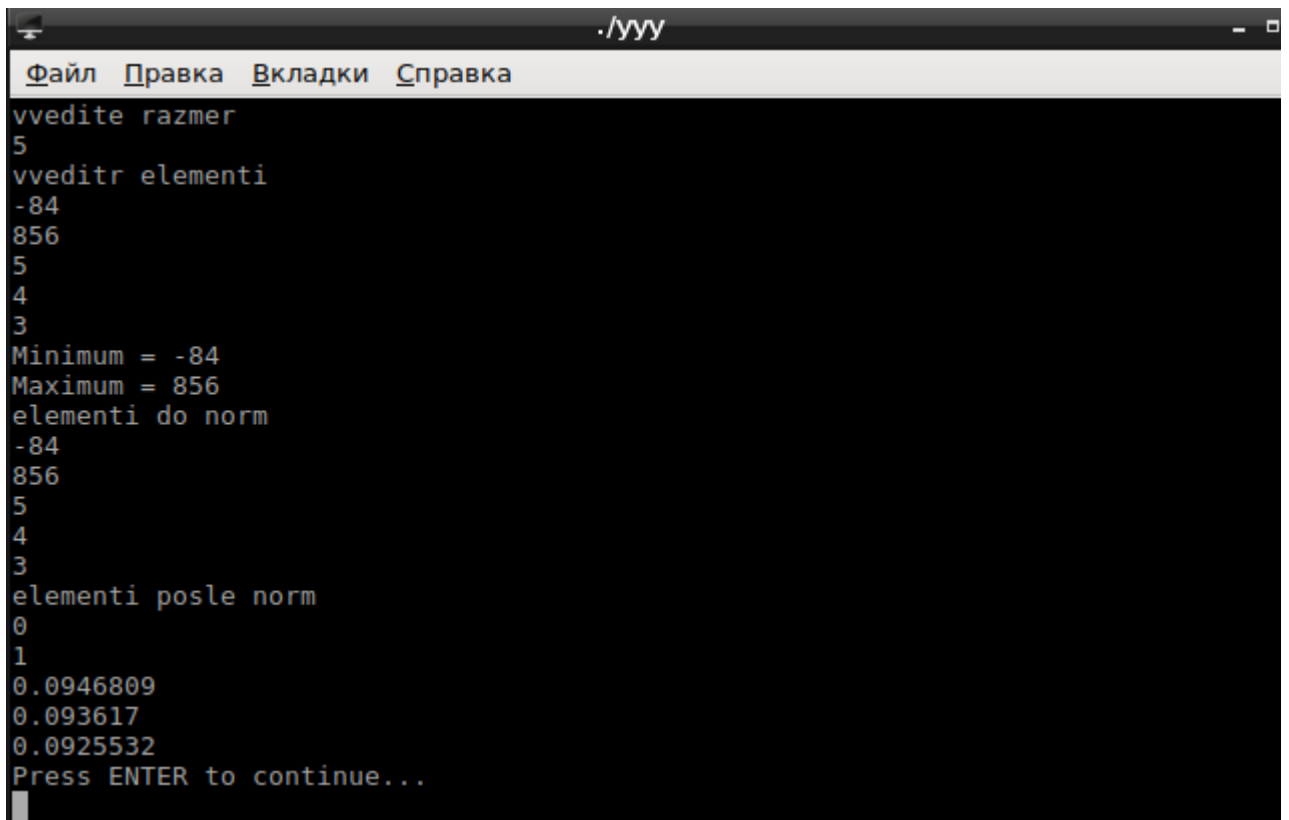
```
#include <iostream>
using namespace std;
double * normaliz(double * m, int razmer);
double * normaliz(double * m, int razmer)
{
    double max = m[0];
    double min = m[0];
    double * p = new double [razmer];
    for (int i = 1; i<razmer; i++) {
        if (m[i]>max)
            max=m[i];
        if (m[i]<min)
            min=m[i];
    }
    cout<<"Minimum = ";
    cout<<min<<endl;
    cout<<"Maximum = ";
```

```

        cout<<max<<endl;
        for (int i = 0; i<razmer; i++) {
            p[i]= (m[i]-min)/(max-min);
        }
        return p;
    }
int main()
{
    int a;
    cout<<"vvedite razmer"<<endl;
    cin>>a;
    double * mas = new double[a];
    cout<<"vveditr elementi"<<endl;
    for (int i = 0; i < a; i++) {
        cin>>mas[i];
    }
    double * sos = normaliz(mas, a);
    cout<<"elementi do norm"<<endl;
    for (int i = 0; i < a; i++) {
        cout<<mas[i]<<endl;
    }
    cout<<"elementi posle norm"<<endl;
    for (int i = 0; i < a; i++) {
        cout<<sos[i]<<endl;
    }
    return 0;
}

```

Результаты работы программы представлен на рисунке 5, с теми же данными, что и в прошлом тестировании.

A terminal window with a dark background and light-colored text. The title bar shows a file icon and the text ".\ууу". The menu bar contains "Файл", "Правка", "Вкладки", and "Справка". The output text is as follows:

```
vvedite razmer
5
vveditr elementi
-84
856
5
4
3
Minimum = -84
Maximum = 856
elementi do norm
-84
856
5
4
3
elementi posle norm
0
1
0.0946809
0.093617
0.0925532
Press ENTER to continue...
```

Рисунок 5 — Результаты работы при копировании в новый массив

3.4 Была выполнена проверка программ написанных в результате выполнения заданий 2.2 и 2.3 на корректность работы с памятью с помощью программы MemCheck. Ошибки не были обнаружены, как показано на рисунке 6.

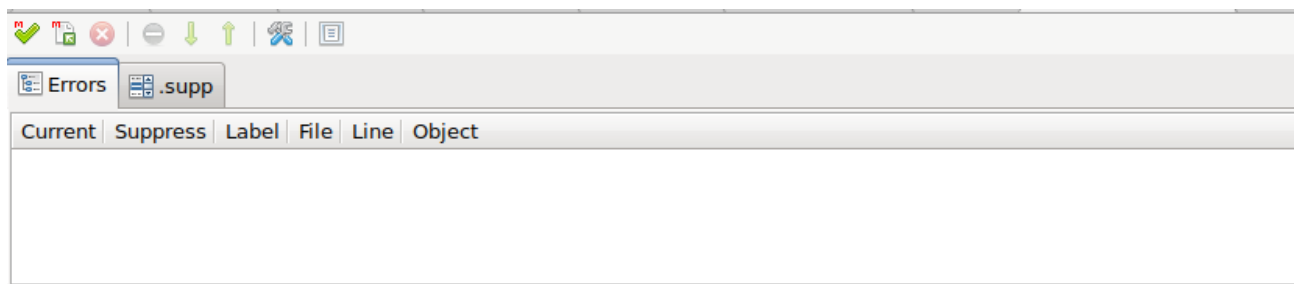


Рисунок 6 — Отсутствие ошибок памяти

#### 4 Вывод:

В процессе выполнения лабораторной работы были изучены массивы и указатели в программах на языке Си++ и освоена их реализация. Были получены практические навыки по созданию и использованию функций, массивов и указателей.

