Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Заведующий
методическим кабинетом
_____Е.В.Фалей
« ____ » _____ 2016

Специальность: 2-40 01 01 «Программное обеспечение информационных технологий»	Дисциплина: «Основы алгоритмизации и программирование»
Составлена в соответствии с тематическим планом, утвержденным директором колледжа 31.08.2016	

Лабораторная работа № 5
Инструкционно-технологическая карта

Тема: Разработка, отладка и испытание циклических алгоритмов и программ с заданным числом повторений.
Цель: Научиться применять операторы повторения (цикла) при программировании заданного числа повторений (итераций) некоторой группы выражений в программе. Научиться составлять условия, задающие необходимое число итераций цикла.
Время выполнения: 2 часа

1. Краткие теоретические сведения

Оператор цикла for

Прочитать нижеследующий текст, выполнить примеры кода

Общий вид оператора:
for(*инициализирующее_выражение*; *условие_выполнения_цикла*; корректирующее_*выражение*)
{
тело цикла//выражения одной итерации цикла
}
Инициализирующее выражение выполняется **только один раз** в начале выполнения цикла и, как правило, инициализирует счетчик цикла (итерационную переменную цикла). В этом выражении можно продекларировать и/или проинициализировать одну или несколько итерационных переменных цикла. Примеры:

```
int i = 0;
int i = 100, j = -50;
int i = k;
int i = 0, j = ++i, k = pow(w, 2);
int i = j = k = 6;
```

Выражение проверки условия выполнения цикла содержит операцию отношения, которая выполняется в начале каждого цикла после выражения инициализации и далее перед каждой итерацией цикла. Если проверка условия дала ИСТИНО (то есть true, 1, «да, верно, так и есть»), то выполняется очередная итерация этого цикла, иначе (проверка закончилась результатом ЛОЖНО (false, 0, «нет, не верно, это не так») выполняется следующее за телом цикла выражение, то есть данный цикл прекращается. В выражении проверки может находиться составное проверяющее выражение, части которого объединяются знаками алгебры логики И, ИЛИ, исключающее ИЛИ: &, |, ^. Одно или несколько проверяющих выражений можно брать в круглые скобки и инвертировать (изменять его результат на **противоположный**) знаком НЕ – «восклицательный знак»: !. Примеры выражений проверки:

a>5;
a<=9 & b==3;
i<10 & j>=0 | j%2==0;
a>1 ^ b!=5;
!(i<10 & j>=0 | j%2==0);
i<10 & !(j>=0) | j%2==1;
true;
1;

выражения с булевскими переменными.

Корректирующее выражение (коррекция), как правило, предназначено для изменения значения счетчика цикла. Модификация счетчика происходит после каждого выполнения тела цикла.

Тело цикла – это одно (в таком случае его можно не брать в фигурные скобки) или несколько выражений (тогда они **должны** быть помещены в фигурные скобки { }), которые выполняются каждый раз, когда *условие_выполнения_цикла* дает результат ИСТИНО («да, верно»). Тело цикла – это одна итерация цикла, то есть набор действий, которые выполняются столько раз, сколько «витков» будет выполнено в данном цикле. Собственно, итерация – это один «виток» прохождения по циклу.

Пример цикла for(;;){}

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n-------------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    //создадим цикл на 10 итераций: с нулевой итерации до девятой итерации включительно
    for (int i = 0; i < 10; i = i + 1)
    {//в теле цикла печатается номер итерации, это же число возводится в квадрат и из него извлекается квадратный корень
        cout << "Итерация № " << i << ":\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
    }

    cout << "-------------------------------\n\tЦикл завершен.\n";//этот код НЕ относится к циклу for(;;){}
    system("pause");
    return 0;
}
```

Результат работы программы:

```
D:\2019\Labs\x64\Debug\Lab3.exe
Итерация № i    i^2     sqrt(i)
------------------------------
Итерация № 0:   0       0
Итерация № 1:   1       1
Итерация № 2:   4       1.41421
Итерация № 3:   9       1.73205
Итерация № 4:   16      2
Итерация № 5:   25      2.23607
Итерация № 6:   36      2.44949
Итерация № 7:   49      2.64575
Итерация № 8:   64      2.82843
Итерация № 9:   81      3
------------------------------
        Цикл завершен.
Для продолжения нажмите любую клавишу . . .
```

Цикл может создавать значения не только в сторону увеличения, но и в сторону уменьшения, например, от 15 до нуля. Создадим цикл, в котором итерационная переменная (счетчик цикла) изменяется от 15 до 0 включительно, тем самым обеспечивая 16 итераций. В выражении коррекции итерационная переменная должна уменьшаться на единицу, а в выражении проверки условия_выполнения_цикла нужно написать условие «пока i больше – 1», то есть «пока i больше либо равно нулю».

```
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n------------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    //создадим цикл на 16 итераций: счетчик цикла изменяется с 15 до 0 включительно
    for (int i = 15; i > -1; i = i - 1)//счетчик цикла УМЕНЬШАЕТСЯ на единицу
    {//в теле цикла печатается номер итерации, это же число возводится в квадрат и из него извлекается квадратный корень
        cout << "Итерация № " << i << ":\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
    }

    cout << "------------------------------\n\tЦикл завершен.\n";//этот код НЕ относится к циклу for(;;){}
    system("pause");
    return 0;
}
```

Результат работы программы:

```
D:\2019\Labs\x64\Debug\Lab3_0.exe
Итерация № i    i^2     sqrt(i)
------------------------------
Итерация № 15:  225     3.87298
Итерация № 14:  196     3.74166
Итерация № 13:  169     3.60555
Итерация № 12:  144     3.4641
Итерация № 11:  121     3.31662
Итерация № 10:  100     3.16228
Итерация № 9:   81      3
Итерация № 8:   64      2.82843
Итерация № 7:   49      2.64575
Итерация № 6:   36      2.44949
Итерация № 5:   25      2.23607
Итерация № 4:   16      2
Итерация № 3:   9       1.73205
Итерация № 2:   4       1.41421
Итерация № 1:   1       1
Итерация № 0:   0       0
------------------------------
        Цикл завершен.
Для продолжения нажмите любую клавишу . . .
```

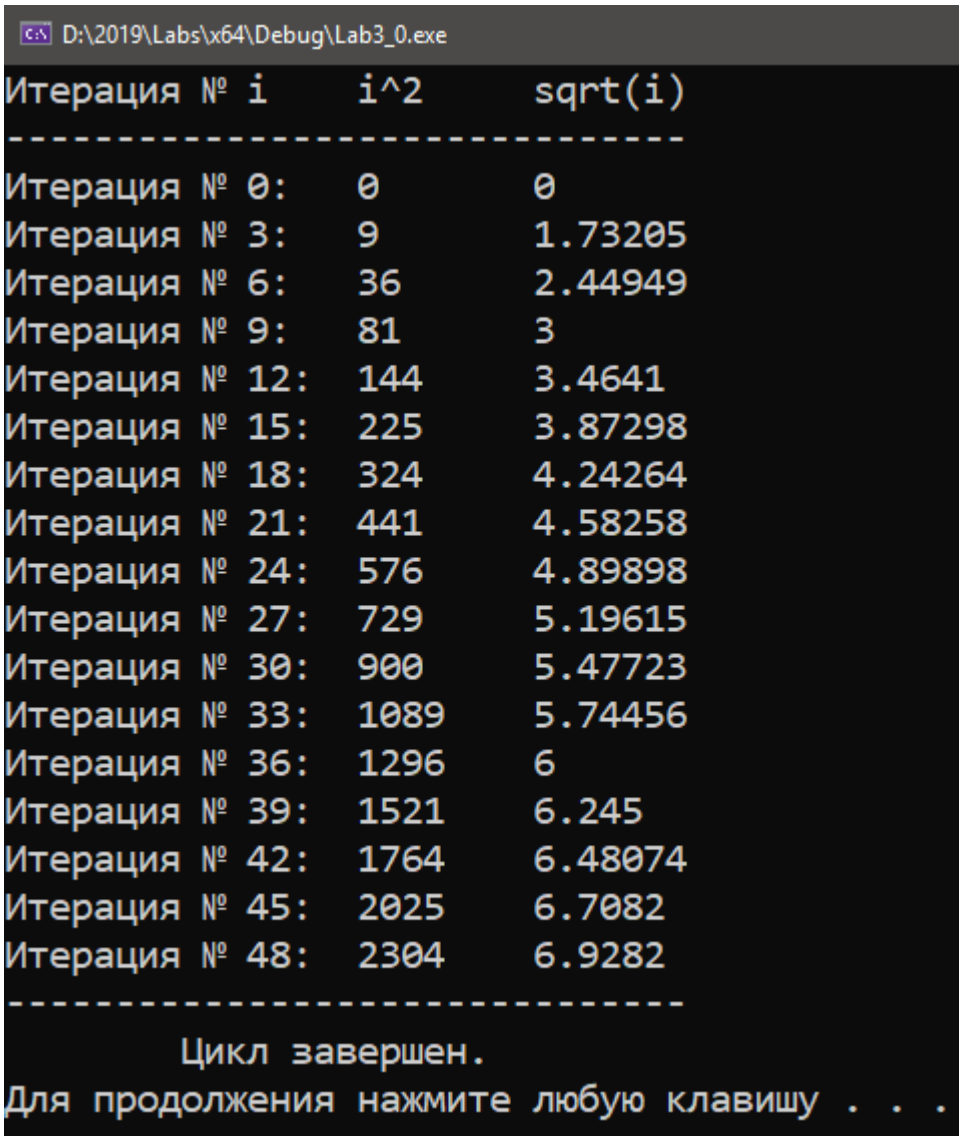
Значение итерационной переменной можно изменять с различным шагом, например, увеличивать с нуля с шагом +3.

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n------------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    //создадим цикл с итерационной переменной, изменяющейся от 0 до 50 с шагом +3
    for (int i = 0; i < 50; i = i + 3)//счетчик цикла увеличивается на 3. В последней итерации счетчик будет равен 48
    {//в теле цикла печатается номер итерации, это же число возводится в квадрат и из него извлекается квадратный корень
        cout << "Итерация № " << i << ":\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
    }//когда счетчик станет равным 51 проверка условия даст false - цикл завершится

    cout << "------------------------------\n\tЦикл завершен.\n";//этот код НЕ относится к циклу for(;;){}
    system("pause");
    return 0;
}
```

