

Министерство высшего 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:

Проверил: к.т.н., доцент Лупанов М.Ю.

Пенза, 2019

1 Цель работы

1.1 Освоить работу с массивами и указателями в программах на языке C++
+. Освоить создание и использование функций в программах на языке C++.

2 Задания к практической работе

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 Результат выполнения работы

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



Рисунок 1 — Ошибки в программе из приложения Г.

Ошибка «Invalid write of size 1» возникает из-за того, что в строке 26 мы выходим за границу массива. Для исправления ошибки необходимо уменьшить индекс элемента массива на 1. Для исправления ошибки «Invalid read of size 1»

необходимо изменить порядок строк в программе и в строке 29 уменьшить индекс элемента массива на 1. Строчки 28 и 30 необходимо поменять местами, так как должна сначала прочитать значения, находящиеся в памяти, выделенной под массив, вывести их на экран, а затем очистить эту память. Вторая ошибка возникает из-за того, что память, выделяющаяся под массив pT не освобождается. Для того, чтобы исправить эту ошибку необходимо очистить эту память. Код исправленной программы:

```
#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 105
#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 и реализованы программы нормировки массива чисел с плавающей точкой. Параметрами функции является указатель на массив и размер массива. В функции инициализируется переменная, в которой записывается начальное максимальное значение равное 0. Выполняется проверка первого элемента массива, если он больше или равен нулю, то запускается цикл, который работает до тех пор, пока не сравнит все элементы массива с максимальным значением, если элемент больше максимального, то он становится новым максимальным значением. Если первый элемент массива меньше 0, то он становится новым максимальным значением, запускается цикл в котором все элементы массива сравниваются с максимальным, если элемент больше максимального, то он становится новым максимальным значением. Переменной, которая содержит минимальное значение элемента, присваивается значение максимального элемента. Запускается цикл, в котором сравниваются все значения с минимальным. Если значение меньше минимального, то оно присваивается переменной, содержащей минимальное значение. После запускается новый цикл, в котором происходит нормировка. Была разработана программа для проверки корректности работы функции. Алгоритм работы программы представлен на рисунках 2 и 3.

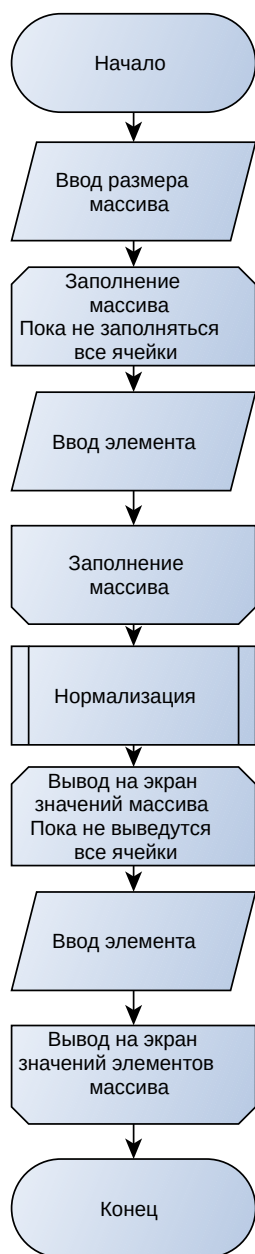


Рисунок 2 — Алгоритм программы проверки корректности работы функции нормализации.

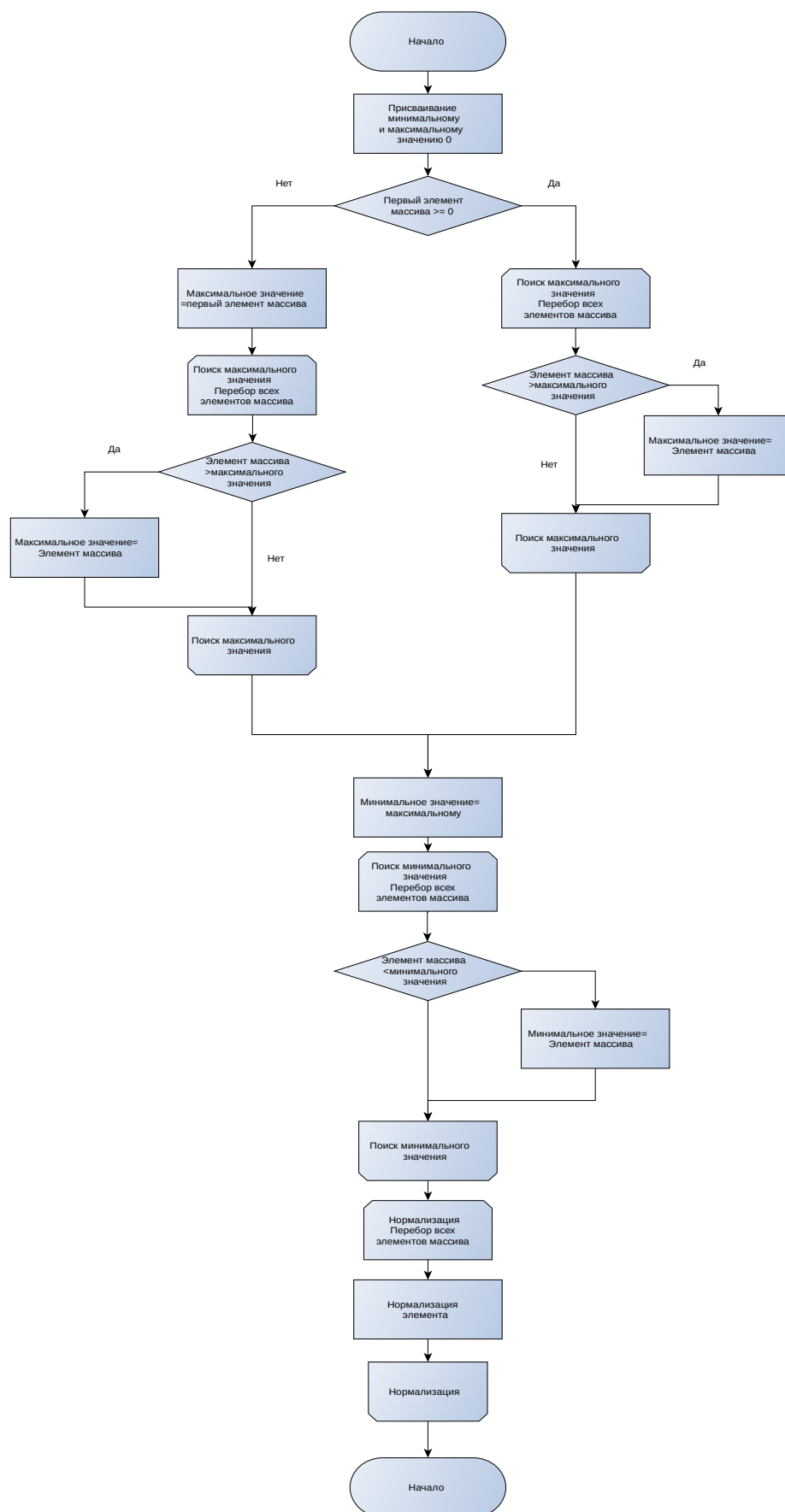


Рисунок 3 — Алгоритм функции нормализации значений.

Код программы:

