МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
РУТ (МИИТ)

Кафедра «Вычислительные системы, сети и информационная безопасность»

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»

ЛАБОРАТОРНАЯ РАБОТА №3

Направление: 10.03.01 *Информационная безопасность*

Профиль: *Безопасность компьютерных систем*

Выполнил:
студент группы УИБ-115
Шевченко Димитрий

Проверил:

_____
(должность, ФИО)

_____
(должность, ФИО)

Москва 2021 г.

Задание №3

Задание: Вычислить с заданной точностью **e** сумму:

y=1-x2/2!+x4/4!-...(-1)n*x2n/(2n)!+…

Проверка: **y=cos(x)**

1 Таблица имён:

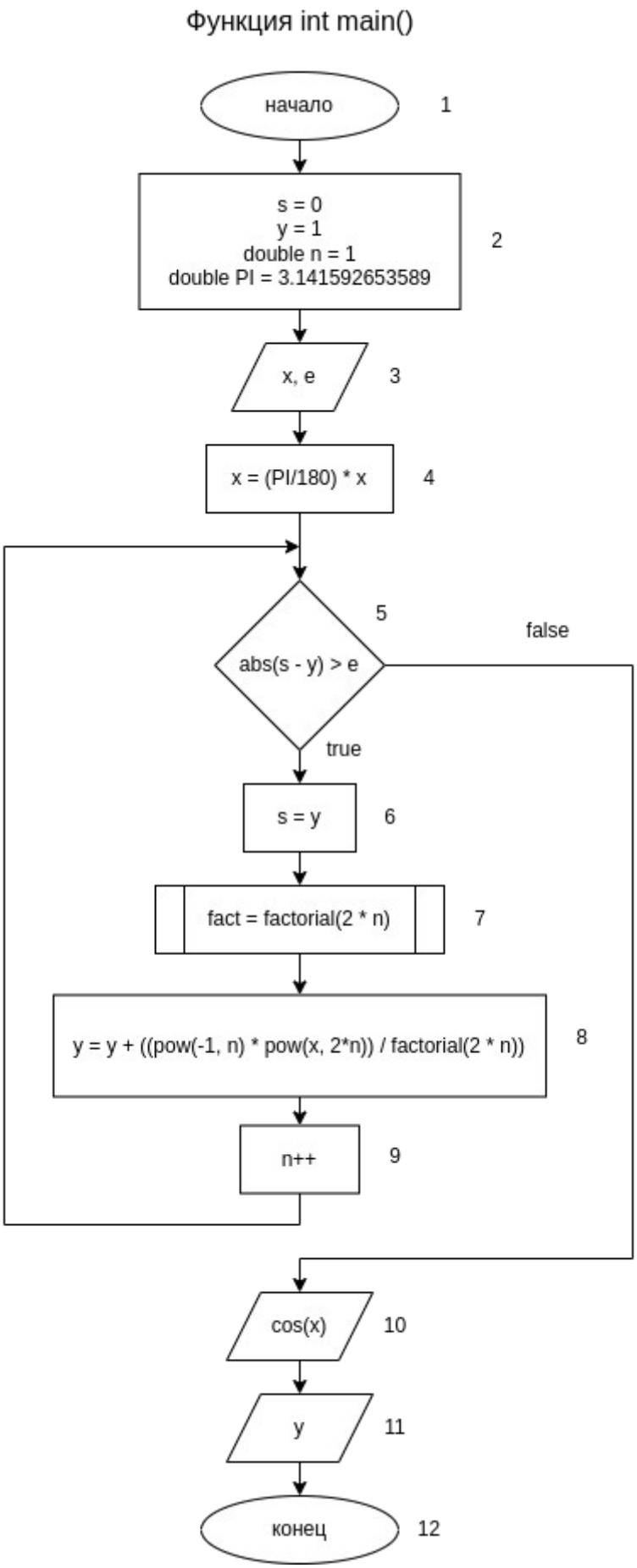
Функция factorial:

