Московский Технологический Университет

(МИРЭА)

Институт информационных технологий

Кафедра математического обеспечения и стандартизации информационных технологий

Итоговый отчёт

по лабораторному практикуму

по дисциплине «Программирование»

Студент: Прохоров Андрей

Группа: ИВБО-02-15

Преподаватель: к.т.н. доцент Сыромятников В.П.

Москва 2017

**Содержание**

Введение 3

Лабораторная работа №1 5

Лабораторная работа №2 8

Лабораторная работа №3 16

Лабораторная работа №4 25

Лабораторная работа №5 37

Лабораторная работа №6 43

Лабораторная работа №7 60

Эскизы по программированию №1 74

Эскизы по программированию №2 81

Приложение 94

Заключение 96

Список литературы 97

**Введение**

В III семестре меня ждала (и ждет) дисциплина «Программирование», где изучается С++. Я уже был знаком с этим языком ранее. В прошлом семестре я занимался программированием на С++. Поэтому мне было легко в начале, однако повторение должно было позволить мне более глубоко проникнуться как логикой языка в целом, так и приемами в частностях. Да и абсолютно новый материал ждал меня впереди.

Язык C++ — это ключ к современному объектно-ориентированному программированию. Он создан для разработки высокопроизводительного программного обеспечения и чрезвычайно популярен среди программистов. Сегодня быть профессиональным программистом высокого класса означает быть компетентным в C++.

Этот язык не просто популярен. Он обеспечивает концептуальный фундамент, на который опираются другие языки программирования и многие современные средства обработки данных. Не случайно ведь потомками C++ стали такие почитаемые языки, как C# и Java.

Поскольку язык C++ предназначен для профессионального программирования, для изучения он не самый простой; тем не менее, C++ — самый лучший язык для изучения. Освоив C++, можно писать профессиональные высокопроизводительные программы. Кроме того, можно легко изучить такие языки программирования, как C# и Java, поскольку они используют тот же базовый синтаксис и те же принципы разработки.

Об актуальности С++ сказано много. Этот классический язык программирования применяется крайне широко: он позволяет создавать высокопроизводительные приложения с «нативным» управлением оперативной памятью, применяется для написания низкоуровневых фреймворков и драйверов. Поэтому вопрос изучения С++ для меня не стоял. Учить и самосовершенствоваться.

Однако нужно было освежить былые знания. Ранее я прошел весьма краткий курс С++, который должен был скорее познакомить с программированием в общем, чем с С++ в частности. Мы создавали очень простые линейные программы, с малым количеством циклов. В новом же семестре мы начали от крайне простых и тривиальных программ, а закончили – сложными, массивными алгоритмами, способными работать с динамическими данными.

Не менее важно – системный подход. Каждая последующая программа не упрощалась. Можно сказать, что первая программа обрастала различными конструкциями, формируя последующие. Это гарантировало красоту консольного интерфейса и ориентированность на конечного (гипотетического) пользователя.

Перейдем же к лабораторному практикуму и взглянем на все ступени в моем пути.

## Лабораторная работа № 1

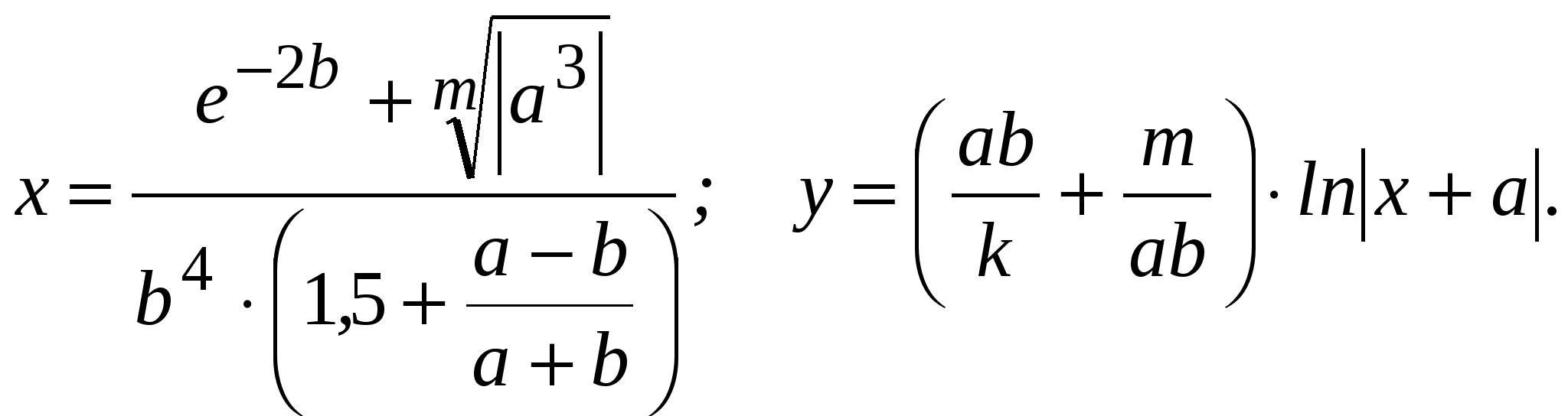
## ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

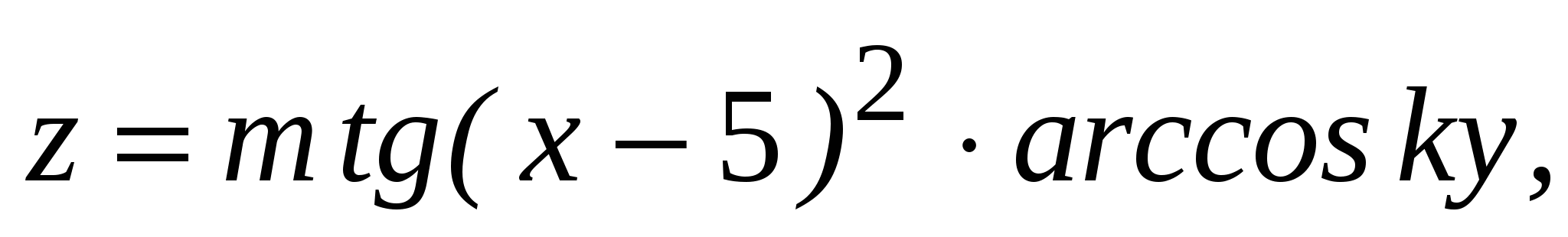
**Цель работы**

Получить навыки программирования линейных алгоритмов на языке C++.

**Постановка задачи**

Составить программу на языке С++ для расчета значений заданной функции z.





при инициализированных a=-1.1, b=-3, k=0.1, m=2.

**Описание алгоритма**

Начало

Инициализация a, b, k, m

Вычисление x,y,z

Вывод x,y,z

Конец

Рис. 1. Схема алгоритма программы main().

**Текст исходного кода программы с комментариями**

#include <iostream> //заголовочный файл с классами, функциями и переменными для организации ввода-вывода

#include <math.h> //заголовочный файл стандартной библиотеки, для выполнения простых математических операций

#include <cmath>

using namespace std; //пространство имен - используется для того, чтобы ограничить видимость переменных, функций и т.п.

int main()

{

//Установка языка

setlocale(LC\_ALL, "Russian"); // русификатор

//Обьявление входных данных.

int b = -3, m = 2;

float x, y, z, a = -1.1, k = 0.1;

x = (exp(-2 \* b) + pow((abs(pow(a, 3))), (1 / m)))/((pow(b, 4))\*(1.5+((a-b)/(a+b))));

y = (((a\*b)/k) + (m/(a\*b)))\*(log(abs(x + a)));

z = m\*(tan(pow((x - 5), 2)))\*(acos(k\*y));

//Вывод значений функций

cout << "Вычисление математических функций." << endl;

cout << "Результаты вычислений: " << endl;

cout << "x= " << x << endl;

cout << "y= " << y << endl;

cout << "z=: " << z << endl;

system("Pause"); ");// вызывает системную команду pause, которая не дает программе завершится сразу после вывода

return 0; /\*Программа после выполнения может возвратить код ошибки системе, которая запустила эту программу,

будь то командная строка или другая программа (0 - нет ошибки)\*/

}

**Контрольный прогон программы**

Тест №1 – вывод программой корректных значений функций x,y,z.

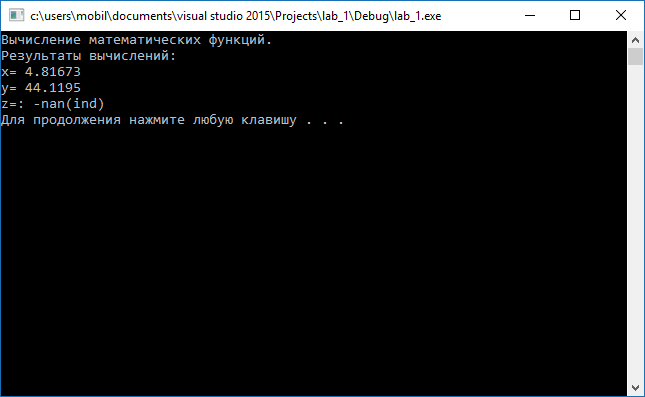


Рис 2. Результат теста №1

**Вывод**

Эта программа помогает освоить запись основных стандартных математических функций, правила записи выражений на языке С++, назначение и формат оператора присваивания, операторы ввода/вывода.

Эта программа позволяет освоить запись натурального логарифма (log), возведение числа в степень (pow), а именно: возведение в степень как больше, так и меньше 1. Узнать о записи модуля числа (abs) и arccos (acos).

Стало ясно, что проверка исходных данных – неотъемлемый шаг к созданию конечного продукта.

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: <http://cppstudio.com/post/1141/>

3. Интернет-ресурс: http://cppstudio.com/post/1167/

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», МТУ МИРЭА, лекции, 2016/2017.

## Лабораторная работа № 2

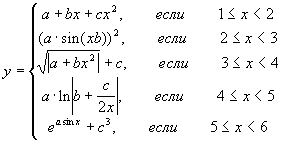
## ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

**Цель работы**

Получить навыки программирования разветвляющихся алгоритмов на языке C++.

**Постановка задачи**

Составить программу для вычисления функции:

****

a, b, с – определить (инициировать) в программе,

x – ввести с клавиатуры,

y – вывести на экран.

**Описание программы**

Программа получает значение x с клавиатуры, сразу же проводя проверку на принадлежность лучу от 1 до 6. В случае, если полученное значение не принадлежит ему ил является иным типом данных – предлагается повторный ввод. Проверка реализована при помощи стандартной функции cin.ignor().

Функция cin.rdbuf() обращается к буферу чтения, а in\_avail отображает количество символом этого буфера. сin.clear() и удаляет их. Так достигается проверка ввода.

Далее следует проверка на принадлежность диапазонам:

1. 1<=x<2
2. 2<=x<3
3. 3<=x<4
4. 4<=x<5
5. 5<=x<6

В соответствии с диапазоном значений выбирается соответствующая функция вычисления y. Значение y выводится на экран.

**Описание алгоритма**

Рис. 1. Схема алгоритма программы main()

Начало

Инициализация a, b, c;

double x, y;

Ввод x

Cout<<”Введие верное значение”;

x<6 и x>=1?

false

true

x<3 и x>=2?

x<2 и x>=1?

true

y = a + b\*x +y = a + b\*x + c\*x\*x; c\*x\*x;

false

y = pow((a\*sin(b\*x)), 2);

x<5 и x>=4?

x<4 и x>=3?

true

false

y = sqrt(abs(a + b\*pow(x, 2))) + c;

true

false

true

y = sqrt(abs(a + b\*pow(x, 2))) + c;

11/2

11/1

false

10/2

10/1

y = exp(a\*sin(x)) + pow(c, 3);

Вывод у

Конец

Рис. 1. Схема алгоритма программы main()

**Текст исходного кода программы с комментариями**

#include <iostream>

#include <cmath>

#include <math.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian"); //русификация

int a = 2, b = 3, c = 5, answer; //инициализация a, b, c

double y, x; //инициализация x,y

cout << "Введите значение x от 1, но меньше 6: ";

cin >> x; //ввод x

if (x<1 || x>=6 || cin.fail()) //проверка на диапазон значений и тип переменной

cout << "Неверно. Введите число от 1, но меньше 6." << endl;

cin.clear(); //очистка потока вводных данных, предотвращающая циклический вывод предупреждения

cin.ignore(cin.rdbuf()->in\_avail()); //игнорирование (ignore) колличества символов (avail) буфера (rdbuf)

while (x<1 || x >= 6) {

cout << "Введите число: ";

cin >> x;

if (x<1 || x >= 6 || cin.fail())

cout << "Неверно. Введите число от 1, но меньше 6." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

//различные варианты функции при соответствующих значениях х

if (x >= 1 && x < 2)

y = a + b\*x + c\*x\*x;

if (x >= 2 && x < 3)

y = pow((a\*sin(b\*x)), 2);

if (x >= 3 && x < 4)

y = sqrt(abs(a + b\*pow(x, 2))) + c;

if (x >= 4 && x < 5)

y = a\*log(abs(b + (c / (2 \* x))));

if (x >= 5 && x < 6)

y = exp(a\*sin(x)) + pow(c, 3);

cout << "Значение y: " << y << endl; //вывод y

cout << endl;

system("Pause");

return 0;

}

**Контрольные прогоны программы**

Тест №1 – ввод значений, не удовлетворяющих допустимому диапазону значений. Последующий ввод корректного значений x (1).