```
#include <iostream>
using namespace std;
void normmass(double *mas, int raz);
void normmass(double *mas, int raz)
{
    double max=0, min=0;
    if(mas[0]>=0) {
        for (int i = 0; i <= raz-1; i++) {
            if (mas[i]>=max)
                max=mas[i];
        }
    } else {
        max=mas[0];
        for (int i = 0; i <= raz-1; i++) {
            if (mas[i]>=max)
                max=mas[i];
        }
    }
    min=max;
    for (int i = 0; i <= raz-1; i++) {
        if(mas[i]<min)
            min=mas[i];
    }

    cout<<"Максимальное значение массива
"<<max<<endl;
    cout<<"Минимальное значение массива
"<<min<<endl<<endl;
    for (int i = 0; i <= raz-1; i++) {
        mas[i]= (mas[i]-min)/(max-min);
    }

}

int main(int argc, char **argv)
{
    int a;
    cout<<"Введите размер массива"<<endl;
    cin >>a;
    cout<<endl;
    double * sas = new double[a];
    for (int i = 0; i <= a-1; i++) {
        cout<<"Введите элемент массива"<<endl;
```



```

        cin>>sas[i];
        cout<<endl;
    }
    normmass(sas, a);
    for (int i = 0; i <= a-1; i++){
        cout<<sas[i]<<endl<<endl;
    }
    delete[] sas;
    return 0;
}

```

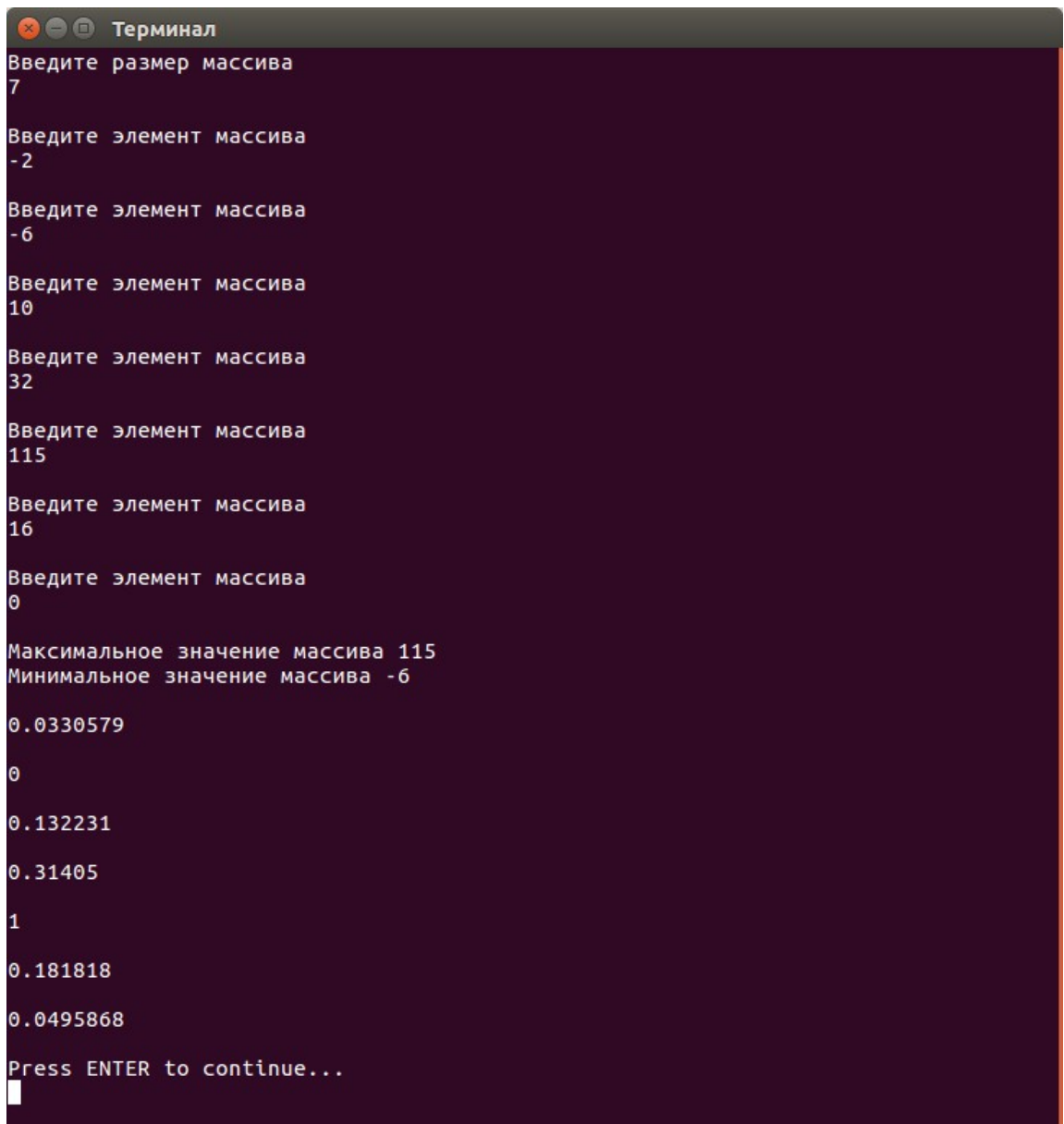
Проверка корректности работы программы проводилась для следующих наборов данных: размер массива 5, элементы: 3, 4, 5, 6, -1; размер массива 7, элементы: -2, -6, 10, 32, 115, 16, 0. Результаты работы программы для этих наборов данных представлены на рисунках 4 и 5.