Рабочие переменные		
x	Double (дробное)	Искомое число факториала

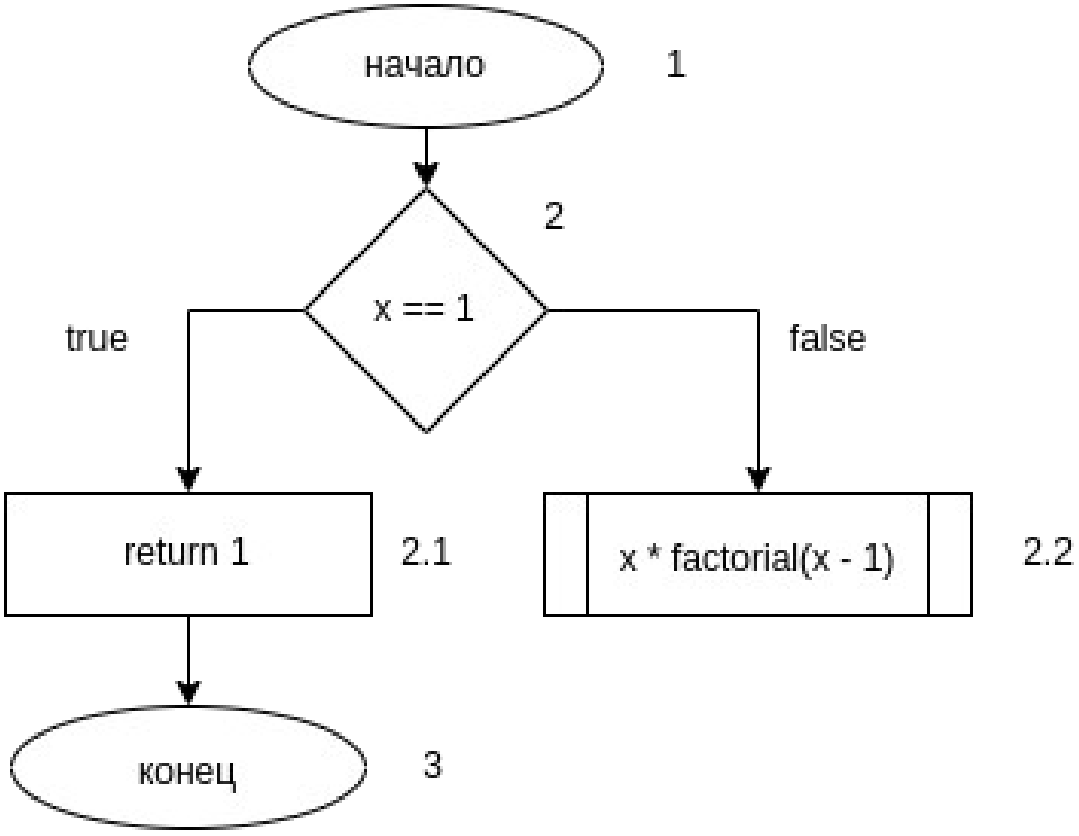
Функция main:

Исходные данные		
x	Double (дробное)	Искомое число
e	Double (дробное)	Точность
Рабочие переменные		
n	Double (дробное)	Переменная функции
s	Double (дробное)	Хранение прошлого значения y
PI	Double (дробное)	Число π
fact	Double (дробное)	Значение факториала
Результат		
y	Double (дробное)	Результат вычисления уравнения

2 Блок-схема:


Функция int main()
начало
1
s = 0
y = 1
double n = 1
double PI = 3.141592653589
2
x, e
3
x = (PI/180) * x
4
5
abs(s - y) > e
false
true
s = y
6
fact = factorial(2 * n)
7
y = y + ((pow(-1, n) * pow(x, 2*n)) / factorial(2 * n))
8
n++
9
cos(x)
10
y
11
конец
12

Функция double factorial(double x)



3 Отладочные примеры:

Вариант I
1) Начало
2) y = 1; n = 1; PI = 3.141592653589;
3) ввод x и e
x = 5
e = 0.1

4) x = (3.141592653589/180) * 5 = 0.08726646259969445

5) Пока |(s — y)| больше 0.1, выполнять
    6) присваиваем s значение переменной y (1)
    7) fact = factorial(2 * n)
    8) присваиваем y значение y + ((pow(-1, n) * pow(x, 2*n)) / fact)
    9) увеличиваем переменную n на 1

8) Вывод cos(x) и y
cos(x) = 0.996195
y = 0.996192

9) Конец

Вариант II
1) Начало
2) y = 1; n = 1; PI = 3.141592653589;
3) ввод x и e
x = 10
e = 0.00001

4) x = (3.141592653589/180) * 10 = 0.1745329251993889

5) Пока |(s — y)| больше 0.00001, выполнять
    6) присваиваем s значение переменной y (1)
    7) fact = factorial(2 * n)
    8) присваиваем y значение y + ((pow(-1, n) * pow(x, 2*n)) / fact)
    9) увеличиваем переменную n на 1

8) Вывод cos(x) и y
cos(x) = 0.984808
y = 0.984808
-0.40142572795859444
9) Конец

Вариант III
1) Начало
2) y = 1; n = 1; PI = 3.141592653589;
3) ввод x и e
x = -23
e = 0.000000001