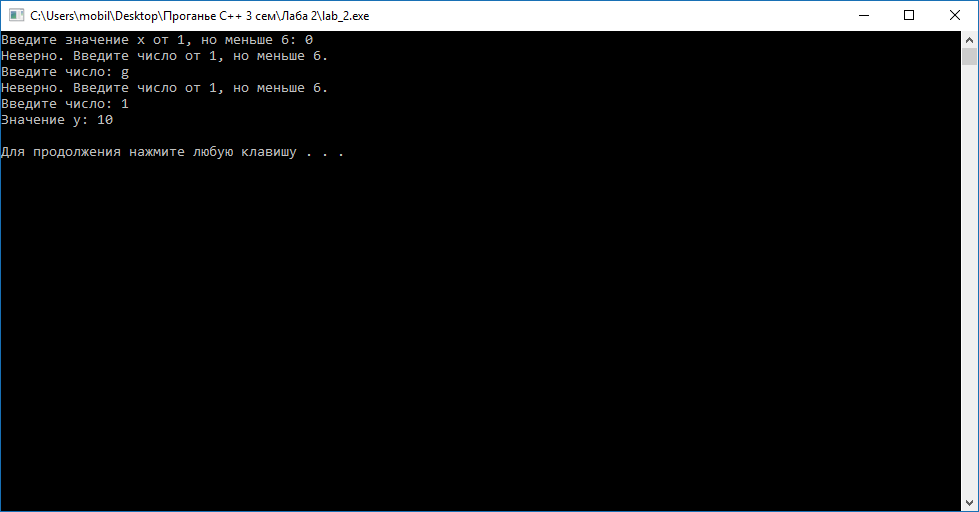


Рис. 2. Результат теста №1

Тест №2 – ввод значения x (2).

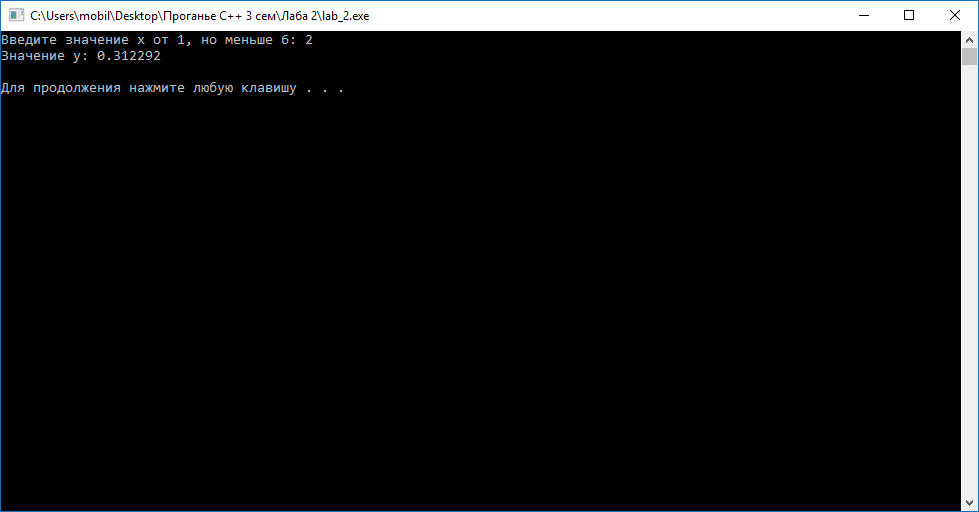


Рис. 3. Результат теста №2

Тест №3 – ввод значения x (3).

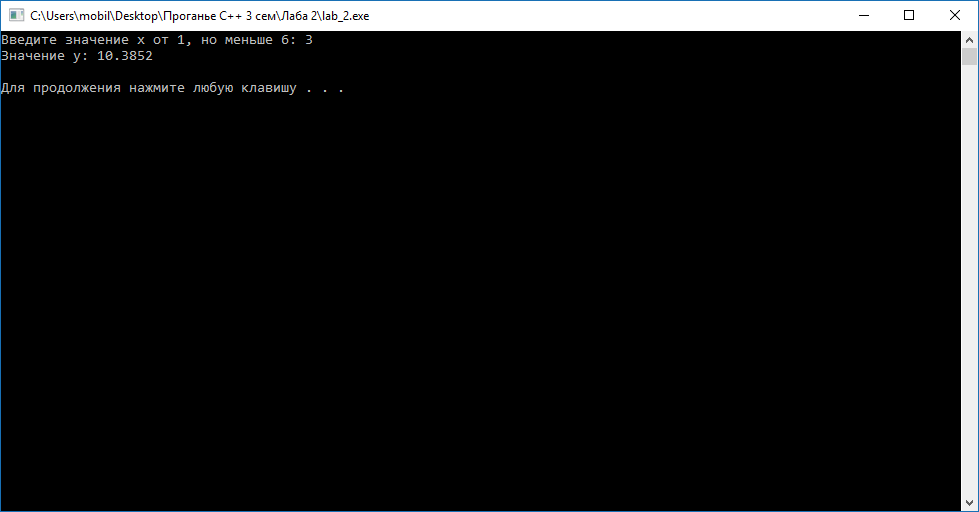


Рис. 4. Результат теста №3

Тест №4 – ввод значения x (4).

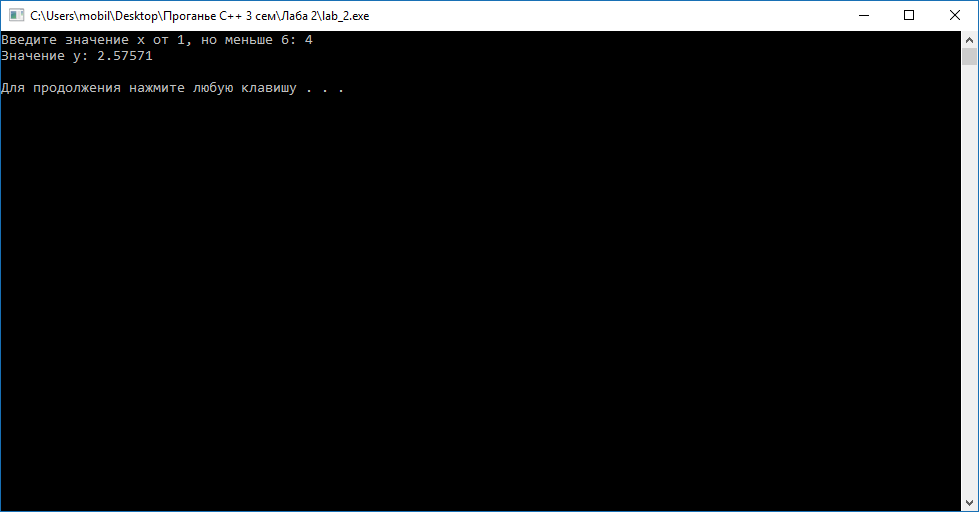


Рис. 5. Результат теста №4

Тест №5 – ввод значения x (5).

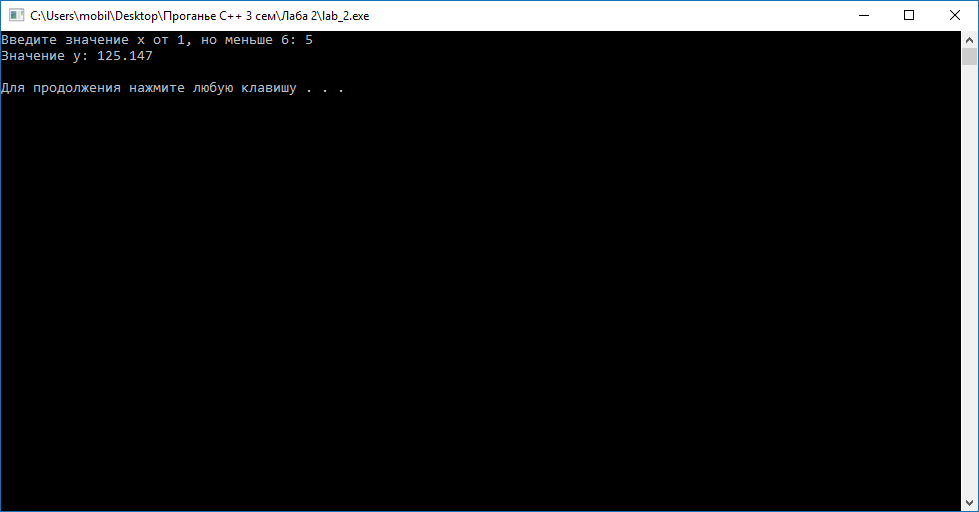


Рис. 6. Результат теста №5

**Вывод**

Задание дает представление о написании ветвящихся алгоритмов на языке С++. Так же программа позволила мне не только освоить проверку введённых с клавиатуры значений, но и создать конструкцию, позволяющую проверять как диапазон введенных значений, так и их тип (cin.fail()). Эта функция функционирует вместе с cin.rdbuf() и cin.clear().

Она будет использоваться мной в будущих программах. Так же была изучена функция извлечения корня (sqrt()).

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: http://www.programmersforum.ru/showthread.php?t=98038

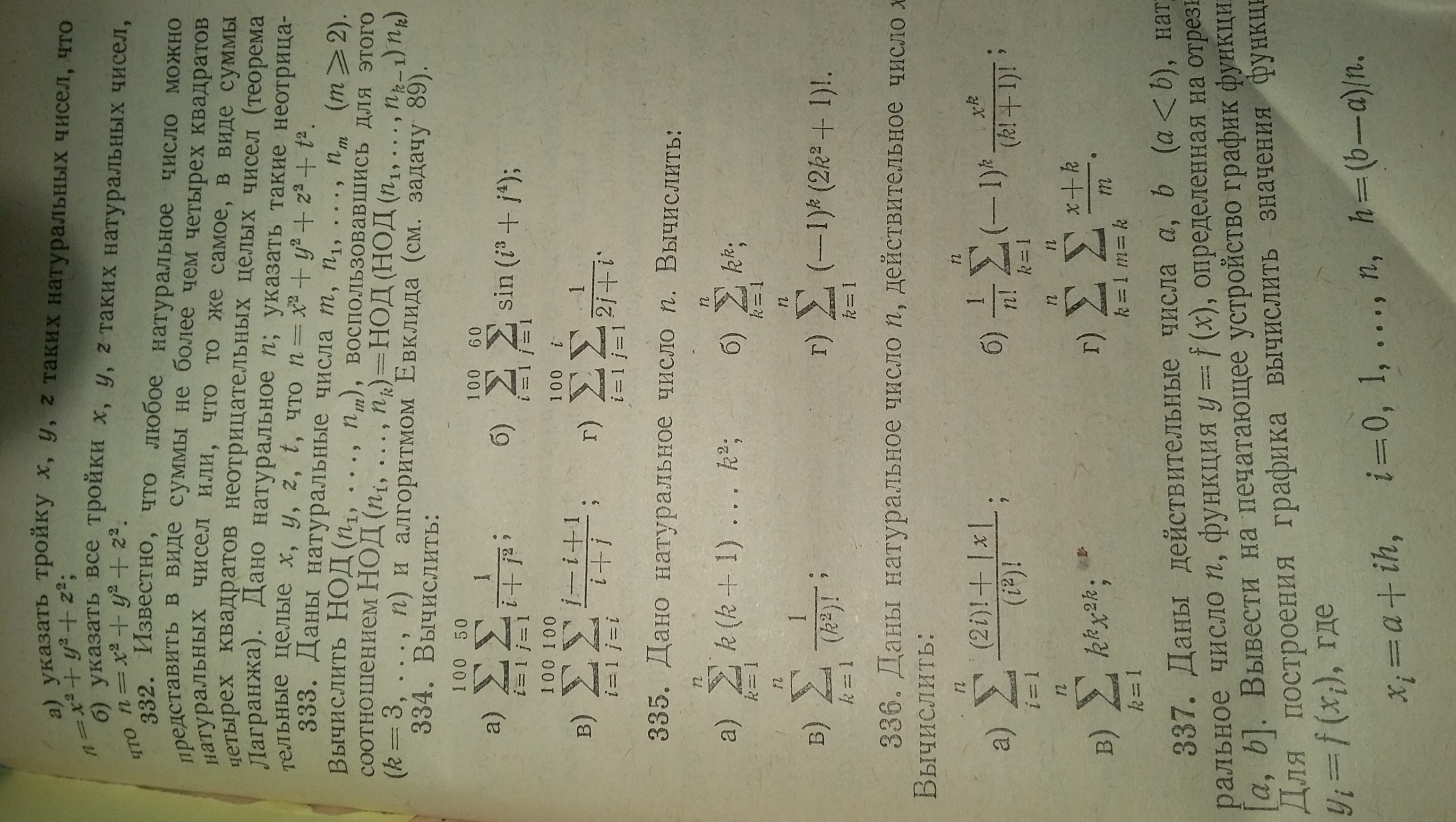
3. Интернет-ресурс: https://cppstudy.wordpress.com/2009/03/27/cin-get-and-co/

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», МТУ МИРЭА, лекции, 2016/2017.