Результат работы программы:



Оператор цикла while-do

Оператор цикла с предусловием. Общий вид оператора:

while(*условие_выполнения_цикла*)
{
тело цикла//выражения одной итерации цикла
}

Организует повторение операторов *тела цикла* до тех пор, пока проверка *условия_выполнения_цикла* ИСТИННО (возвращает ответ «да, верно»). Часто в *условии_выполнения_цикла* используются переменные, которые должны быть созданы (продекларированы) и проинициализированы **заранее** – до начала цикла. Почему? В случае, если первая проверка условия выполнения цикла while-do даст ответ «нет, не верно» (ЛОЖНО, false), то тело цикла не выполнится ни разу и управление сразу передастся коду, следующему за этим циклом (за телом этого цикла).

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n-----------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    int i = 0;//итерационную переменную надо создать ДО цикла, заранее
    while (i < 10)//проверка пишется перед телом цикла - это цикл с предусловием//если первая проверка цикла с предусловием
    {//даст результат ЛОЖНО ("не верно", false), то ниженаписанное тело цикла не выполнится ни разу
        cout << "Итерация № " << i << ":\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
        i++;//изменять счетчик цикла надо В ТЕЛЕ цикла while-do (обычно в конце тела цикла, то есть в конце итерации цикла)
    }

    cout << "-----------------------------\n\tЦикл while-do завершен.\n";//этот код НЕ относится к циклу while()do{}
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab3_1.exe
Итерация № i    i^2     sqrt(i)
-----------------------------
Итерация № 0:   0       0
Итерация № 1:   1       1
Итерация № 2:   4       1.41421
Итерация № 3:   9       1.73205
Итерация № 4:   16      2
Итерация № 5:   25      2.23607
Итерация № 6:   36      2.44949
Итерация № 7:   49      2.64575
Итерация № 8:   64      2.82843
Итерация № 9:   81      3
-----------------------------
        Цикл while-do завершен.
Для продолжения нажмите любую клавишу . . .
```

Оператор цикла do-while

Оператор цикла с постусловием. Общий вид оператора:

do
{
тело цикла//выражения одной итерации цикла
}
while(*условие_продолжения_цикла*)**;**//обязательно заканчивается знаком «ТОЧКА //С ЗАПЯТОЙ» (;)

Организует повторение операторов *тела цикла* до тех пор, пока *условие_продолжения_цикла* истинно. Часто в *условии_продолжения_цикла* используются переменные, которые должны быть созданы (продекларированы) и прои-

нициализированы **заранее** – до начала цикла. Эти переменные могут быть проинициализированы и в теле цикла do-while. Почему?

В случае, если первая проверка *условия_продолжения_цикла* do-while даст ответ «нет, не верно» (ЛОЖНО, false), то тело цикла **все равно уже выполнилось один раз**, поскольку тело цикла находится **перед** проверкой *условия_продолжения_цикла*. Только после выполнения первой итерации, управление передастся на проверку *условия_продолжения_цикла*, которая может дать результат ЛОЖНО («нет, не верно», false), и это завершит данный цикл, то есть не выполнится **следующая вторая** итерация цикла (но первая итерация все равно **уже успела** выполниться).

```
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n-----------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    int i = 0;//итерационную переменную надо создать ДО цикла, заранее
    do//это цикл с постусловием: сначала выполнится первая итерация (строки с 13 по 16), только потом будет выполнена проверка в строке 17
    {
        cout << "Итерация № " << i << ":\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
        i++;//изменять счетчик цикла надо В ТЕЛЕ цикла do-while (обычно в конце тела цикла, то есть в конце итерации цикла)
    }
    while (i < 10);//проверка после тела итерации//нужно писать ТОЧКУ С ЗАПЯТОЙ в конце цикла do-while

    cout << "-----------------------------\n\tЦикл do-while завершен.\n";//этот код НЕ относится к циклу do{}while();
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab3_2.exe
Итерация № i    i^2     sqrt(i)
-----------------------------
Итерация № 0:   0       0
Итерация № 1:   1       1
Итерация № 2:   4       1.41421
Итерация № 3:   9       1.73205
Итерация № 4:   16      2
Итерация № 5:   25      2.23607
Итерация № 6:   36      2.44949
Итерация № 7:   49      2.64575
Итерация № 8:   64      2.82843
Итерация № 9:   81      3
-----------------------------
        Цикл do-while завершен.
Для продолжения нажмите любую клавишу . . .
```

Проверим работу цикла do-while с изначально неподходящим значением счетчика:

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n------------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    int i = 56;//итерационную переменную надо создать ДО цикла, заранее//сделаем ее значение изначально неподходящим: 56 НЕ меньше 10
    do//это цикл с постусловием: сначала выполнится первая итерация (строки с 13 по 16), только потом будет выполнена проверка в строке 17
    {
        cout << "Итерация № " << i << ":\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
        i++;//изменять счетчик цикла надо В ТЕЛЕ цикла do-while (обычно в конце тела цикла, то есть в конце итерации цикла)
    }
    while (i < 10);//проверка после тела итерации//нужно писать ТОЧКУ С ЗАПЯТОЙ в конце цикла do-while

    cout << "------------------------------\n\tЦикл do-while завершен.\n";//этот код НЕ относится к циклу do{}while();
    system("pause");
    return 0;
}
```

Но первая итерация все равно выполнилась, поскольку do-while – это цикл с постусловием (постпроверкой) – проверка возможности продолжения цикла выполняется **после выполнения первой итерации**:

```
D:\2019\Labs\x64\Debug\Lab3_2.exe
Итерация № i    i^2     sqrt(i)
--------------------------------
Итерация № 56:  3136    7.48331
--------------------------------
        Цикл do-while завершен.
Для продолжения нажмите любую клавишу . . .
```

Операторы break и continue

Во всех трех видах циклов могут использоваться операторы break и continue. Они используются в теле циклов внутри проверок if(){} и/или if(){}else{}, поскольку иначе они будут срабатывать безусловно (всегда). Примеры кода:

```
if(x < 5)
{
        break;
}
////////////////////////////////
if(f >=9 & s != 78 | g%5==0)
{
        continue;
}
```

Когда срабатывает оператор break, то он сразу прекращает (завершает) данный цикл и передает управление выражению, которое следует за данным циклом. Оператор break полностью прекращает цикл, в котором он срабатывает. Если в цикле находится цикл, а во вложенном цикле срабатывает break, то он прекратит вложенный цикл, но не внешний. Чтобы прекратить внешний цикл, нужен оператор break во внешнем цикле.