```

Терминал
Введите размер массива
5
Введите элемент массива
3
Введите элемент массива
4
Введите элемент массива
5
Введите элемент массива
6
Введите элемент массива
-1
Максимальное значение массива 6
Минимальное значение массива -1
0.571429
0.714286
0.857143
1
0
Press ENTER to continue...

```

Рисунок 4 — Результат работы программы для первого набора данных.



```
Терминал
Введите размер массива
7
Введите элемент массива
-2
Введите элемент массива
-6
Введите элемент массива
10
Введите элемент массива
32
Введите элемент массива
115
Введите элемент массива
16
Введите элемент массива
0
Максимальное значение массива 115
Минимальное значение массива -6
0.0330579
0
0.132231
0.31405
1
0.181818
0.0495868
Press ENTER to continue...
█
```

Рисунок 5 - Результат работы программы для второго набора данных.

3.3 Алгоритм работы функции нормировки значений описан в пункте 3.2, с отличием в том, что нормированные значения вносятся в новый массив.

Код программы:

```
#include <stdio.h>
#include <iostream>
using namespace std;
double *normmass(double *mas, int raz);
double *normmass(double *mas, int raz)
{
    double max=0, min=0;
    double*p = new double[raz];
```

```

    if(mas[0]>=0) {
        for (int i = 0; i <= raz-1; i++) {
            if (mas[i]>=max)
                max=mas[i];
        }
    } else {
        max=mas[0];
        for (int i = 0; i <= raz-1; i++) {
            if (mas[i]>=max)
                max=mas[i];
        }
    }
    min=max;
    for (int i = 0; i <= raz-1; i++) {
        if(mas[i]<min)
            min=mas[i];
    }

    cout<<"Максимальное значение массива
"<<max<<endl;
    cout<<"Минимальное значение массива
"<<min<<endl<<endl;
    for (int i = 0; i <= raz-1; i++) {
        p[i]= (mas[i]-min)/(max-min);
    }
    return p;
}

int main(){
    int a;
    cout<<"Введите размер массива"<<endl;
    cin >>a;
    cout<<endl;
    double * sas = new double[a];
    for (int i = 0; i <= a-1; i++) {
        cout<<"Введите элемент массива"<<endl;
        cin>>sas[i];
        cout<<endl;
    }
    double * nm = normmass(sas, a);
    cout<<"Элементы исходного массива"<<endl;
    for (int i = 0; i <= a-1; i++){
        cout<<sas[i]<<endl<<endl;
    }
    cout<<"Элементы нового массива"<<endl;

```

```

        for (int i=0;i <= a-1; i++){
            cout<<nm[i]<<endl<<endl;
        }
        delete[] nm;
        delete[] sas;
        return 0;
    }

```

Проверка корректности работы программы осуществлялась на тех же наборах данных, что и в пункте 3.2.

```

Терминал  Файл  Правка  Вид  Поиск  Терминал  Справка
Введите размер массива
5
Введите элемент массива
3
Введите элемент массива
4
Введите элемент массива
5
Введите элемент массива
6
Введите элемент массива
-1
Максимальное значение массива 6
Минимальное значение массива -1
Элементы исходного массива
3
4
5
6
-1
Элементы нового массива
0.571429
0.714286
0.857143
1
0
Press ENTER to continue...

```

Рисунок 6 — Результат работы программы с записью нормализованных значений в новый массив для первого набора данных.

```
Терминал Файл Правка Вид Поиск Терминал Справка
Введите размер массива
7
Введите элемент массива
-2
Введите элемент массива
-6
Введите элемент массива
10
Введите элемент массива
32
Введите элемент массива
115
Введите элемент массива
16
Введите элемент массива
0
Максимальное значение массива 115
Минимальное значение массива -6
Элементы исходного массива
-2
-6
10
32
115
16
0
Элементы нового массива
0.0330579
0
0.132231
0.31405
1
0.181818
0.0495868
Press ENTER to continue...
█
```

Рисунок 7 — Результат работы программы с записью нормализованных значений в новый массив для второго набора данных.

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.

4 Вывод

В процессе выполнения лабораторной работы были изучены алгоритмы ветвления и циклы, была освоена реализация ветвления и циклов в языке C++, были получены практические навыки по использованию инструкции ветвления if-else в языке C++, по использованию циклов с постусловием и предусловием при помощи инструкций while и do — while и инструкции for в языке C++.