## Лабораторная работа №3

## ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

**Цель работы**

Получить навыки программирования циклических алгоритмов на языке C++.

**Постановка задачи**

Cоставить программу вычисления суммы S.

Значение ***n*** ввести с клавиатуры.

**Описание алгоритма**

Рис. 1. Схема алгоритма программы main()

Ввод n

n>=1?

Начало

Инициализация n, k, i, answer, sumall=0;

cout<<”Введите корректное n”;

false

true

fact(n)

17/1

proizv(n)

Рис. 1. Схема алгоритма программы main()

16/1

false

true

false

false

true

i <= n

i=1

i++

Показать слагаемые?

Вывод sumall

true

i++

sumall += proizv(i)

i <= n

i=1

cout << i << " слагаемое=" << proizv(i)

Конец

Начало

Инициализация N;

true

N<0

return 0;

false

true

N=0

return 1;

false

return N\*fact(N-1);

Конец

Рис. 2. Схема алгоритма функции fact()

Начало

Инициализация n, k, sum, sum1, sumpart;

k = pow(n, 2);

sum = fact(k);

sumpart = fact(n - 1);

true

sum=1

sum1 = 1;

false

sum1 = sum/sumpart;

return sum1;

Конец

Рис. 3. Схема алгоритма функции proizv()

**Описание программы**

Программа получает с клавиатуры номер слагаемого, до которого будет проходить суммирование, проводя проверку на значение и тип данных при помощи стандартной функции cin.ignor(). В цикле for рекурсивная подпрограмма нахождения факториала fact() передает во подпрограмму proizv() значение факториала счетчика. В proizv() факториал квадрата счетчика делится на факториал счетчика. В итоге находится значение текущего элемента. В этом же цикле for проходит суммирование элементов.

Далее на экран выводится сумма ряда и пользователю предлагается просмотреть слагаемые. Отказ ведет к выходу из программы.

**Текст исходного кода программы с комментариями**

#include <iostream>

#include <math.h>

#include <cmath>

#include <cstdlib>

using namespace std;

long double fact(int N) //нахождение факториала числа N

{

if (N < 0)

return 0;

if (N == 0)

return 1;

else

return N \* fact(N - 1);

}

long double proizv(int n) { //нахождение произведения n\*(n+1)\*...\*(n^2)

long double sum, sum1, sumpart;

int k;

k = pow(n, 2); //верхняя граница произведения

sum = fact(k);

sumpart = fact(n - 1);

if (sum == 1)

sum1 = 1;

else

sum1 = sum / sumpart;

return sum1;

}

int main() {

setlocale(LC\_ALL, "Russian");

int n, k, i, answer;

long double sum1, sumall = 0;

cout << "Введите n: ";

cin >> n;

if (n<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (n<1) {

cout << "Введите натуральное число: ";

cin >> n;

if (n<1 || cin.fail()) cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

cout << "Максимальный элемент=" << proizv(n) << endl;

for (int i = 1; i <= n; i++) //цикл нахождения суммы элементов

{

sumall += proizv(i); //сумма элементов

}

cout << "Сумма: " << sumall << endl;

cout << endl;

cout << "Желаете просмотреть слагаемые? Нажмите 1 для их просмотра и 0 для завершения программы." << endl;

cin >> answer;

if (answer<0 || answer>1 || cin.fail())

cout << "Неверно. Нажмите 1 для их просмотра и 0 для завершения программы." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (answer<0 || answer>1) {

cout << "Сделайте выбор: ";

cin >> answer;

if (answer < 0 || answer > 1 || cin.fail()) cout << "Неверно. Нажмите 1 для их просмотра и 0 для завершения программы." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

switch (answer) {

case 1:

for (int i = 1; i <= n; i++) //цикл вывода слагаемых

{

cout << i << " слагаемое=" << proizv(i)<<endl; //вывод слагаемых

}

break;

case 0:

exit(0);

break;

}

system("Pause");

return 0;

}

**Контрольные прогоны программы**

Тест №1 – ввод некорректных, и последующий ввод корректных значений (3) с отображением слагаемых.

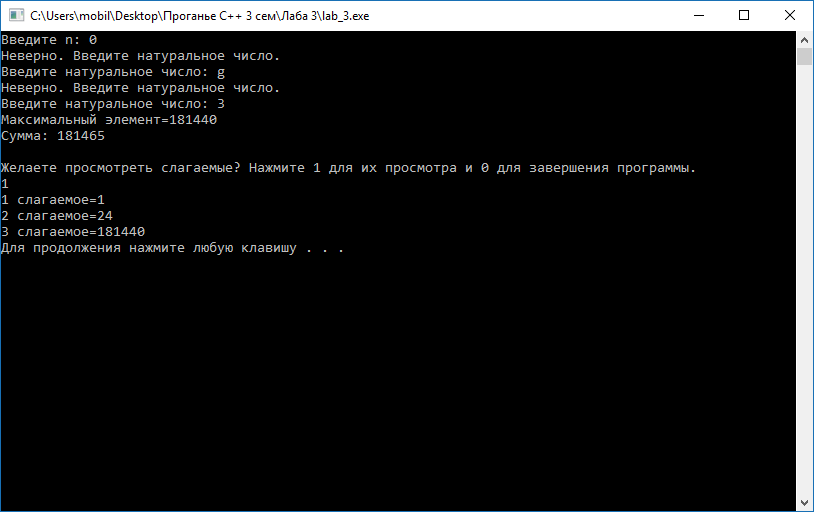


Рис. 4. Результат теста №1

Тест №2 – ввод корректных значений (7) с отображением слагаемых.

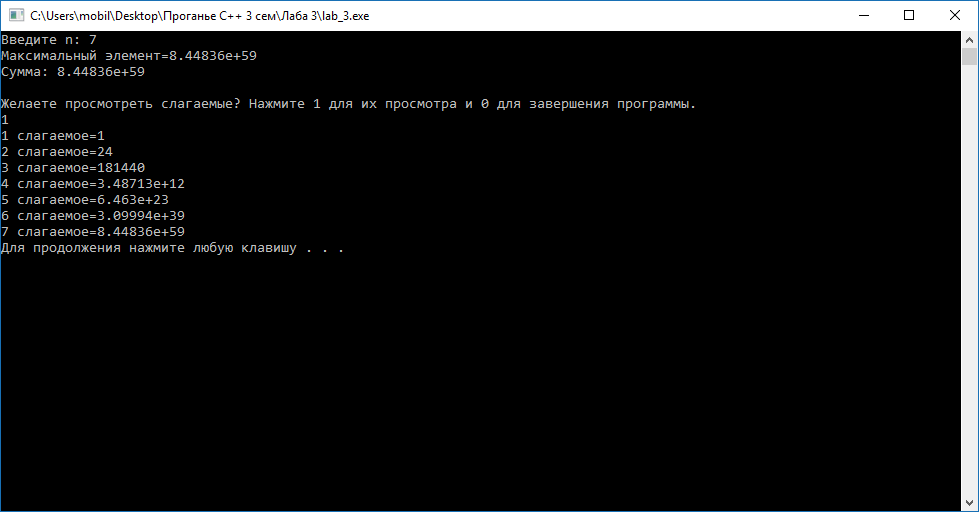


Рис. 5. Результат теста №2

Тест №3 – ввод корректных значений (3) и проверка на фильтрацию ввода для отображения слагаемых.

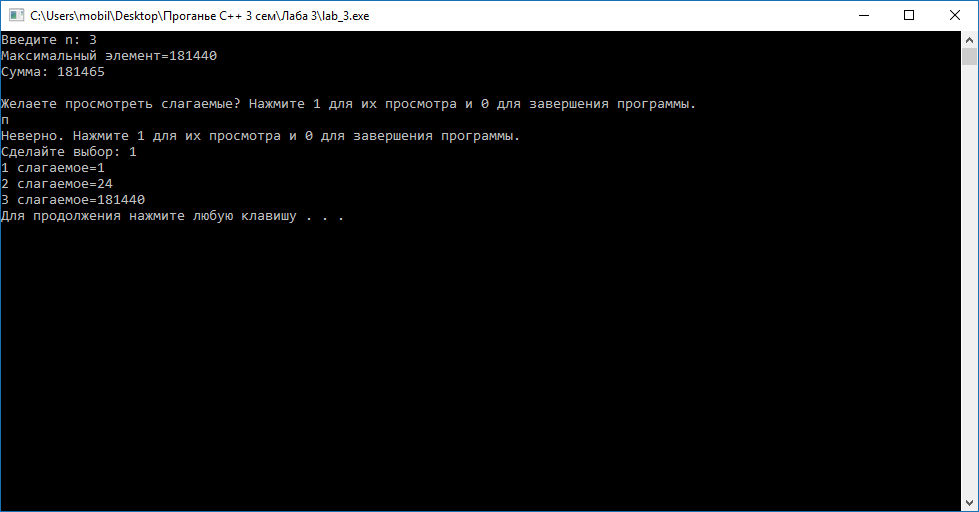


Рис. 6. Результат теста №3

**Вывод**

Данная программа позволяет освоить навыки написания циклических алгоритмов на языке С++. Также я получил опыт в написании функций (fact, proizv) и их использовании в главной процедуре (int(main)). В данной работе я использовал оператор множественного выбора switch.

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: http://cppstudio.com/post/348/

3. Интернет-ресурс: http://cppstudio.com/post/396/

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

## 

## Лабораторная работа №4

**ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ**

**Цель работы**

Приобретение навыков разработки алгоритмов обработки двумерного массива. Закрепление понятий базовых структур цикл, разветвление.

**Постановка задачи**

Составить программу формирования двумерного массива А [n\*n]. Значение n – задается вводом с клавиатуры.

Упорядочить элементы каждой строки по возрастанию. Найти среднее арифметическое элементов каждого столбца матрицы.

**Описание алгоритма**

Рис. 1. Схема алгоритма программы main()

Начало

Инициализация \*mas, sizemas, answer, sum;

Ввод sizemas;

cout<<”Введите корректное sizemas”;

false

sizemas<1

true

int i = 0;

26/1

25/1

false

i <sizemas;

true

mas[i] = new int[sizemas];

cin >> answer;

cout<<"Желаете заполнить массив вручную - 1 или случайными значениями - 2?";

i++;

cout<<”Введите корректное sizemas”;

false

answer <1 || answer > 2

false

true

answer =1

true

27/2

27/3

Рис. 1. Схема алгоритма программы main()

26/3

26/2

j <sizemas;

mas[i][j] = rand() % 100;

j++;

i++;

i=0, j=0;

i++;

j++;

j <sizemas;

i <sizemas;

i=0, j=0;

i <sizemas;

false

false

false

false

true

true

true

true

cin >> mas[i][j];

cout << "Массив с упорядоченными строками:" << endl;

cout << "Исходный массив:" << endl;

i=0, j=sizemas-1, k=0;

28/4

Рис. 1. Схема алгоритма программы main()

27/4

k <sizemas;

false

true

j>i;

i <sizemas;

false

true

false

true

k++;

i++;

j--;

int tmp = mas[k][j]; mas[k][j] = mas[k][j - 1]; mas[k][j - 1] = tmp;

29/5

Рис. 1. Схема алгоритма программы main()

Рис. 1. Схема алгоритма программы main()

28/5

i=0, j=0;

j++;

i++;

false

j <sizemas;

i <sizemas;

true

false

true

cout << mas[i][j];

cout << "Средние арифметические столбцов массива:" << endl;

i=0, j=0;

false

i <sizemas;

30/8

30/7

30/6

30/8

30/7

29/6

true

i++;

sum = 0;

j <sizemas;

sum += mas[j][i];

j++;

false

true

cout << "Среднее " << i + 1 << " = " << sum/sizemas << endl;

Конец

Рис. 1. Схема алгоритма программы main()

**Описание программы**

В программе инициализируется двумерный массив и его размер. Пользователю предлагается ввести его размер с клавиатуры. При вводе проводится проверка на размер (строго >1) и тип данных. В случае ошибки предлагается ввести размер повторно. Проверка реализована средствами стандартной функции cin.fail().

Далее массив можно заполнить двумя способами: вручную или датчиком псевдослучайных чисел rand(). Пользователь выбирает тот метод, который желает.

На экран выводится введенный массив. Далее в тройном цикле for происходит упорядочивание строк массива по возрастанию. Упорядоченный массив выводится на экран. След за этим двойным циклом for вычисляются средние арифметические значения каждого столбца массива и выводятся на экран.