Когда срабатывает оператор continue, то он прекращает выполнение текущей итерации данного цикла и передает управление на начало следующей итерации данного цикла. Если в цикле находится цикл, а во вложенном цикле срабатывает continue, то он прекратит текущую итерацию вложенного цикла, **но не** итерацию **внешнего** цикла. Чтобы прекратить текущую итерацию внешнего цикла, нужен оператор continue во внешнем цикле.
В цикле for(;;) оператор continue прекращает текущую итерацию цикла и передает управление на выражение *коррекции*, за которым следует проверка *условия_выполнения_цикла*, которая решает, будет ли выполняться следующая итерация данного цикла for или нет.
В цикле while-do оператор continue прекращает текущую итерацию цикла и передает управление выражению проверки *условия_выполнения_цикла*, которое решает, будет ли выполняться следующая итерация данного цикла while-do или нет.
В цикле do-while оператор continue прекращает текущую итерацию цикла и передает управление выражению проверки *условия_продолжения_цикла*, которое решает, будет ли выполняться следующая итерация данного цикла do-while или нет.

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Итерация № i\ti^2\tsqrt(i)\n------------------------------\n";//печать заголовков столбцов таблицы//"шапка"
    for(int i = 0; i < 50; i++)//создать цикл for на 50 итераций от 0 до 49 включительно
    {
        cout << "Итерация № " << i << " началась:\t" << pow(i, 2) << '\t' << sqrt(i) << endl;
        if (i == 5)
        {
            cout << "Сработал continue.\n";
            continue;
        }
        if (i == 10)
        {
            cout << "Сработал break.\n";
            break;
        }
        cout << "Итерация № " << i << " завершена.\n";//выражение в конце итерации
    }
    cout << "------------------------------\n\tЦикл for завершен.\n";//этот код НЕ относится к циклу for
    system("pause");
    return 0;
}
```

Протестируем программу и посмотрим на результаты работы continue и break в цикле на 50 итераций:

```
D:\2019\Labs\x64\Debug\Lab3_3.exe
Итерация № i    i^2     sqrt(i)
------------------------------
Итерация № 0 началась:  0       0
Итерация № 0 завершена.
Итерация № 1 началась:  1       1
Итерация № 1 завершена.
Итерация № 2 началась:  4       1.41421
Итерация № 2 завершена.
Итерация № 3 началась:  9       1.73205
Итерация № 3 завершена.
Итерация № 4 началась:  16      2
Итерация № 4 завершена.
Итерация № 5 началась:  25      2.23607
Сработал continue.
Итерация № 6 началась:  36      2.44949
Итерация № 6 завершена.
Итерация № 7 началась:  49      2.64575
Итерация № 7 завершена.
Итерация № 8 началась:  64      2.82843
Итерация № 8 завершена.
Итерация № 9 началась:  81      3
Итерация № 9 завершена.
Итерация № 10 началась: 100     3.16228
Сработал break.
------------------------------
        Цикл for завершен.
Для продолжения нажмите любую клавишу . . .
```

Вычисление суммы ряда.

Рассмотрим вычисление значения функции Ch x (гиперболический косинус) с помощью бесконечного ряда Тейлора с точностью **e** (которую вводит пользователь с клавиатуры; точность лучше именовать **ε** «эпсилон» epsilon или eps) по формуле:

$$y = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \cdots + \frac{x^{2n}}{2n!} + \cdots$$

. Данную формулу нагляднее расписать так:

$$y = \frac{x^0}{0!} + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!} + \cdots + \frac{x^{(2n)}}{(2n)!} + \frac{x^{(2n+2)}}{(2n+2)!} + \cdots$$

Этот ряд сходится при |x| < ∞ (+ бесконечности). Для достижения заданной точности **e** требуется суммировать члены ряда, абсолютная величина которых больше **e**.

Для сходящегося ряда модуль члена ряда C_n при увеличении n стремится к нулю. При некотором n неравенство $|C_n| >= e$ перестает выполняться, и вычисления прекращаются.

Общий алгоритм решения этой задачи: требуется задать начальное значение суммы ряда, а затем многократно вычислять очередной член ряда и прибавлять его к ранее найденной сумме. Вычисления заканчиваются, когда абсолютная (по модулю) величина очередного члена ряда станет меньше заданной точности **e**.

Прямое вычисление члена ряда по приведенной выше общей формуле, когда **x** возводится в степень, вычисляется факториал, а затем числитель делится на знаменатель, имеет два недостатка, которые делают этот способ неоптимальным.

Первый недостаток – большая погрешность вычислений. При возведении в степень и вычислении факториала можно получить очень большие числа, при делении которых друг на друга произойдет потеря точности, поскольку количество значащих цифр, хранимых в ячейке памяти переменной, ограничено.

Второй недостаток связан с эффективностью вычислений: можно заметить, что при вычислении очередного члена ряда нам уже известно значение предыдущего члена, поэтому вычислять каждый следующий член ряда по формуле с самого начала нерационально, ведь очередной член ряда связан с предыдущим, а потому можно взять предыдущий вычисленный член ряда и умножить его на коэффициент, получив значение очередного члена ряда и так далее. Но сначала нужно вычислить этот коэффициент.

Для уменьшения количества выполняемых действий следует воспользоваться рекуррентной формулой получения последующего члена ряда через предыдущий $C_{n+1} = C_n * T$, где T – некоторый множитель. Подставив в эту формулу C_n и C_{n+1}, получим выражение для вычисления T:

$$T = \frac{C_{n+1}}{C_n} = \frac{(2n)! \cdot x^{2(n+1)}}{(2(n+1))! \cdot x^{2n}} = \frac{x^2}{(2n+1)(2n+2)}.$$

То есть нужно последующий член последовательности разделить на предыдущий (тот, который идет по порядку перед ним) член последовательности, записав их в общем виде, то есть делим $C_{(n+1)}$ член на C_n член. Вышеприведенную формулу по нахождению T нагляднее расписать так:

$$T = \frac{C_{n+1}}{C_n} = \frac{x^{(2n+2)}}{(2n+2)!} : \frac{x^{(2n)}}{(2n)!} = \frac{x^{(2n+2)} * (2n)!}{(2n+2)! * x^{(2n)}} = \frac{x^{(2n)} * x^2 * (2n)!}{(2n+2)! * x^{(2n)}}$$
$$= \frac{x^{(2n)} * x^2 * (2n)!}{x^{(2n)} * (2n+2)!} = \frac{x^2 * (2n)!}{(2n+2)!} = \frac{x^2}{(2n+1) * (2n+2)}$$

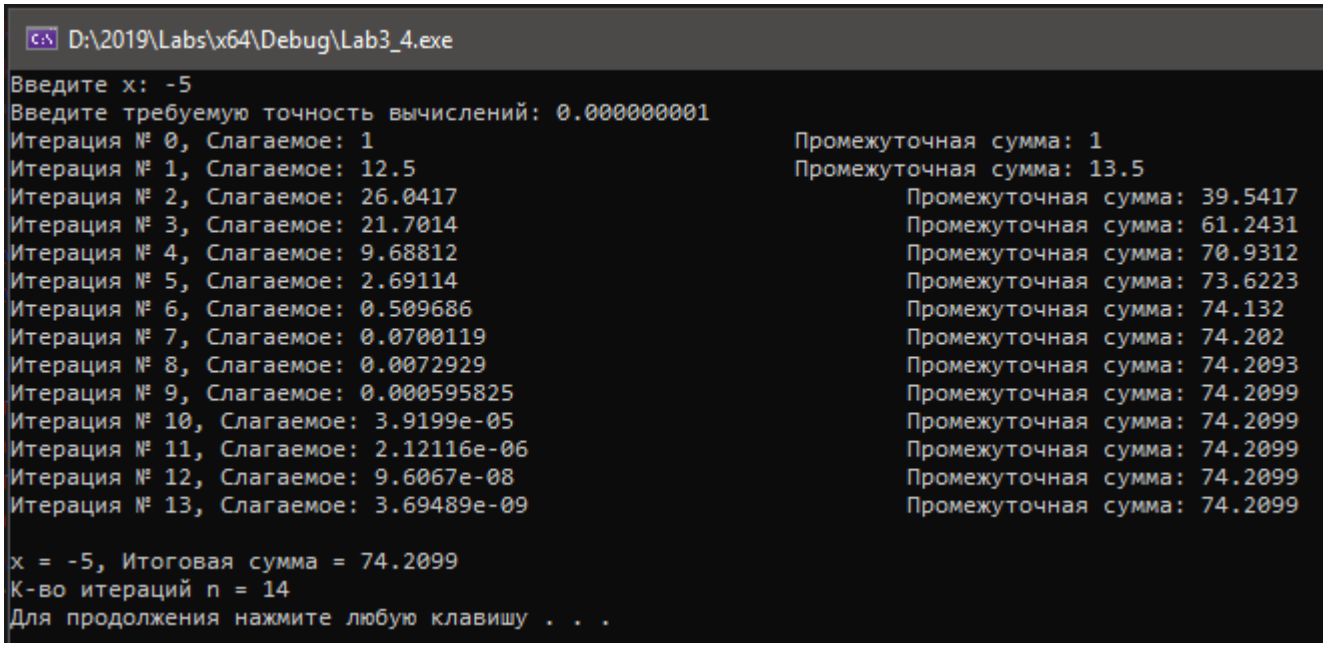
Если данную задачу решать в программе, вычисляя каждый очередной член последовательности с начала, то потребуется больше циклов, но после такой изначальной оптимизации, основанной на рекуррентной связи последующего члена ряда с предыдущим, заранее вычислив коэффициент T, можно написать оптимизированную программу, сводящуюся к одному циклу:

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    unsigned long long int n = 0;
    long double x, eps;
    cout << "Введите x: ";
    cin >> x;
    cout << "Введите требуемую точность вычислений: ";
    cin >> eps;
    long double ch = 1, y = ch;//первый член ряда и начальное значение суммы
    for (; fabs(ch) > eps; n++)//итерационная переменная создана и проинициализирована заранее, поэтому выражение декларации "пустое"
    {//цикл выполняется, пока очередное слагаемое (член ряда) не станет по модулю меньше требуемой пользователем точности
        cout << "Итерация № " << n << ", Слагаемое: " << ch << "\t\t\t\tПромежуточная сумма: " << y << endl;
        ch = ch * pow(x, 2) / ((2 * n + 1) * (2 * n + 2));//очередной член ряда (промежуточное слагаемое)// Cn+1 = Cn * T
        y = y + ch;//добавление очередного рассчитанного члена ряда к "итоговой" сумме
    }
    cout << endl << "x = " << x << ", Итоговая сумма = " << y << endl;
    cout << "К-во итераций n = " << n << endl;
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab3_4.exe
Введите x: 20
Введите требуемую точность вычислений: 0.00000001
Итерация № 0, Слагаемое: 1                              Промежуточная сумма: 1
Итерация № 1, Слагаемое: 200                            Промежуточная сумма: 201
Итерация № 2, Слагаемое: 6666.67                                Промежуточная сумма: 6867.67
Итерация № 3, Слагаемое: 88888.9                                Промежуточная сумма: 95756.6
Итерация № 4, Слагаемое: 634921                         Промежуточная сумма: 730677
Итерация № 5, Слагаемое: 2.82187e+06                            Промежуточная сумма: 3.55255e+06
Итерация № 6, Слагаемое: 8.55112e+06                            Промежуточная сумма: 1.21037e+07
Итерация № 7, Слагаемое: 1.87937e+07                            Промежуточная сумма: 3.08973e+07
Итерация № 8, Слагаемое: 3.13228e+07                            Промежуточная сумма: 6.22201e+07
Итерация № 9, Слагаемое: 4.09448e+07                            Промежуточная сумма: 1.03165e+08
Итерация № 10, Слагаемое: 4.30998e+07                           Промежуточная сумма: 1.46265e+08
Итерация № 11, Слагаемое: 3.73158e+07                           Промежуточная сумма: 1.83581e+08
Итерация № 12, Слагаемое: 2.70405e+07                           Промежуточная сумма: 2.10621e+08
Итерация № 13, Слагаемое: 1.66403e+07                           Промежуточная сумма: 2.27261e+08
Итерация № 14, Слагаемое: 8.80439e+06                           Промежуточная сумма: 2.36066e+08
Итерация № 15, Слагаемое: 4.04799e+06                           Промежуточная сумма: 2.40114e+08
Итерация № 16, Слагаемое: 1.63226e+06                           Промежуточная сумма: 2.41746e+08
Итерация № 17, Слагаемое: 581909                                Промежуточная сумма: 2.42328e+08
Итерация № 18, Слагаемое: 184733                                Промежуточная сумма: 2.42513e+08
Итерация № 19, Слагаемое: 52555.6                               Промежуточная сумма: 2.42565e+08
Итерация № 20, Слагаемое: 13475.8                               Промежуточная сумма: 2.42579e+08
Итерация № 21, Слагаемое: 3130.27                               Промежуточная сумма: 2.42582e+08
Итерация № 22, Слагаемое: 661.79                                Промежуточная сумма: 2.42582e+08
Итерация № 23, Слагаемое: 127.882                               Промежуточная сумма: 2.42583e+08
Итерация № 24, Слагаемое: 22.6741                               Промежуточная сумма: 2.42583e+08
Итерация № 25, Слагаемое: 3.7019                                Промежуточная сумма: 2.42583e+08
Итерация № 26, Слагаемое: 0.558356                              Промежуточная сумма: 2.42583e+08
Итерация № 27, Слагаемое: 0.0780372                             Промежуточная сумма: 2.42583e+08
Итерация № 28, Слагаемое: 0.0101347                             Промежуточная сумма: 2.42583e+08
Итерация № 29, Слагаемое: 0.00122622                            Промежуточная сумма: 2.42583e+08
Итерация № 30, Слагаемое: 0.000138556                           Промежуточная сумма: 2.42583e+08
Итерация № 31, Слагаемое: 1.46542e-05                           Промежуточная сумма: 2.42583e+08
Итерация № 32, Слагаемое: 1.45379e-06                           Промежуточная сумма: 2.42583e+08
Итерация № 33, Слагаемое: 1.35552e-07                           Промежуточная сумма: 2.42583e+08
Итерация № 34, Слагаемое: 1.1901e-08                            Промежуточная сумма: 2.42583e+08

x = 20, Итоговая сумма = 2.42583e+08
К-во итераций n = 35
Для продолжения нажмите любую клавишу . . .
```