4) x = (3.141592653589/180) * -23 = -0.40142572795859444

5) Пока |(s — y)| больше 0.000000001, выполнять
    6) присваиваем s значение переменной y (1)
    7) fact = factorial(2 * n)
    8) присваиваем y значение y + ((pow(-1, n) * pow(x, 2*n)) / fact)
    9) увеличиваем переменную n на 1

8) Вывод cos(x) и y
cos(x) = 0.920505
y = 0.920505

9) Конец

Вариант IIII
1) Начало
2) y = 1; n = 1;
3) ввод x и e
x = -420
e = 0.0000000000000000000000000000000000000000001

4) x = (3.141592653589/180) * -420 = -7.3303828583743336

5) Пока |(s — y)| больше 0.0000000000000000000000000000000000000000001, выполнять
    6) присваиваем s значение переменной y (1)
    7) fact = factorial(2 * n)
    8) присваиваем y значение y + ((pow(-1, n) * pow(x, 2*n)) / fact)
    9) увеличиваем переменную n на 1

8) Вывод cos(x) и y
cos(x) = 0.5
y = 0.5

9) Конец

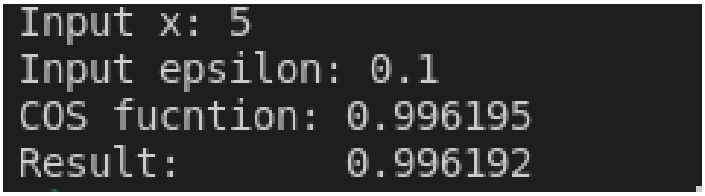
4 Код программы:

```
#include <iostream> //подключение библиотеки функции ввода-вывода
#include <cmath> // подключение библиотеки для работы с мат. операциями
using namespace std; //подключение пространства имён std
double factorial(double x){
	if (x == 1){ // если x равен 1, то
		return 1; // вернуть 1
	}
	else { // иначе
		return x * factorial(x - 1);
		// вернуть x умноженный на факториал (x - 1)
	}
}

int main(){
	double e; // заданная точность
	double x; // искомое значение
	double s = 0, y = 1; // s - хранит предыдущее значение функции,
	y - текущее
	double n = 1; // переменная вычисления функции
	double PI = 3.141592653589; // число пи для перевода x из градусов в
	радианы
	double fact; // получает значение функции factorial()

	cout << "Input x: ";
	cin >> x; // ввод x

	cout << "Input epsilon: ";
	cin >> e; //ввод точности
```

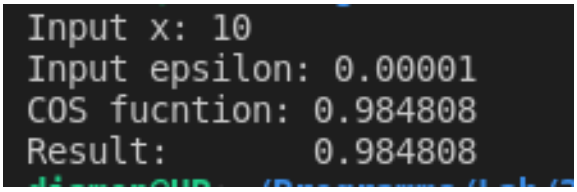
```
    x = (PI/180) * x;  //перевод x в радианы для правильного вычисления

    while (abs(s - y) > e){ // Разница между предыдущим и нынешним
    значением заданной функции для правильного вычисления точности.
        s = y; // присваиваем s значение предыдущего значения y
        fact = factorial(2 * n); // вычисление факториала для текущего n
        y = y + ((pow(-1, n) * pow(x, 2*n)) / fact); // вычисление функции
        n++; // увеличение переменной функции
    }

    // вывод проверки и результата работы программы соответственно
    cout << "COS fucntion: " << cos(x) << endl;
    cout << "Result:       " << y << endl;

    return 0; // успешное завершение программы
}
```

5 Результат выполнения работы программы:

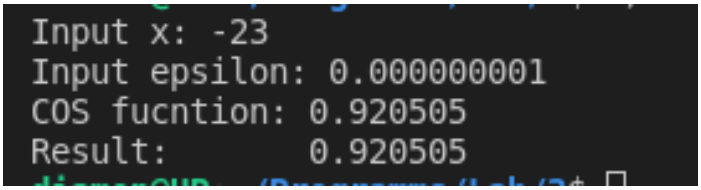
1)



2)



3)



4)

6 Вывод:

В ходе выполнения работы были изучены алгоритм вычисления косинуса и факториала, а также работы с ними.

Была проделана работа по созданию простейшего терминального UI, алгоритма поиска косинуса и факториала, проверке значений, вводу и выводу данных, тестированию и защите блок-схем с программным кодом.

На контрольных примерах мы убедились, что программа работает корректно и отвечает заданным в ТЗ требованиям.

Был оформлен комплект документации на программный код.