**Текст исходного кода программы с комментариями**

#include <iostream>

#include <cstdlib>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int \*\*mas, answer, sizemas;

double sum;

cout << "Введите размер массива: ";

cin >> sizemas;

if (sizemas <1 || cin.fail())

cout << "Неверно. Введите число от 1." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (sizemas<1) {

cout << "Размер массива:";

cin >> sizemas;

if (sizemas < 1 || cin.fail()) cout << "Неверно. Введите число от 1." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

mas = new int\*[sizemas];

for (int i = 0; i < sizemas; i++) {

mas[i] = new int[sizemas];

}

cout << "Желаете заполнить массив вручную - 1 или случайными значениями - 2?" << endl;

cin >> answer;

if (answer <1 || answer > 2 || cin.fail())

cout << "Неверно. Введите 1 или 2." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (answer <1 || answer > 2) {

cout << "Ваш выбор: ";

cin >> answer;

if (answer < 1 || answer >2 || cin.fail()) cout << "Неверно. Введите 1 или 2." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

switch (answer) {

case 1: //ручной ввод массива

for (int i = 0; i < sizemas; i++) {

for (int j = 0; j < sizemas; j++) {

cout << "Элемент " << i << " " << j << ": ";

cin >> mas[i][j];

}

cout << endl;

}

break;

case 2: //заполнение случайнми числами

for (int i = 0; i < sizemas; i++) {

for (int j = 0; j < sizemas; j++) {

mas[i][j] = rand() % 100;

}

cout << endl;

}

break;

}

cout << "Введенный массив:" << endl;

for (int i = 0; i < sizemas; i++) {

for (int j = 0; j < sizemas; j++) {

cout << mas[i][j] << "\t";

}

cout << endl;

}

cout << endl;

cout << "Массив с упорядоченными строками:" << endl;

for (int k = 0; k < sizemas; k++) //счетчик цикла

{

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

{

for (int j = sizemas - 1; j > i; j--)

if (mas[k][j] < mas[k][j - 1])

{

int tmp = mas[k][j];

mas[k][j] = mas[k][j - 1];

mas[k][j - 1] = tmp;

}

}

}

for (int i = 0; i < sizemas; i++) {

for (int j = 0; j < sizemas; j++) {

cout << mas[i][j] << "\t";

}

cout << endl;

}

cout << endl;

cout << "Средние арифметические столбцов массива:" << endl;

for (int i = 0; i<sizemas; i++) {

sum = 0;

for (int j = 0; j<sizemas; j++) {

sum += mas[j][i]; //сумма столбца

}

cout << "Среднее " << i + 1 << " = " << sum/sizemas << endl;

}

cout << endl;

system("Pause");

return 0;

}

**Контрольные прогоны программы**

Тест №1 – ввод массива с клавиатуры при значении n=3.

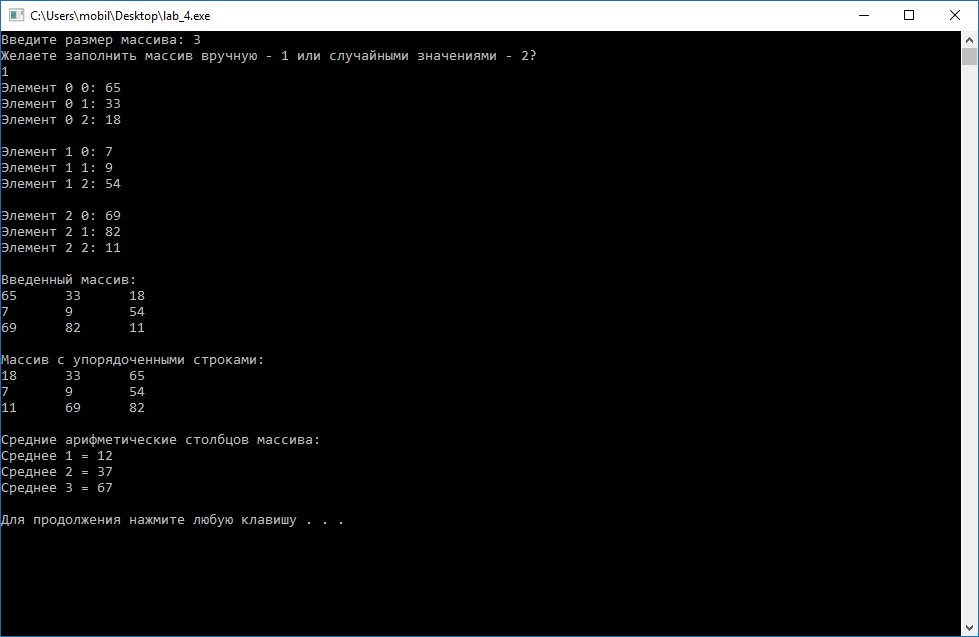


Рис. 2. Результат теста №1

Тест №2 – формирование массива путем генерации псевдослучайных значений при n=6.

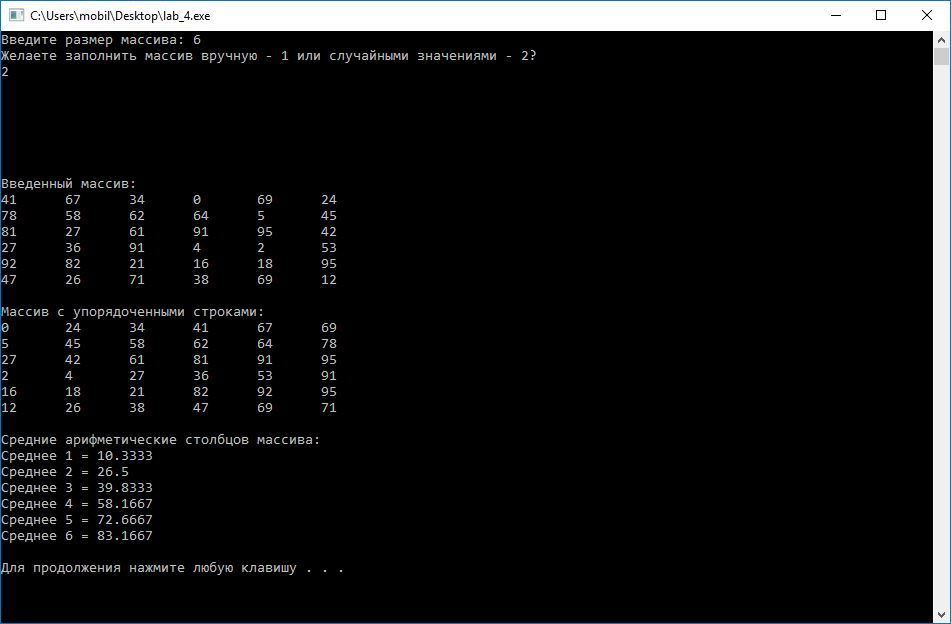


Рис. 3. Результат теста №2

**Выводы**

Данная программа позволила мне освоить работу с двумерными массивами в С++. Это абсолютно новый опыт – до этого я не работал с двумерными динамическими массивами. Также данное задание позволило мне улучшить навыки работы с вложенными циклами (for).

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: https://code-live.ru/post/cpp-array-tutorial-part-2/

3. Интернет-ресурс: http://cppstudio.com/post/9407/

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

## 

## Лабораторная работа №5

## ОБРАБОТКА СТРОКОВЫХ ДАННЫХ

**Цель работы**

Приобретение навыков разработки алгоритмов обработки строковых данных. Изучение специальных функций работы со строками символов из стандартной библиотеки.

**Постановка задачи**

Составить программу на языке С++, которая вводит с клавиатуры строку символов длиной не более 200 символов, находит в ней самое короткое слово и выводит его и позицию его вхождения в строку.

**Описание алгоритма**

Рис. 1. Схема алгоритма программы main()

Начало

Инициализация N, \*string, curlen=0, minlen=200, index=0, i;

i<200

i=0;

Ввод string;

false

true

38/3

38/2

38/1

37/2

i++;

index = i;

false

false

curlen=0;

index = i;

curlen++;

index = i;

minlen = curlen;

index = i;

index = i;

true

curlen<minlen

string[i] == ' ' || string[i] == '\0'

true

37/1

37/3

Вывод minlen;

Вывод index - minlen + 1;

Контрольные прогоны.

Конец

Рис. 1. Схема алгоритма программы main()

**Описание программы**

В программе инициализируется константа N=200, являющаяся максимальной длиной строки. Далее инициализируется сама строка и указатель на нее.

Происходит инициализация и объявление минимального слова в строке.

Пользователю предлагается ввести строку с клавиатуры. При помощи функции библиотеки <string> - cin.getline происходит считывание массива символов. Далее в цикле for находится минимальное слово в строке.

На экран выводится минимальное слово и номер его первой буквы.

**Текст исходного кода программы с комментариями**

#include <iostream>

#include <string>

using namespace std;

const int N = 200;

int main()

{

setlocale(LC\_ALL, "Russian");

char \*string = new char[N]; //указатель на строковый тип

int curlen = 0, minlen = 200, index = 0, i; //минимальная длина

cout << "Введите строку: ";

cin.getline(string, 200);

for (i = 0;; i++)

{

if (string[i] == ' ' || string[i] == '\0')

{

if (curlen<minlen) { //поиск минимального

minlen = curlen;

index = i;

}

if (string[i] == '\0')

break;

curlen = 0;

}

else curlen++;

}

cout << "Наименьшее колличество символов в слове:" << minlen << endl;

cout << "Номер первой буквы наименьшего слова:" << index - minlen + 1 << endl;

//system("Pause");

return 0;

}

**Контрольные прогоны программы**

Тест №1 – ввод строки на русском языке.

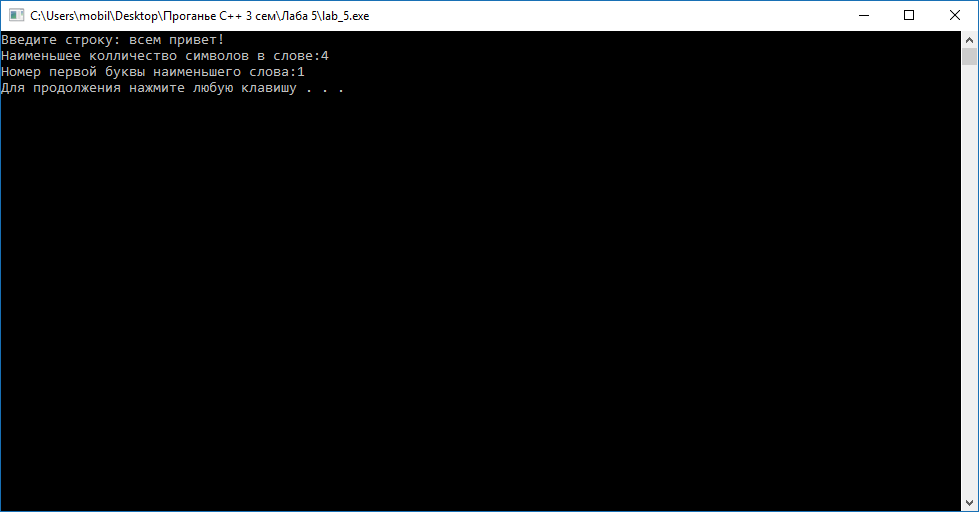


Рис. 2. Результат теста №1

Тест №2 – ввод строки на английском языке.

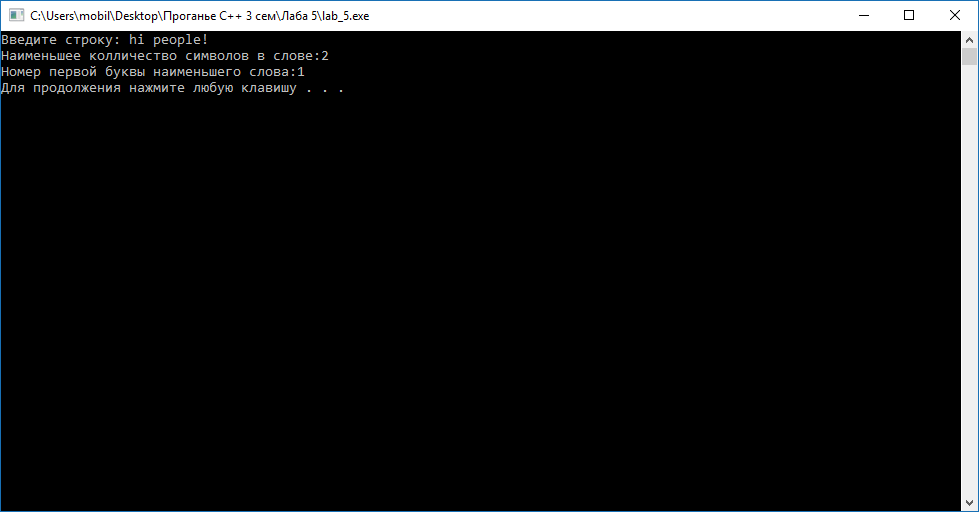


Рис. 3. Результат теста №2

**Выводы**

Данная программа позволила мне освоить работу со строковым и символьным типом данных в С++. В работе я использовал строковый тип данных string, который появился впервые в С++ и подключается библиотекой <string>. Так же я использовал указатель для создания строки динамической длины (\*string). Научился использовать команду new для выделения памяти и cin.getline() для получения потока символов.