D:\2019\Labs\x64\Debug\Lab3_4.exe
Введите x: -5
Введите требуемую точность вычислений: 0.000000001
Итерация № 0, Слагаемое: 1 Промежуточная сумма: 1
Итерация № 1, Слагаемое: 12.5 Промежуточная сумма: 13.5
Итерация № 2, Слагаемое: 26.0417 Промежуточная сумма: 39.5417
Итерация № 3, Слагаемое: 21.7014 Промежуточная сумма: 61.2431
Итерация № 4, Слагаемое: 9.68812 Промежуточная сумма: 70.9312
Итерация № 5, Слагаемое: 2.69114 Промежуточная сумма: 73.6223
Итерация № 6, Слагаемое: 0.509686 Промежуточная сумма: 74.132
Итерация № 7, Слагаемое: 0.0700119 Промежуточная сумма: 74.202
Итерация № 8, Слагаемое: 0.0072929 Промежуточная сумма: 74.2093
Итерация № 9, Слагаемое: 0.000595825 Промежуточная сумма: 74.2099
Итерация № 10, Слагаемое: 3.9199e-05 Промежуточная сумма: 74.2099
Итерация № 11, Слагаемое: 2.12116e-06 Промежуточная сумма: 74.2099
Итерация № 12, Слагаемое: 9.6067e-08 Промежуточная сумма: 74.2099
Итерация № 13, Слагаемое: 3.69489e-09 Промежуточная сумма: 74.2099
x = -5, Итоговая сумма = 74.2099
К-во итераций n = 14
Для продолжения нажмите любую клавишу . . .

```
D:\2019\Labs\x64\Debug\Lab3_4.exe
Введите x: 30
Введите требуемую точность вычислений: 0.0000000001
Итерация № 0, Слагаемое: 1                              Промежуточная сумма: 1
Итерация № 1, Слагаемое: 450                            Промежуточная сумма: 451
Итерация № 2, Слагаемое: 33750                          Промежуточная сумма: 34201
Итерация № 3, Слагаемое: 1.0125e+06                             Промежуточная сумма: 1.0467e+06
Итерация № 4, Слагаемое: 1.62723e+07                            Промежуточная сумма: 1.7319e+07
Итерация № 5, Слагаемое: 1.62723e+08                            Промежуточная сумма: 1.80042e+08
Итерация № 6, Слагаемое: 1.10948e+09                            Промежуточная сумма: 1.28952e+09
Итерация № 7, Слагаемое: 5.48642e+09                            Промежуточная сумма: 6.77594e+09
Итерация № 8, Слагаемое: 2.05741e+10                            Промежуточная сумма: 2.735e+10
Итерация № 9, Слагаемое: 6.0512e+10                             Промежуточная сумма: 8.7862e+10
Итерация № 10, Слагаемое: 1.43318e+11                           Промежуточная сумма: 2.3118e+11
Итерация № 11, Слагаемое: 2.79191e+11                           Промежуточная сумма: 5.10371e+11
Итерация № 12, Слагаемое: 4.55202e+11                           Промежуточная сумма: 9.65573e+11
Итерация № 13, Слагаемое: 6.3028e+11                            Промежуточная сумма: 1.59585e+12
Итерация № 14, Слагаемое: 7.50333e+11                           Промежуточная сумма: 2.34619e+12
Итерация № 15, Слагаемое: 7.76207e+11                           Промежуточная сумма: 3.12239e+12
Итерация № 16, Слагаемое: 7.0422e+11                            Промежуточная сумма: 3.82661e+12
Итерация № 17, Слагаемое: 5.64882e+11                           Промежуточная сумма: 4.3915e+12
Итерация № 18, Слагаемое: 4.03487e+11                           Промежуточная сумма: 4.79498e+12
Итерация № 19, Слагаемое: 2.58278e+11                           Промежуточная сумма: 5.05326e+12
Итерация № 20, Слагаемое: 1.49006e+11                           Промежуточная сумма: 5.20227e+12
Итерация № 21, Слагаемое: 7.78779e+10                           Промежуточная сумма: 5.28015e+12
Итерация № 22, Слагаемое: 3.70455e+10                           Промежуточная сумма: 5.31719e+12
Итерация № 23, Слагаемое: 1.61068e+10                           Промежуточная сумма: 5.3333e+12
Итерация № 24, Слагаемое: 6.42557e+09                           Промежуточная сумма: 5.33972e+12
Итерация № 25, Слагаемое: 2.36041e+09                           Промежуточная сумма: 5.34208e+12
Итерация № 26, Слагаемое: 8.01045e+08                           Промежуточная сумма: 5.34289e+12
Итерация № 27, Слагаемое: 2.51901e+08                           Промежуточная сумма: 5.34314e+12
Итерация № 28, Слагаемое: 7.36074e+07                           Промежуточная сумма: 5.34321e+12
Итерация № 29, Слагаемое: 2.00383e+07                           Промежуточная сумма: 5.34323e+12
Итерация № 30, Слагаемое: 5.09449e+06                           Промежуточная сумма: 5.34324e+12
Итерация № 31, Слагаемое: 1.21233e+06                           Промежуточная сумма: 5.34324e+12
Итерация № 32, Слагаемое: 270610                                Промежуточная сумма: 5.34324e+12
Итерация № 33, Слагаемое: 56771.3                               Промежуточная сумма: 5.34324e+12
Итерация № 34, Слагаемое: 11214.7                               Промежуточная сумма: 5.34324e+12
Итерация № 35, Слагаемое: 2089.69                               Промежуточная сумма: 5.34324e+12
Итерация № 36, Слагаемое: 367.904                               Промежуточная сумма: 5.34324e+12
Итерация № 37, Слагаемое: 61.2946                               Промежуточная сумма: 5.34324e+12
Итерация № 38, Слагаемое: 9.6781                                Промежуточная сумма: 5.34324e+12
Итерация № 39, Слагаемое: 1.45026                               Промежуточная сумма: 5.34324e+12
Итерация № 40, Слагаемое: 0.206525                              Промежуточная сумма: 5.34324e+12
Итерация № 41, Слагаемое: 0.0279844                             Промежуточная сумма: 5.34324e+12
Итерация № 42, Слагаемое: 0.00361245                            Промежуточная сумма: 5.34324e+12
Итерация № 43, Слагаемое: 0.000444761                           Промежуточная сумма: 5.34324e+12
Итерация № 44, Слагаемое: 5.22838e-05                           Промежуточная сумма: 5.34324e+12
Итерация № 45, Слагаемое: 5.87458e-06                           Промежуточная сумма: 5.34324e+12
Итерация № 46, Слагаемое: 6.31525e-07                           Промежуточная сумма: 5.34324e+12
Итерация № 47, Слагаемое: 6.50163e-08                           Промежуточная сумма: 5.34324e+12
Итерация № 48, Слагаемое: 6.41608e-09                           Промежуточная сумма: 5.34324e+12
Итерация № 49, Слагаемое: 6.07456e-10                           Промежуточная сумма: 5.34324e+12

x = 30, Итоговая сумма = 5.34324e+12
К-во итераций n = 50
Для продолжения нажмите любую клавишу . . .
```

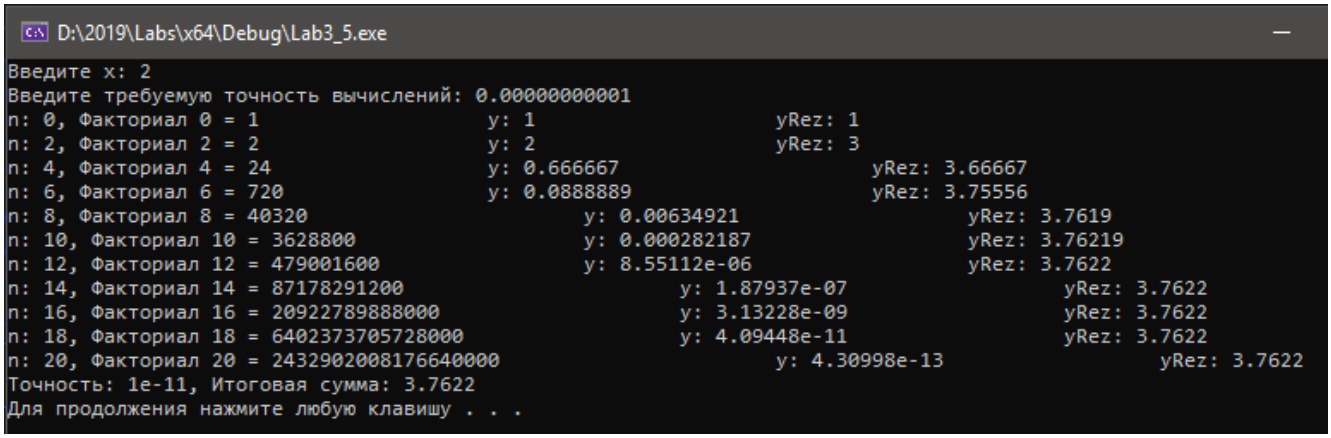
Аналог вышеизложенной программы, который без оптимизации решает поставленную задачу показан ниже. Тут вычисляется факториал – быстрорастущая функция, которая даже с использованием типа данных unsigned long long int для хранения результата после вычисления факториала 20 вскоре дает переполнение (не хватает размера переменной для сохранения результата), поэтому, если допустить более 20 итераций, у нас будут неверные данные. Соответственно, следует

вводить маленькие значения переменной x, тогда можно получить расчет результата с приемлемой точностью. Эта программа без оптимизации «честно» считает каждый член ряда, вычисляет факториал, не используя возможности сокращения числителя и знаменателя дроби на одинаковые значения.

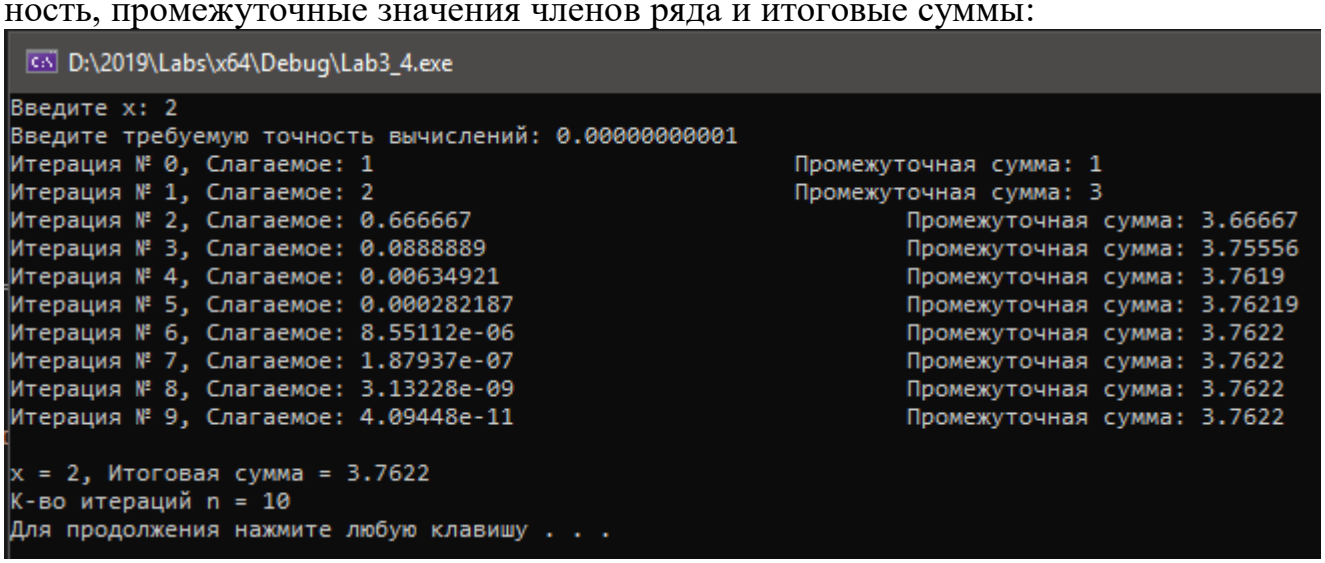
```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    unsigned long long int n = 0;
    long double x, eps;
    cout << "Введите x: ";
    cin >> x;
    cout << "Введите требуемую точность вычислений: ";
    cin >> eps;
    long double y = 1, yRez = 0;//первый член ряда и начальное значение суммы
    while (fabs(y) > eps)
    {
        cout << "n: " << n;
        unsigned long long int f = 1;//для расчета факториала в каждой итерации внешнего цикла
        for (int i = 1; i < n + 1; i++)
        {
            f = f * i;
        }
        cout << ", Факториал " << n << " = " << f;
        y = pow(x, n) / f;//вычисление очередного члена ряда
        yRez = yRez + y;//увеличение итоговой суммы на значение очередного члена ряда (слагаемого)
        cout << "\t\t\ty: " << y << "\t\t\tyRez: " << yRez << endl;
        n = n + 2;//степень икса возрастает с шагом +2
    }
    cout << "Точность: " << eps << ", Итоговая сумма: " << yRez << endl;
    system("pause");
    return 0;
}
```

При маленьком числителе можно получить результат с высокой точностью. Даже при сохранении результата в переменную типа unsigned long long int можно рассчитать точное значение факториала только для чисел от 0 до 20 включительно. Если рассчитать факториал для бОльших значений и сравнить полученные результаты с эталонными (в энциклопедиях, справочниках, специализированных сайтах), то вы увидите, что последний рассчитанный факториал, совпадающий с эталонным значением, является факториалом числа 20. Уже рассчитанный факториал для числа 21 выглядит достоверно, не превышает диапазон unsigned long long int, но меньше эталонного значения, которое на самом деле превышает диапазон unsigned long long int. Какое максимальное значение можно поместить в переменную типа unsigned long long int? Каков на самом деле факториал числа 21?



Сравним с расчетами, полученными на тех же данных в оптимизированной программе. На иллюстрации выше и ниже сравните требуемую пользователем точность, промежуточные значения членов ряда и итоговые суммы:



Поскольку гиперболический косинус также вычисляется по формуле:

$$Ch\,x = \frac{e^{x} + e^{-x}}{2}$$

, то ее также можно применить для расчета итогового результата и сравнить два решения – рассчитанное программой в цикле путем сложения членов и рассчитанное по последней формуле с экспонентами. Обе программы можно дополнить кодом в конце для получения и сравнения двух рассчитанных результатов. Модифицируем вторую программу:

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    unsigned long long int n = 0;
    long double x, eps;
    cout << "Введите x: ";
    cin >> x;
    cout << "Введите требуемую точность вычислений: ";
    cin >> eps;
    long double y = 1, yRez = 0;//первый член ряда и начальное значение суммы
    while (fabs(y) > eps)
    {
        cout << "n: " << n;
        unsigned long long int f = 1;//для расчета факториала в каждой итерации внешнего цикла
        for (int i = 1; i < n + 1; i++)
        {
            f = f * i;
        }
        cout << ", Факториал " << n << " = " << f;
        y = pow(x, n) / f;//вычисление очередного члена ряда
        yRez = yRez + y;//увеличение итоговой суммы на значение очередного члена ряда (слагаемого)
        cout << "\t\t\ty: " << y << "\t\t\tyRez: " << yRez << endl;
        n = n + 2;//степень икса возрастает с шагом +2
    }
    long double yRez2 = (exp(x) + exp(-x)) / 2;
    cout << "Точность: " << eps << ", Итоговая сумма: " << yRez << ", Расчет по формуле2: " << yRez2 << endl;
    system("pause");
    return 0;
}
```

Протестируем на подходящих данных:

```
D:\2019\Labs\x64\Debug\Lab3_5.exe
Введите x: 2
Введите требуемую точность вычислений: 0.0000000001
n: 0, Факториал 0 = 1                   y: 1                    yRez: 1
n: 2, Факториал 2 = 2                   y: 2                    yRez: 3
n: 4, Факториал 4 = 24                  y: 0.666667                     yRez: 3.66667
n: 6, Факториал 6 = 720                 y: 0.0888889                    yRez: 3.75556
n: 8, Факториал 8 = 40320                       y: 0.00634921                   yRez: 3.7619
n: 10, Факториал 10 = 3628800                   y: 0.000282187                  yRez: 3.76219
n: 12, Факториал 12 = 479001600                 y: 8.55112e-06                  yRez: 3.7622
n: 14, Факториал 14 = 87178291200                       y: 1.87937e-07                  yRez: 3.7622
n: 16, Факториал 16 = 20922789888000                    y: 3.13228e-09                  yRez: 3.7622
n: 18, Факториал 18 = 6402373705728000                  y: 4.09448e-11                  yRez: 3.7622
Точность: 1e-10, Итоговая сумма: 3.7622, Расчет по формуле2: 3.7622
Для продолжения нажмите любую клавишу . . .
```

Модифицируем первую программу:

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    unsigned long long int n = 0;
    long double x, eps;
    cout << "Введите x: ";
    cin >> x;
    cout << "Введите требуемую точность вычислений: ";
    cin >> eps;
    long double ch = 1, y = ch;//первый член ряда и начальное значение суммы
    for (; fabs(ch) > eps; n++)//итерационная переменная создана и проинициализирована заранее, поэтому выражение декларации "пустое"
    {//цикл выполняется, пока очередное слагаемое (член ряда) не станет по модулю меньше требуемой пользователем точности
        cout << "Итерация № " << n << ", Слагаемое: " << ch << "\t\t\t\tПромежуточная сумма: " << y << endl;
        ch = ch * pow(x, 2) / ((2 * n + 1) * (2 * n + 2));//очередной член ряда (промежуточное слагаемое)// Cn+1 = Cn * T
        y = y + ch;//добавление очередного рассчитанного члена ряда к "итоговой" сумме
    }
    long double y2 = (exp(x) + exp(-x)) / 2;//расчет по другой формуле
    cout << endl << "x: " << x << ", Точность: " << eps << ", Итоговая сумма: " << y << ", Сумма2: " << y2 << endl;
    cout << "К-во итераций n = " << n << endl;
    system("pause");
    return 0;
}
```

Протестируем на наборе данных:

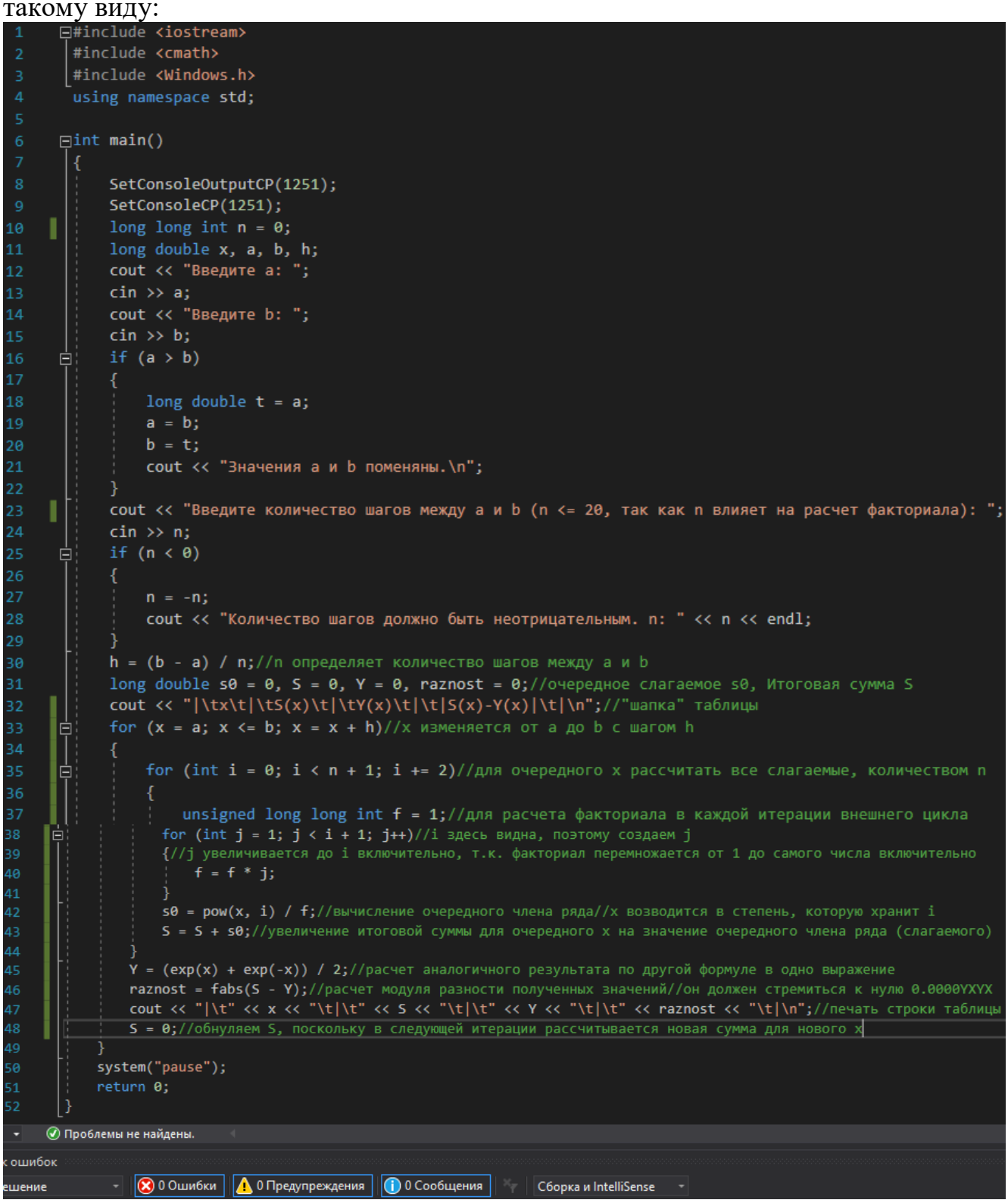
```
D:\2019\Labs\x64\Debug\Lab3_4.exe
Введите x: 31
Введите требуемую точность вычислений: 0.0000000001
Итерация № 0, Слагаемое: 1                                  Промежуточная сумма: 1
Итерация № 1, Слагаемое: 480.5                              Промежуточная сумма: 481.5
Итерация № 2, Слагаемое: 38480                              Промежуточная сумма: 38961.5
Итерация № 3, Слагаемое: 1.23264e+06                                Промежуточная сумма: 1.27161e+06
Итерация № 4, Слагаемое: 2.11531e+07                                Промежуточная сумма: 2.24247e+07
Итерация № 5, Слагаемое: 2.25868e+08                                Промежуточная сумма: 2.48292e+08
Итерация № 6, Слагаемое: 1.64438e+09                                Промежуточная сумма: 1.89268e+09
Итерация № 7, Слагаемое: 8.68271e+09                                Промежуточная сумма: 1.05754e+10
Итерация № 8, Слагаемое: 3.4767e+10                                 Промежуточная сумма: 4.53424e+10
Итерация № 9, Слагаемое: 1.09187e+11                                Промежуточная сумма: 1.54529e+11
Итерация № 10, Слагаемое: 2.76127e+11                               Промежуточная сумма: 4.30656e+11
Итерация № 11, Слагаемое: 5.74369e+11                               Промежуточная сумма: 1.00502e+12
Итерация № 12, Слагаемое: 9.99942e+11                               Промежуточная сумма: 2.00497e+12
Итерация № 13, Слагаемое: 1.47838e+12                               Промежуточная сумма: 3.48334e+12
Итерация № 14, Слагаемое: 1.87926e+12                               Промежуточная сумма: 5.3626e+12
Итерация № 15, Слагаемое: 2.07583e+12                               Промежуточная сумма: 7.43843e+12
Итерация № 16, Слагаемое: 2.01096e+12                               Промежуточная сумма: 9.44938e+12
Итерация № 17, Слагаемое: 1.7224e+12                                Промежуточная сумма: 1.11718e+13
Итерация № 18, Слагаемое: 1.31367e+12                               Промежуточная сумма: 1.24854e+13
Итерация № 19, Слагаемое: 8.97892e+11                               Промежуточная сумма: 1.33833e+13
Итерация № 20, Слагаемое: 5.53124e+11                               Промежуточная сумма: 1.39365e+13
Итерация № 21, Слагаемое: 3.08683e+11                               Промежуточная сумма: 1.42451e+13
Итерация № 22, Слагаемое: 1.56789e+11                               Промежуточная сумма: 1.44019e+13
Итерация № 23, Слагаемое: 7.27894e+10                               Промежуточная сумма: 1.44747e+13
Итерация № 24, Слагаемое: 3.10065e+10                               Промежуточная сумма: 1.45057e+13
Итерация № 25, Слагаемое: 1.21621e+10                               Промежуточная сумма: 1.45179e+13
Итерация № 26, Слагаемое: 4.40717e+09                               Промежуточная сумма: 1.45223e+13
Итерация № 27, Слагаемое: 1.47984e+09                               Промежуточная сумма: 1.45238e+13
Итерация № 28, Слагаемое: 4.61728e+08                               Промежуточная сумма: 1.45242e+13
Итерация № 29, Слагаемое: 1.34217e+08                               Промежуточная сумма: 1.45244e+13
Итерация № 30, Слагаемое: 3.64357e+07                               Промежуточная сумма: 1.45244e+13
Итерация № 31, Слагаемое: 9.25824e+06                               Промежуточная сумма: 1.45244e+13
Итерация № 32, Слагаемое: 2.20664e+06                               Промежуточная сумма: 1.45244e+13
Итерация № 33, Слагаемое: 494308                                    Промежуточная сумма: 1.45244e+13
Итерация № 34, Слагаемое: 104265                                    Промежуточная сумма: 1.45244e+13
Итерация № 35, Слагаемое: 20745                             Промежуточная сумма: 1.45244e+13
Итерация № 36, Слагаемое: 3899.83                                   Промежуточная сумма: 1.45244e+13
Итерация № 37, Слагаемое: 693.769                                   Промежуточная сумма: 1.45244e+13
Итерация № 38, Слагаемое: 116.967                                   Промежуточная сумма: 1.45244e+13
Итерация № 39, Слагаемое: 18.7155                                   Промежуточная сумма: 1.45244e+13
Итерация № 40, Слагаемое: 2.84582                                   Промежуточная сумма: 1.45244e+13
Итерация № 41, Слагаемое: 0.411749                                  Промежуточная сумма: 1.45244e+13
Итерация № 42, Слагаемое: 0.0567542                                 Промежуточная сумма: 1.45244e+13
Итерация № 43, Слагаемое: 0.00746113                                Промежуточная сумма: 1.45244e+13
Итерация № 44, Слагаемое: 0.000936539                               Промежуточная сумма: 1.45244e+13
Итерация № 45, Слагаемое: 0.000112361                               Промежуточная сумма: 1.45244e+13
Итерация № 46, Слагаемое: 1.28977e-05                               Промежуточная сумма: 1.45244e+13
Итерация № 47, Слагаемое: 1.41783e-06                               Промежуточная сумма: 1.45244e+13
Итерация № 48, Слагаемое: 1.494e-07                                 Промежуточная сумма: 1.45244e+13
Итерация № 49, Слагаемое: 1.51035e-08                               Промежуточная сумма: 1.45244e+13
Итерация № 50, Слагаемое: 1.46611e-09                               Промежуточная сумма: 1.45244e+13
Итерация № 51, Слагаемое: 1.36763e-10                               Промежуточная сумма: 1.45244e+13

x: 31, Точность: 1e-10, Итоговая сумма: 1.45244e+13, Сумма2: 1.45244e+13
К-во итераций n = 52
Для продолжения нажмите любую клавишу . . .
```