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: http://cppstudio.com/post/423/

3. Интернет-ресурс: <http://cppstudio.com/post/437/>

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

## Лабораторная работа №6

## СОРТИРОВКА В МАССИВАХ

**Цели работы**

Приобретение навыков составления и отладки программ:

* Изучение алгоритмов сортировки в массивах,
* Разработка многомодульных программ,
* Построение динамических массивов,
* Отработка принципов передачи массивов в качестве аргументов функций.

**Постановка задачи**

* Составить программу сортировки массива методом пирамидальной сортировки.
* В программе должен использоваться динамический массив.
* При инициализации исходного массива предусмотреть режим «ручного» ввода с клавиатуры (при размере массива n<=10) и режим генерации псевдослучайных чисел (при размере массива n>10).
* Инициализацию исходного массива, сортировку массива, распечатку массива оформить в виде отдельных функций.

**Описание алгоритма**

Рис. 1. Схема алгоритма программы main()

Начало

Инициализация mas, sizemas;

Ввод sizemas;

44/2

44/1

43/2

43/1

cout << "Массив с примененной пирамидальной сортировкой";

false

randomas(mas);

true

sizemas>=10

true

sizemas<1

false

cout<<”Введите корректное sizemas”;

handmas(mas);

build\_maxheap(mas,sizemas);

heapsort(mas, sizemas);

output(mas);

Конец

Рис. 1. Схема алгоритма программы main()

Начало

Инициализация \*mas, sizemas, randomansw;

i = 1;

false

i <=sizemas;

i++;

mas[i] = rand() % 1000;

Конец

true

Рис. 2. Схема алгоритма функции randomas()

Инициализация \*mas, sizemas;

Начало

cout << "Элемент массива [" << i << "]= ";

i++;

i = 1;

Конец

i <=sizemas;

false

true

cin>>mas[i];

Рис. 3. Схема алгоритма функции handomas()

Инициализация \*mas, sizemas;

Начало

Конец

i++;

i <=sizemas;

i = 1;

false

true

cout << "Элемент массива [" << i << "]= " << mas[i] << endl;

Рис. 4. Схема алгоритма функции output()

i = sizemas/2;

Начало

Инициализация \*mas, sizemas;

false

i>=1

Конец

i--;

true

max\_heapify(mas, i, sizemas);

Рис. 5. Схема алгоритма функции build\_maxheap()

Конец

i--;

temp = mas[i]; mas[i] = mas[1]; mas[1] = temp; max\_heapify(mas, 1, i - 1);

i>=2

Начало

Инициализация \*mas, sizemas, temp;

i = sizemas;

i>=2

temp = mas[i]; mas[i] = mas[1]; mas[1] = temp; max\_heapify(mas, 1, i - 1);

i--;

Конец

false

true

Рис. 6. Схема алгоритма функции heapsort()

**Описание программы**

Программа инициализирует массив размера sizemas (sizemas – глобальная переменная) и указатель на него. Далее пользователю предлагается ввести размер массива. Он проходит проверку на значение и тип данных посредством функции cin.fail. В случае, если размер массив превышает 10 – массив заполняется датчиком псевдослучайных чисел rand(), иначе – заполняется с клавиатуры. Оба метода реализованы подпрограммами:

handmas() - заполняет созданный массив mas элементами, вводимыми пользователем с клавиатуры. Получает лишь указатель на массив.

randomas() - заполняет созданный массив mas элементами, сгенерированными датчиком псевдослучайных чисел.

Далее объявленный массив сортируется пирамидальным методом, реализованным тремя подпрограммами:

build\_maxheap() – создание «кучи» из массива.

heapsort() – сортировка «кучи» и внутренней max\_heapify().

Отсортированный массив выводится на экран при помощи подпрограммы output(), которая представляет собой обычный cout в «красивой упаковке».

**Пирамидальная сортировка** (англ. Heapsort, «Сортировка кучей») — алгоритм сортировки, работающий в худшем, в среднем и в лучшем случае (то есть гарантированно) за Θ(n\*log n) операций при сортировке n элементов.

Может рассматриваться как усовершенствованная сортировка пузырьком, в которой элемент всплывает (min-heap) / тонет (max-heap) по многим путям.

Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево — это такое дерево, у которого выполнены условия:

1. Каждый лист имеет глубину либо {\displaystyle d}d, либо {\displaystyle d-1}d-1, {\displaystyle d}d — максимальная глубина дерева.
2. Значение в любой вершине не меньше (другой вариант — не больше) значения её потомков.

Удобная структура данных для сортирующего дерева — такой массив Array, что Array[0] — элемент в корне, а потомки элемента Array[i] являются Array[2i+1] и Array[2i+2].

Алгоритм сортировки будет состоять из двух основных шагов:

1. Выстраиваем элементы массива в виде сортирующего дерева

2. Будем удалять элементы из корня по одному за раз и перестраивать дерево.

Это требует n\*log n {\displaystyle O(n\log n)}операций.

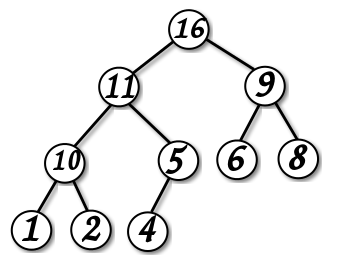
****

Рис. 7 – пример сортирующего дерева

C:\Users\mobil\AppData\Local\Microsoft\Windows\INetCache\Content.Word\306px-Сортирующее_дерево_развернутое_в_массив.svg.png

Рис. 8 – структура хранения данных

**Текст исходного кода программы с комментариями**

#include <iostream>

#include <cstdlib>

using namespace std;

int sizemas; //глобальный размер массива

void randomas(int \*mas) { //заполение массива датчиком псевдослучайных чисел

int randomansw;

for (int i = 1; i <= sizemas; i++)

{

mas[i] = rand() % 1000;

}

cout << "Посмотреть массив - 0. Пропустить - 1" << endl;

cin >> randomansw;

if (randomansw > 1) cout << "Неверно" << endl;

while (randomansw > 1) {

cout << "Выбор: ";

cin >> randomansw;

if (randomansw > 1) cout << "Неверно" << endl;

}

switch (randomansw) {

case 0:

for (int i = 1; i <= sizemas; i++)

{

cout << "Элемент массива [" << i << "]= " << mas[i] << endl;

}

break;

case 1:

break;

}

}

void handmas(int \*mas) { //ручной ввод массива

for (int i = 1; i <= sizemas; i++)

{

cout << "Элемент массива [" << i << "]= ";

cin >> mas[i];

if (mas[i] > 1000) cout << "Неверный размер: < 1.000." << endl;

while (mas[i] > 1000)

{

cout << "Элемент массива [" << i << "]= ";

cin >> mas[i];

if (mas[i] > 1000) cout << "Неверный размер: < 1.000." << endl;

}

if (mas[i] <1) cout << "Неверный размер; > 0." << endl;

while (mas[i] < 1)

{

cout << "Элемент массива [" << i << "]= ";

cin >> mas[i];

if (mas[i] > 1000000) cout << "Неверный размер: > 0." << endl;

}

}

}

void output(int \*mas){ //вывод полученного массива

for (int i = 1; i <= sizemas; i++)

{

cout << "Элемент массива [" << i << "]= " << mas[i] << endl;

}

}

void max\_heapify(int \*mas, int i, int sizemas) //работает с элеиентом и следующим за ним

{

int j, temp;

temp = mas[i];

j = 2\*i;

while (j <= sizemas) //внутри массива

{

if (j < sizemas && mas[j+1] > mas[j])

j = j+1;

if (temp > mas[j])

break;

else if (temp <= mas[j])

{

mas[j/2] = mas[j];

j = 2\*j;

}

}

mas[j/2] = temp;

return;

}

void heapsort(int \*mas, int sizemas) //сортировка

{

int i, temp;

for (i = sizemas; i >= 2; i--)

{

temp = mas[i];

mas[i] = mas[1];

mas[1] = temp;

max\_heapify(mas, 1, i - 1);

}

}

void build\_maxheap(int \*mas, int sizemas) //создаем кучу(пирамиду)

{

int i;

for(i = sizemas/2; i >= 1; i--)

{

max\_heapify(mas, i, sizemas);

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int \*mas;

mas = new int[sizemas];

cout << "Введите размер массива: ";

cin >> sizemas;

if (sizemas < 1 || cin.fail())

cout << "Неверно. Введите число от 1." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (sizemas<1) {

cout << "Введите размер массива: ";

cin >> sizemas;

if (sizemas<1|| cin.fail()) cout << "Неверно. Введите число от 1." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

if (sizemas >= 10)

randomas(mas);

else handmas(mas);

cout << endl;

cout << "Массив с примененной пирамидальной сортировкой" << endl;

cout << endl;

build\_maxheap(mas,sizemas);

heapsort(mas, sizemas);

output(mas);

cout << endl;

system("Pause");

return 0;

}

**Контрольные прогоны программы**

Тест №1 – ввод значения, соответствующего ручному ввода массива. Последующая сортировка полученного массива.

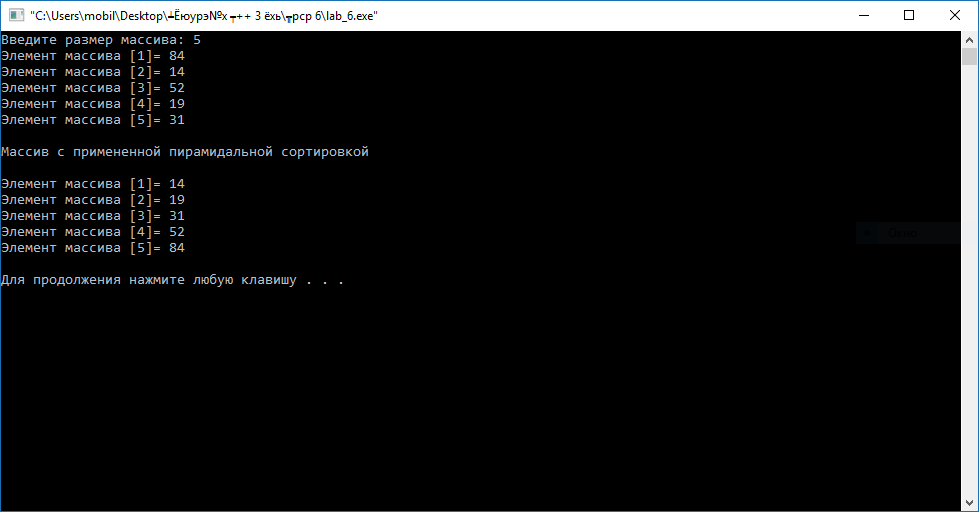


Рис. 9. Результат теста №1

Тест №2 – ввод значения, соответствующего заполнению массива псевдослучайными числами. Последующая сортировка полученного массива.

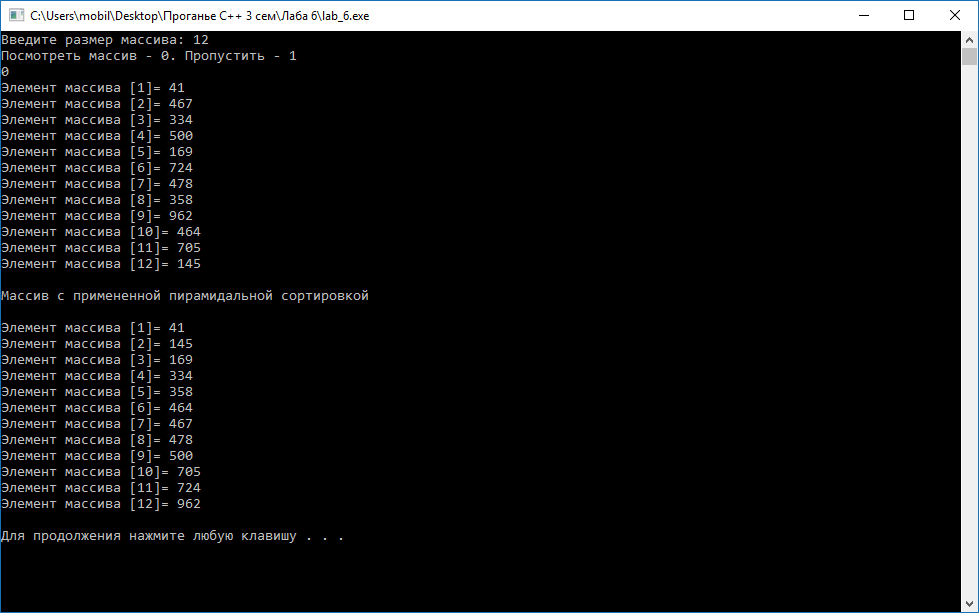


Рис. 10. Результат теста №2

Тест №3 – ввод значения, соответствующего заполнению массива псевдослучайными числами. Последующая сортировка полученного массива.

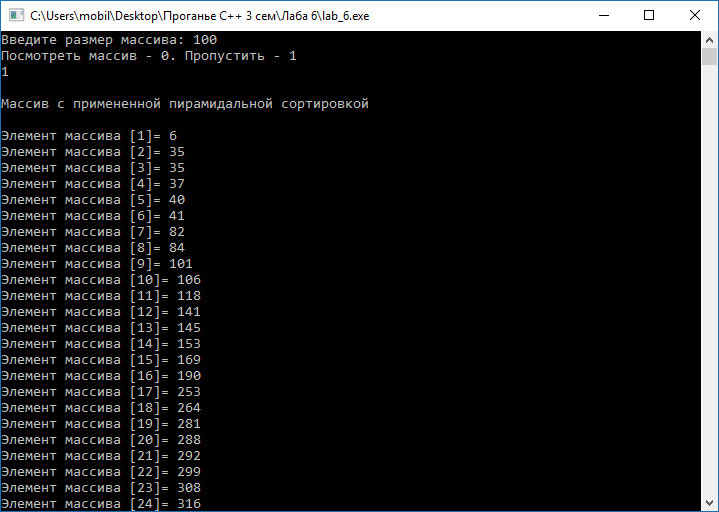


Рис. 11. Результат теста №3

Тест №4 – ввод значения, соответствующего заполнению массива псевдослучайными числами. Последующая сортировка полученного массива.

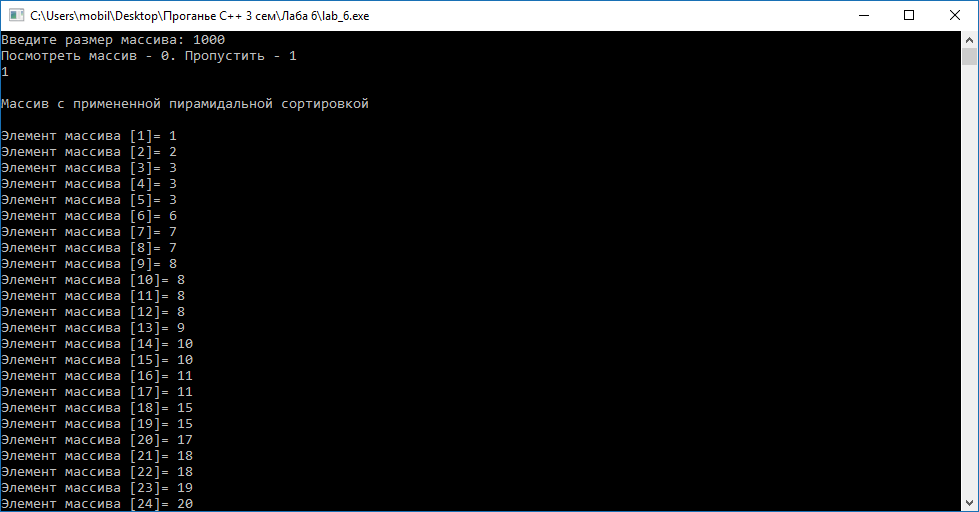


Рис. 12. Результат теста №4

**Выводы**

Данная программа позволила мне освоить работу с динамическими массивами, подпрограммами и сортировкой в С++.

Особенно стоит отметить работу с памятью в динамическом массиве. Очистка памяти с помощью функции delete() позволяет избежать утечки памяти. delete() находится в гармонии с new(), позволяя создавать высокопроизводительное ПО на С++ с малым потреблением оперативной памяти.

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: http://proprogramming.org/heap-sort-in-c/

3. Интернет-ресурс: http://cybern.ru/heapsortcpp.html

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

## 

## Лабораторная работа №7

## РАБОТА СО ДИНАМИЧЕСКИМИ СТРУКТУРАМИ ДАННЫХ

**Цели работы**

Приобретение навыков работы с очередями, а именно – с линейными односвязными списками - ЛОС.

Линейный односвязный список — это динамическая структура данных, каждый элемент которой посредством указателя связывается со следующим элементом.

**Постановка задачи**

Составить универсальную программу (псевдокласс) работы с очередями. Вызов функций работы с очередями (формирование очереди вводом с клавиатуры, вывод очереди на экран монитора, добавление элементов в очередь, удаление элементов из очереди, удаление (стирание) очереди, завершение работы с программой) оформить с помощью консольного меню. Очередь реализовать на базе двусвязного списка.

**Описание программы**

Программа объявляет переменные: размер списка, добавляемые при инициализации элементы, новый элемент, удаляемый элемент. Пользователь вводит с клавиатуры размер списка, это значение проходит проверку на тип данных при помощи cin.fail(). Проверяется и принадлежность к натуральным числам. В при некорректном вводе он повторяется.

Объявляется пустой список и инициализируется с клавиатуры пользователем. Исходный список выводится на экран. Далее пользователю предлагается добавить новый элемент в конец (push\_back) и начало (push\_front) списка. Далее требуется удалить (remove()) любой элемент на выбор. Измененный список выводится на экран. Далее список сортируется (sort()) по возрастанию и отображается. После всех операций список удаляется, программа завершается.

**Описание алгоритма**

Начало

Инициализация int element, newelement, deleted, search, answer; list<int> myList;

menu\_items();

i <100

i=1

cin >> answer;

false

Конец

answer < 0 || answer > 7

true

false

true

Неверно

actions(answer);

i++

Рис. 1. Схема алгоритма программы main()

Начало

cout << "Создать список - 0" << endl;

cout << "Отобразить список - 1" << endl;

cout << "Добавить новый элемент в конец - 2" << endl;

cout << "Добавить новый элемент в начало - 3" << endl;

cout << "Удалить элемент из списка. Ввести значение элемента - 4" << endl;

cout << "Отсортировать список по возрастанию - 5" << endl;

cout << "Очистить список - 6" << endl;

cout << "Завершить программу - 7" << endl;

Конец

Рис. 2. Схема алгоритма программы menu\_items()

Начало

Инициализация answer, size, answered;

answer

false

size < 1

0

cin>>size;

true

Неверно.

i=1

i <=size

false

break;

cin >> element;

true

myList.push\_back(element);

i++

64/2

64/1

Рис. 3. Схема алгоритма программы actions()

1

copy(myList.begin(), myList.end(), ostream\_iterator<int>(cout, " "));

63/2

63/1

2

myList.push\_back(newelement);

cin>>newelement;

3

myList.push\_front (newelement);

cin>>newelement;

myList.remove(deleted);

4

cin>>deleted;

myList.sort(); copy(myList.begin(), myList.end(), ostream\_iterator<int>(cout, " "));

5

6

myList.clear(); copy(myList.begin(), myList.end(), ostream\_iterator<int>(cout, " "));

65/4

65/3

Рис. 3. Схема алгоритма программы actions()

64/3

64/4

answerend <0 || answerend >1

7

cin>>answered;

Неверно.

false

true

answered

0

break;

exit(0);

1

Конец

Рис. 3. Схема алгоритма программы actions()

**Текст исходного кода программы с комментариями**

#include <iostream>

#include <list>

#include <iterator>

#include <cstdlib>

using namespace std;

int element, newelement, deleted, search, answer;

list<int> myList;

void menu\_items() {

cout << "Создать список - 0" << endl;

cout << "Отобразить список - 1" << endl;

cout << "Добавить новый элемент в конец - 2" << endl;

cout << "Добавить новый элемент в начало - 3" << endl;

cout << "Удалить элемент из списка. Ввести значение элемента - 4" << endl;

cout << "Отсортировать список по возрастанию - 5" << endl;

cout << "Очистить список - 6" << endl;

cout << "Завершить программу - 7" << endl;

cout << endl;

}

int actions(int answer) {

switch (answer) {

case 0:

int size;

cout << "Введите длину списка: ";

cin >> size;

if (size < 1 || cin.fail())

cout << "Неверно. Введите число от 1." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (size<1) {

cout << "Введите длину списка: ";

cin >> size;

if (size<1 || cin.fail()) cout << "Неверно. Введите число от 1." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

for (int i = 1; i <= size; i++) {

cout << "Введите элемент списка " << i << ": ";

cin >> element;

myList.push\_back(element); // добавляем в список новые элементы

}

cout << endl;

break;

case 1:

copy(myList.begin(), myList.end(), ostream\_iterator<int>(cout, " ")); // вывод на экран элементов списка

cout << endl;

break;

case 2:

cout << "Добавить новый элемент в конец: " << endl;

cin >> newelement;

myList.push\_back(newelement);

cout << endl;

break;

case 3:

cout << "Добавить новый элемент в начало: " << endl;

cin >> newelement;

myList.push\_front(newelement);

cout << endl;

break;

case 4:

cout << "Удалить элемент из списка. Введите значение элемента: ";

cin >> deleted;

myList.remove(deleted);

cout << endl;

break;

case 5:

cout << "Отсортированный список:" << endl;

myList.sort();

copy(myList.begin(), myList.end(), ostream\_iterator<int>(cout, " "));

cout << endl;

break;

case 6:

cout << "Список был очищен. Текущий список: ";

myList.clear();

copy(myList.begin(), myList.end(), ostream\_iterator<int>(cout, " "));

cout << endl;

break;

case 7:

int answerend;

cout << "Вы точно хотите выйти? Введите 1 для завершения программы и 0 - для продолжения работы: ";

cin >> answerend;

if (answerend <0 || answerend >1 || cin.fail())

cout << "Неверно." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (answerend <0 || answerend >1) {

cout << "Сделайте выбор:";

cin >> answerend;

if (answerend <0 || answerend >1 || cin.fail()) cout << "Неверно." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

switch (answerend) {

case 0:

break;

case 1:

exit(0);

}

break;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

menu\_items();

for (int i = 0; i < 100; i++) {

cout << "Выберете желаемую функцию: ";

cin >> answer;

if (answer < 0 || answer > 7 || cin.fail())

cout << "Неверно." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (answer < 0) {

cout << "Сделайте выбор: ";

cin >> answer;

if (answer < 0 || answer > 7 || cin.fail()) cout << "Неверно." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

actions(answer);

}

system("Pause");

return 0;

}

**Контрольный прогон программы**

Тест №1 – выполнение пункта «0».

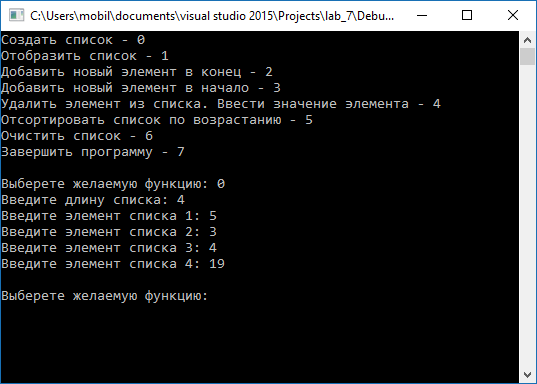


Рис. 4. Результат теста №1

Тест №2 – выполнение пункта «1».

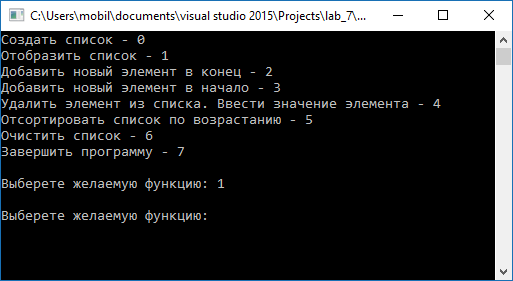


Рис. 5. Результат теста №2

Тест №3 – выполнение пункта «2».

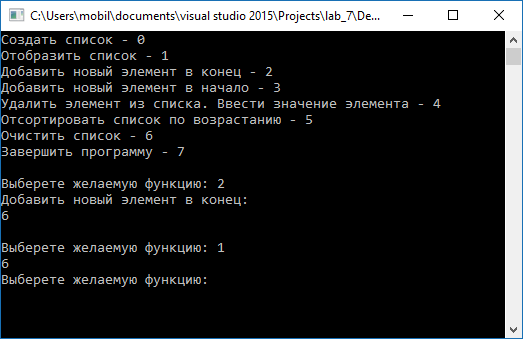


Рис. 6. Результат теста №3

Тест №4 – выполнение пункта «3».

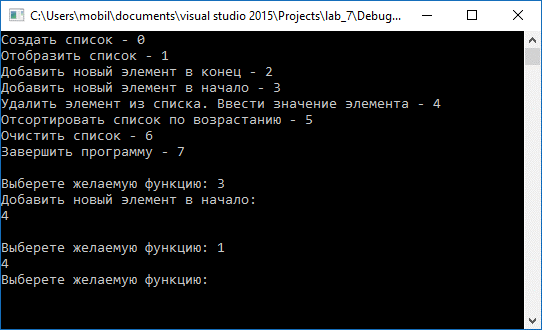


Рис. 7. Результат теста №4

Тест №5 – выполнение пункта «4».

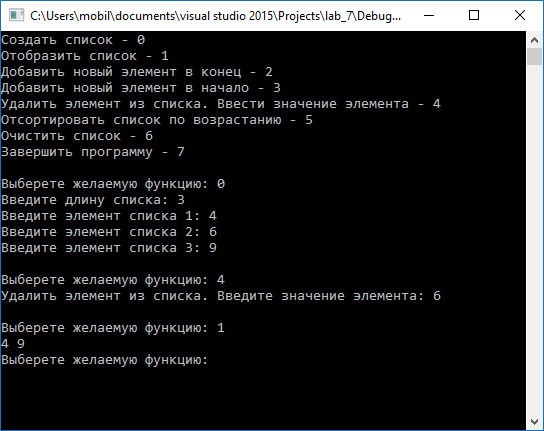


Рис. 8. Результат теста №5

Тест №6 – выполнение пункта «5».

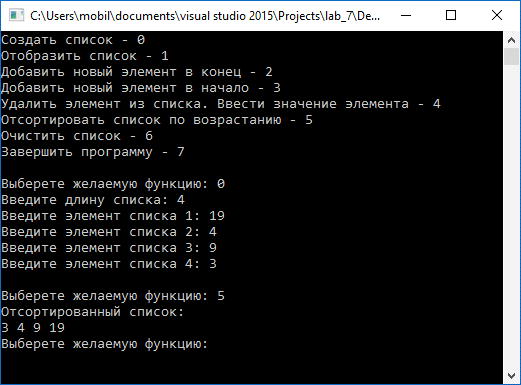


Рис. 9. Результат теста №6

Тест №7 – выполнение пункта «6».

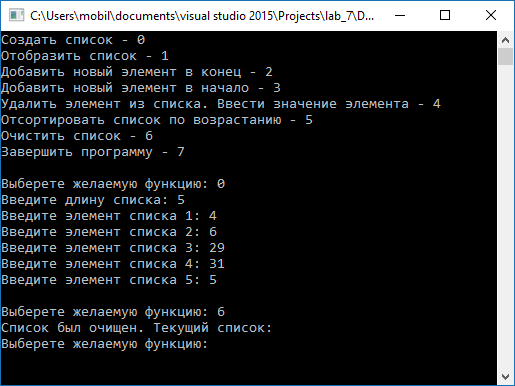


Рис. 10. Результат теста №7

Тест №8 – выполнение пункта «7».

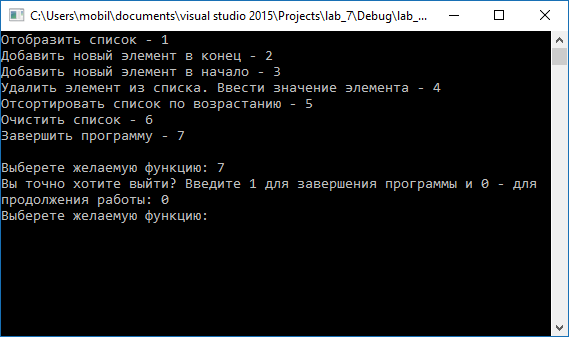


Рис. 11. Результат теста №8

**Выводы**

Данная программа позволила мне освоить работу с ЛОС в С++. Данный раздел для меня абсолютно новый, оттого выше его цена. Хочу отметить, что работа со списками отличается от массивов, но при отсутствии индексации в программе списки в разы удобнее.

**Список используемых информационных источников**

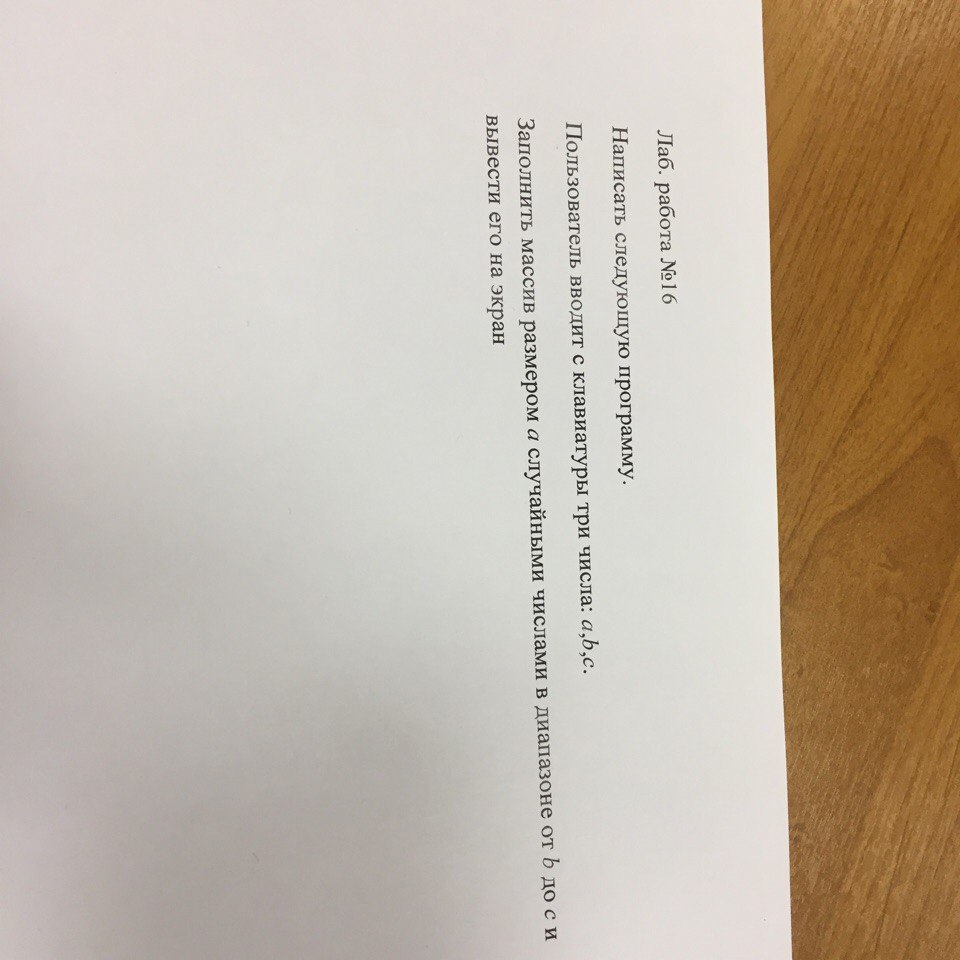
1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: http://cppstudio.com/post/8482/

3. Интернет-ресурс: http://www.cplusplus.com/reference/list/list/clear/

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

**ЭСКИЗ ПО ПРОГРАММИРОВАНИЮ №1**

**Массив случайных чисел**

**Цель работы**

Отточить навыки работы с динамическими массивами и случайными числами.

**Постановка задачи**

**Описание программы**

Программа объявляет переменные и динамический массив, указатель на него.

Пользователю предлагается ввести с клавиатуры числа a, b, c. Введенные данные проходят проверку на принадлежность к типу данных по с помощью функции cin.fail(). Также проверяется принадлежность к натуральным числам.

Далее инициализируется переменная delta, необходимая для работы генератора псевдослучайных чисел rand(). В цикле for массив размера a заполняется случайными числами. Он выводится на экран.

**Описание алгоритма**

Начало

Инициализация a, b, c, \*mas;

cout<<”Введите натуральное a”;

a>=1

Ввод a;

true

Ввод b;

b>=1

cout<<”Введите натуральное b”;

false

true

false

76/1

Рис. 1. Схема алгоритма программы main()

71/1

Ввод с;

Инициализация

delta = c - b + 1;

true

false

с>=1

cout<<”Введите натуральное с”;

i = 1;

mas[i] = rand() % delta + b;

i++;

true

i <=a;

false

77/2

Рис. 1. Схема алгоритма программы main()

76/2

cout << "Массив:" << endl;

i = 1;

cout << "Элемент массива [" << i << "]= " << mas[i] << endl;

i++;

true

i <=a;

false

Конец

Рис. 1. Схема алгоритма программы main()

**Текст исходного кода программы**

#include <iostream>

#include <cstdlib>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int a, b, c, \*mas;

cout << "Введите a: ";

cin >> a;

if (a<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (a <1) {

cout << "Введите a: ";

cin >> a;

if (a<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

mas = new int[a];

cout << "Введите b: ";

cin >> b;

if (b<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (b <1) {

cout << "Введите b: ";

cin >> b;

if (b<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

cout << "Введите c: ";

cin >> c;

if (c<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

while (c <1) {

cout << "Введите c: ";

cin >> c;

if (c<1 || cin.fail())

cout << "Неверно. Введите натуральное число." << endl;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

}

int delta = c - b + 1;

for (int i = 1; i <= a; i++)

{

mas[i] = rand() % delta + b;

}

cout << endl;

cout << "Массив:" << endl;

for (int i = 1; i <= a; i++)

{

cout << "Элемент массива [" << i << "]= " << mas[i] << endl;

}

system("Pause");

return 0;

}

**Контрольный прогон программы**

Тест №1 – выполнение программы при значении a = 9, b = 2, c = 4.

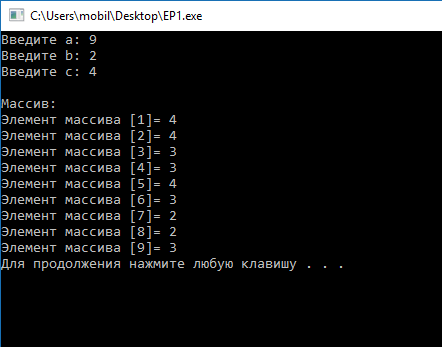


Рис. 2. Результат теста №1

**Выводы**

Данная программа позволила мне отточить работу с динамическими массивами и датчиком псевдослучайных чисел rand() в С++.

**Список используемых информационных источников**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: <http://cppstudio.com/post/834/>

4. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

5. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

**ЭСКИЗ ПО ПРОГРАММИРОВАНИЮ №2**

**Анализ рекурсивных функций**

**Цель работы**

Провести анализ работы рекурсивных функций в С++ и их критического завершения. Изучить их «штатное» функционирование и выявить причины ошибок.

**Постановка задачи**

Составить три рекурсивные функции и провести серию тестов их работоспособности при разных условиях их завершения.

**Описание программ**

В анализе будет использовано 3 функции:

1 – нахождение ряда чисел Фибоначчи fib().

2 – нахождение факториала числа fact()

3 – рекурсивная функция main()

Каждая из них будет содержать ошибку, приводящую к переполнению отводимой под хранение данных программы памяти. Мы выясним, при каком условии это происходит и как с этим бороться.

**Описание алгоритма программы Фибоначчи**

Начало

Инициализация i=0;

false

i<50?

true

Fib(i);

cout << Fib(i) << endl;

i++;

Конец

Рис. 1. Схема алгоритма программы main()

Начало

Инициализация i=0;

true

i<1?

return 0;

false

true

i==0?

return 1;

false

return Fib(i - 1) + Fib(i - 2);

Конец

Рис. 2. Схема алгоритма программы Fib()

**Текст исходного кода программы**

#include <iostream>

#include <cstdlib>

using namespace std;

int Fib(int i)

{

int value = 0;

if (i < 1) return 0;

if (i == 1) return 1;

return Fib(i - 1) + Fib(i - 2);

}

int main()

{

int i = 0;

while (i < 50)

{

cout << Fib(i) << endl;

i++;

}

system("Pause");

return 0;

}

**Описание алгоритма программы нахождения факториала**

Начало

Инициализация N=0;

N<20?

fact(N);

cout << N << ": " << fact(N) << endl;

N++;

Конец

Рис. 3. Схема алгоритма программы main()

Начало

Инициализация N=0;

true

N<0?

return 1;

return 0;

false

true

N==0?

false

return N \* fact(N - 1);

Конец

Рис. 4. Схема алгоритма программы fact()

**Текст исходного кода программы**

#include <iostream>

#include <cstdlib>

using namespace std;

int fact(int N)

{

if (N < 0) // если пользователь ввел отрицательное число

return 0; // возвращаем ноль

if (N == 0) // если пользователь ввел ноль,

return 1; // возвращаем факториал от нуля - не удивляетесь, но это 1 =)

else // Во всех остальных случаях

return N \* fact(N - 1); // делаем рекурсию.

}

int main()

{

int N = 0;

while (N < 20) {

cout << N << ": " << fact(N) << endl;

N++;

}

system("Pause");

return 0;

}

**Описание алгоритма программы**

Начало

cout << "Hello World" << endl;

main();

Конец

Рис. 5. Схема алгоритма программы main()

**Текст исходного кода программы**

#include <iostream>

using namespace std;

int main()

{

cout << "Hello World" << endl;

main();

system("Pause");

return 0;

}

**Контрольные прогоны программ**

Тест №1 – десятиминутное выполнение программы fib(). 47 чисел – предел размерности типа данных int. 48-е число дает в консоли отрицательное значения, превышая отводимую под его хранение память.

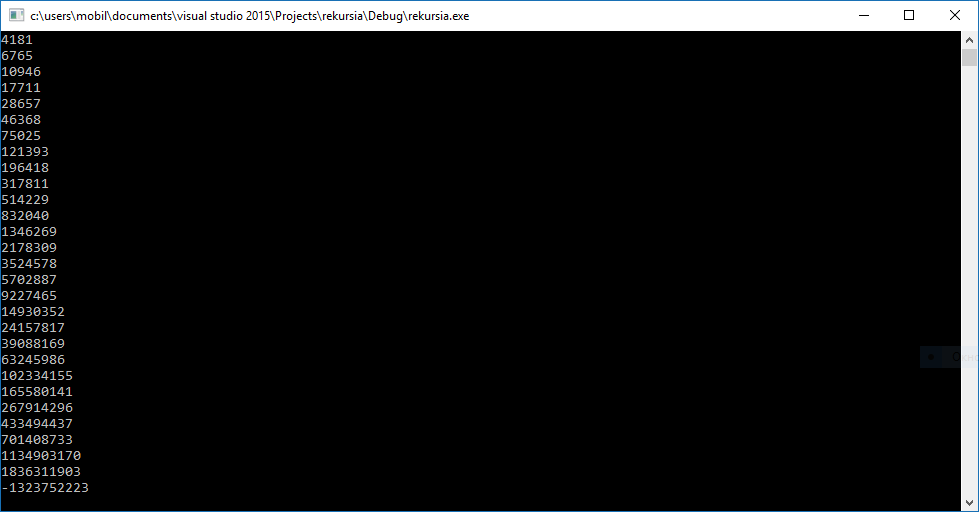


Рис. 6 – результаты теста №1

Тест №2 – выполнение программы fact(). Как видно из результатов запуска программы, 16 – последнее число, факториал которого умещается в int. Последующие числа требуют более вместительных типов данных.

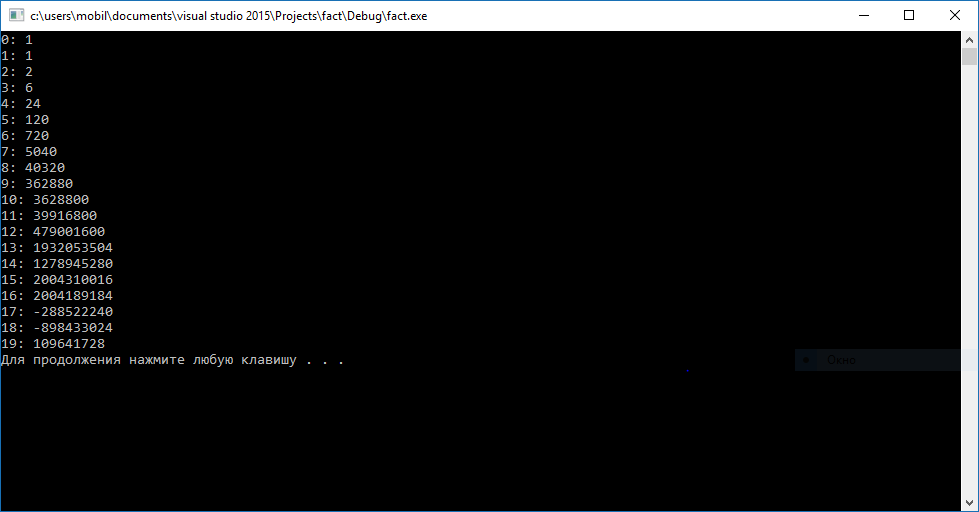


Рис. 7 – результаты теста №2

Тест №3 – выполнение программы main(). После нескольких секунд циклического выведения надписи “Hello World” программа сообщает о переполнении стека и перестает выполняться.

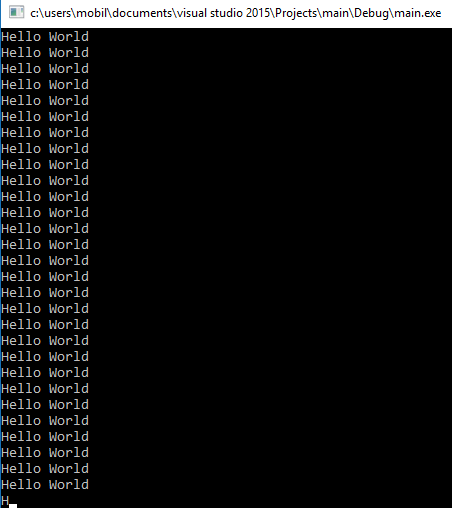


Рис. 8 – результаты теста №3

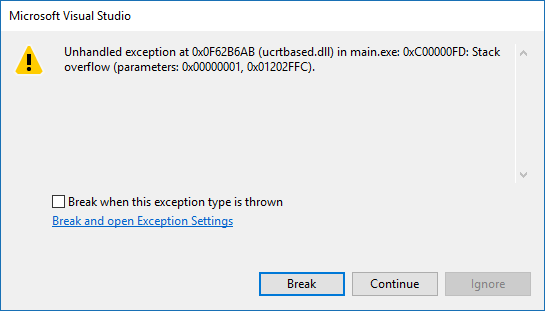


Рис. 9 – результаты теста №3

**Выводы**

Причина некорректной работа программ fact() и fib() – тип целочисленных данных int в C++. Этот тип позволяет хранить лишь число, не превышающее 2 147 483 647. Если значение, записанное данным типом данных, превышает его – наблюдается некорректное отображение, будь то отрицательное число, или последующие числа меньшего значения. Решение кроется в использовании более вместительных типов данных или соблюдение границ:

int – до 2 147 483 647

unsigned int – до  4 294 967 295

double – до  9 223 372 036 854 775 807.0

Программа main() же не имеет условий завершения, при работе использует всю отводимую ей память. Решение – создание этих условий. Например, счетчик вызова функции.

**Список литературы**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: http://cppstudio.com/post/271/

3. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

4. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.

**Приложение к отчету №1**

В связи с некорректным результатом вычислений программы были произведены ручные расчеты функций x, y, и z. Расчеты подтвердили некорректность исходных данных.

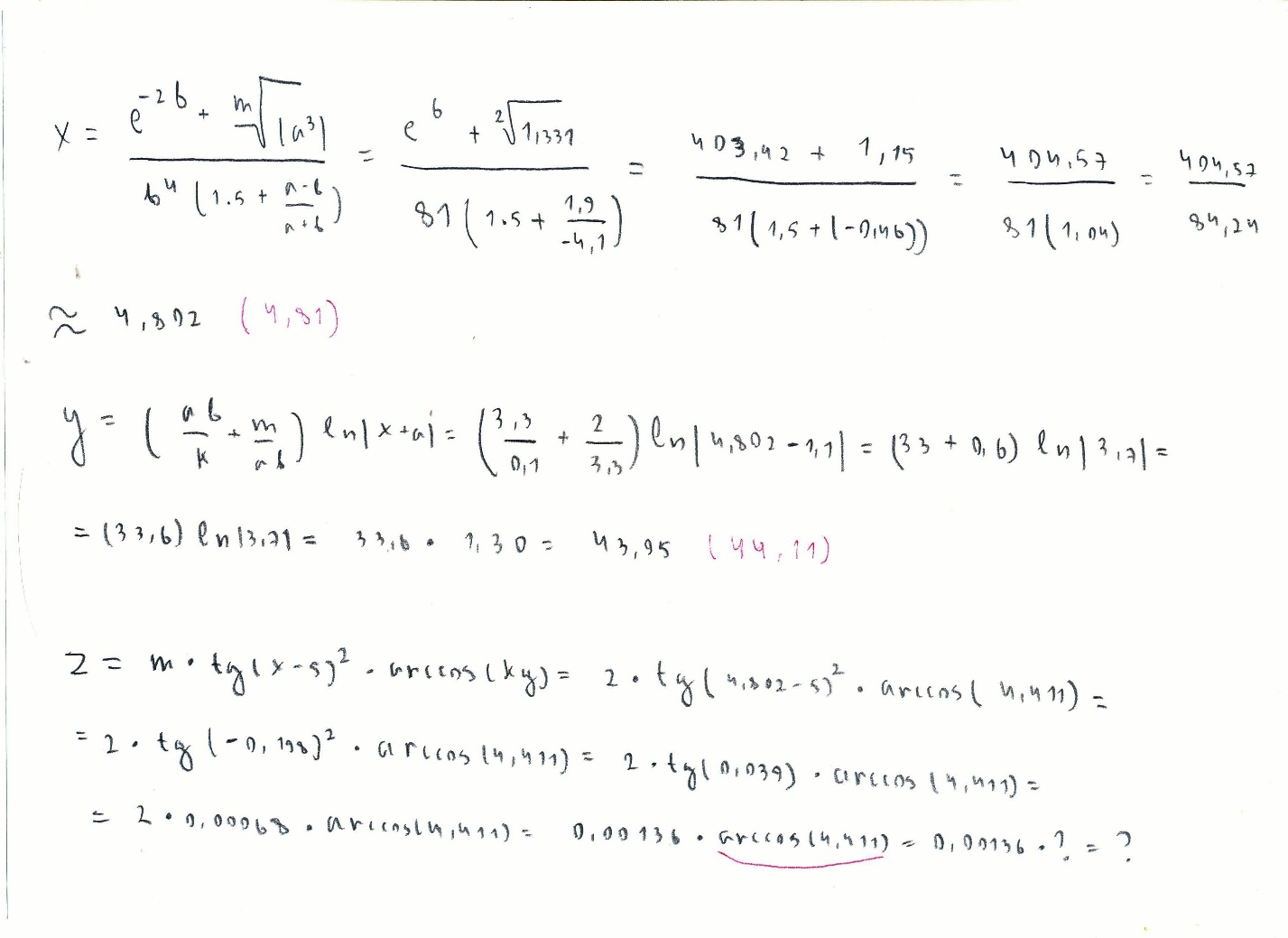
****

Рис 3. Ручные расчёты.

**Приложение к отчету №3**

В ходе выполнения лабораторных работы я столкнулся с ошибкой, возникающей при использовании проверки вводы, реализованной средствами cin.fail(), описанной подробно в отчёте №2 (стр. 8). Проверка корректно работает в лабораторной работе №2, но в лабораторной работе №3 приводит к преждевременному закрытию программы при вводе буквенного символа. Если сначала ввести число, не удовлетворяющее требованиям программы, а потом букву – программа продолжит работу. Если же ввести сначала букву – ошибка выполнения.

Еще более интересный факт: данная ошибка встречается лишь в скомпилированном exe-файле и отсутствует при запуске в Microsoft Visual Studio. Абсолютно идентичный код проверки ввода корректно работает во второй работе при любых условиях и не работает в последующих.

В качестве подтверждения привожу два видеофайла, демонстрирующие выполнение лабораторных работ №2 (lab\_2.mp4) и №3 (lab\_3.mp4) при условии того, что проверка была реализована одинаково, обе программы работают с одинаковым типом данных.

Данные видеофайлы хранятся в папке «Ошибка cinfail».

**Заключение**

Эти 7 лабораторных работ и 2 эскиза значительно повысили мои навыки во владении языком С++. Первые работы позволили мне отточить до автоматизма классические действия с данными, а последние – дали мне много новых знаний, в том числе и о самих этих данных. Абсолютно новая сфера – очереди и списки. Ранее я не работал с ними, и честно сказать – мне они понравились. Оценю курс занятий высоко, отмечу простоту подачи материала Владиславом Петровичем. Буду изучать С++ и дальше, за границами курса.

Но в программировании важно не только умение писать компактный и эффективный код, но и умение доносить его до пользователя в виде конечного ПО. В данном курсе я научился создавать системно-независимые exe-приложения для OC Windows при помощи инструментов Microsoft Visual Studio. А именно: я использовал VS2015 x64 x86 Cross Tools Command Prompt.

В консольном интерфейсе при помощи команды cd получаем доступ к местоположению файла исходного кода cpp. Далее вводим «cl program.cpp». Инструмент создаст нам exe-файл с полным списком необходимых dll, упакованных в него.

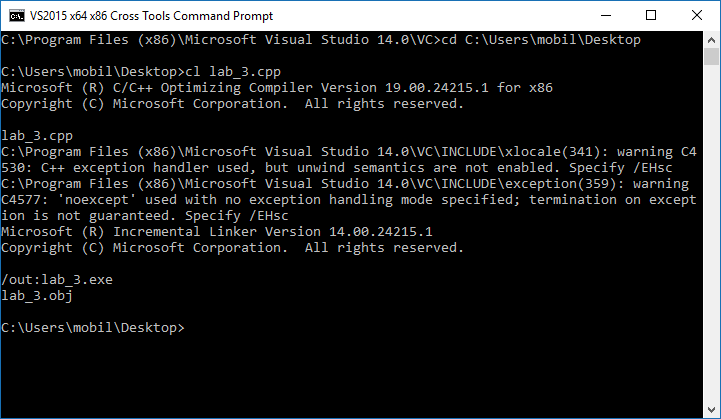


Рис. 1 – создание системно-независимого exe-файла

**Список литературы**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня- Санкт-Петербург. Питер, 2006.

2. Интернет-ресурс: [http://cppstudio.com/](http://cppstudio.com/post/834/)

3. Интернет-ресурс: <https://code-live.ru/tag/cpp-manual/>

4. Интернет-ресурс: http://proprogramming.org

5. Единая система программной документации (ЕСПД) ГОСТ 19.701-90

6. Герберт Шилдт. С++. Базовый курс. Москва. Вильямс, 2010.

7. Сыромятников В.П. «Программирование», лекции, МТУ МИРЭА, 2016/2017.