Совпадение результатов вычислений по «разным» формулам (на самом деле они математически идентичны) является доказательством того, что рассчитанная нашей программой сумма членов ряда верна.

Для последнего задания нужно выводить таблицу значений для икса, изменяющегося от a до b с шагом h (h рассчитывается по формуле от значений a, b и n – количества шагов, запрашиваемых у пользователя). Программа модифицируется к

такому виду:



```
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    long long int n = 0;
    long double x, a, b, h;
    cout << "Введите a: ";
    cin >> a;
    cout << "Введите b: ";
    cin >> b;
    if (a > b)
    {
        long double t = a;
        a = b;
        b = t;
        cout << "Значения a и b поменяны.\n";
    }
    cout << "Введите количество шагов между a и b (n <= 20, так как n влияет на расчет факториала): ";
    cin >> n;
    if (n < 0)
    {
        n = -n;
        cout << "Количество шагов должно быть неотрицательным. n: " << n << endl;
    }
    h = (b - a) / n;//n определяет количество шагов между a и b
    long double s0 = 0, S = 0, Y = 0, raznost = 0;//очередное слагаемое s0, Итоговая сумма S
    cout << "|\tx\t|\tS(x)\t|\tY(x)\t|\t|S(x)-Y(x)|\t|\n";//"шапка" таблицы
    for (x = a; x <= b; x = x + h)//x изменяется от a до b с шагом h
    {
        for (int i = 0; i < n + 1; i += 2)//для очередного x рассчитать все слагаемые, количеством n
        {
            unsigned long long int f = 1;//для расчета факториала в каждой итерации внешнего цикла
          for (int j = 1; j < i + 1; j++)//i здесь видна, поэтому создаем j
          {//j увеличивается до i включительно, т.к. факториал перемножается от 1 до самого числа включительно
              f = f * j;
          }
          s0 = pow(x, i) / f;//вычисление очередного члена ряда//x возводится в степень, которую хранит i
          S = S + s0;//увеличение итоговой суммы для очередного x на значение очередного члена ряда (слагаемого)
        }
        Y = (exp(x) + exp(-x)) / 2;//расчет аналогичного результата по другой формуле в одно выражение
        raznost = fabs(S - Y);//расчет модуля разности полученных значений//он должен стремиться к нулю 0.0000YXYX
        cout << "|\t" << x << "\t|\t" << S << "\t|\t" << Y << "\t|\t" << raznost << "\t|\n";//печать строки таблицы
        S = 0;//обнуляем S, поскольку в следующей итерации рассчитывается новая сумма для нового x
    }
    system("pause");
    return 0;
}
```

Проблемы не найдены.

ошибок

ешение | 0 Ошибки | 0 Предупреждения | 0 Сообщения | Сборка и IntelliSense

Тестируем, получаем на консоли таблицу. Обратите внимание, что значения в столбце S(x) и Y(x) совпадают в целой части и в двух-пяти разрядах значащих цифр после запятой (по заданию требуется совпадение хотя бы в целой части и двух-четырех разрядах после запятой). На самом деле нам выводится количество цифр после запятой по умолчанию, а в реальном числе типа long double цифр еще больше. Для этого есть столбец модуля разности S(x) и Y(x), который покажет

разницу, которая в ряде случаев равна нулю (или стремится к нему), а самый «худший» результат моей программы на первом тестовом наборе данных дал разницу 1.33227e-15 (это научный формат записи числа – scientific notation), что равно $1.33227 * 10^{-15}$ или 0.0000000000000133227 – тоже близкое нулю значение.

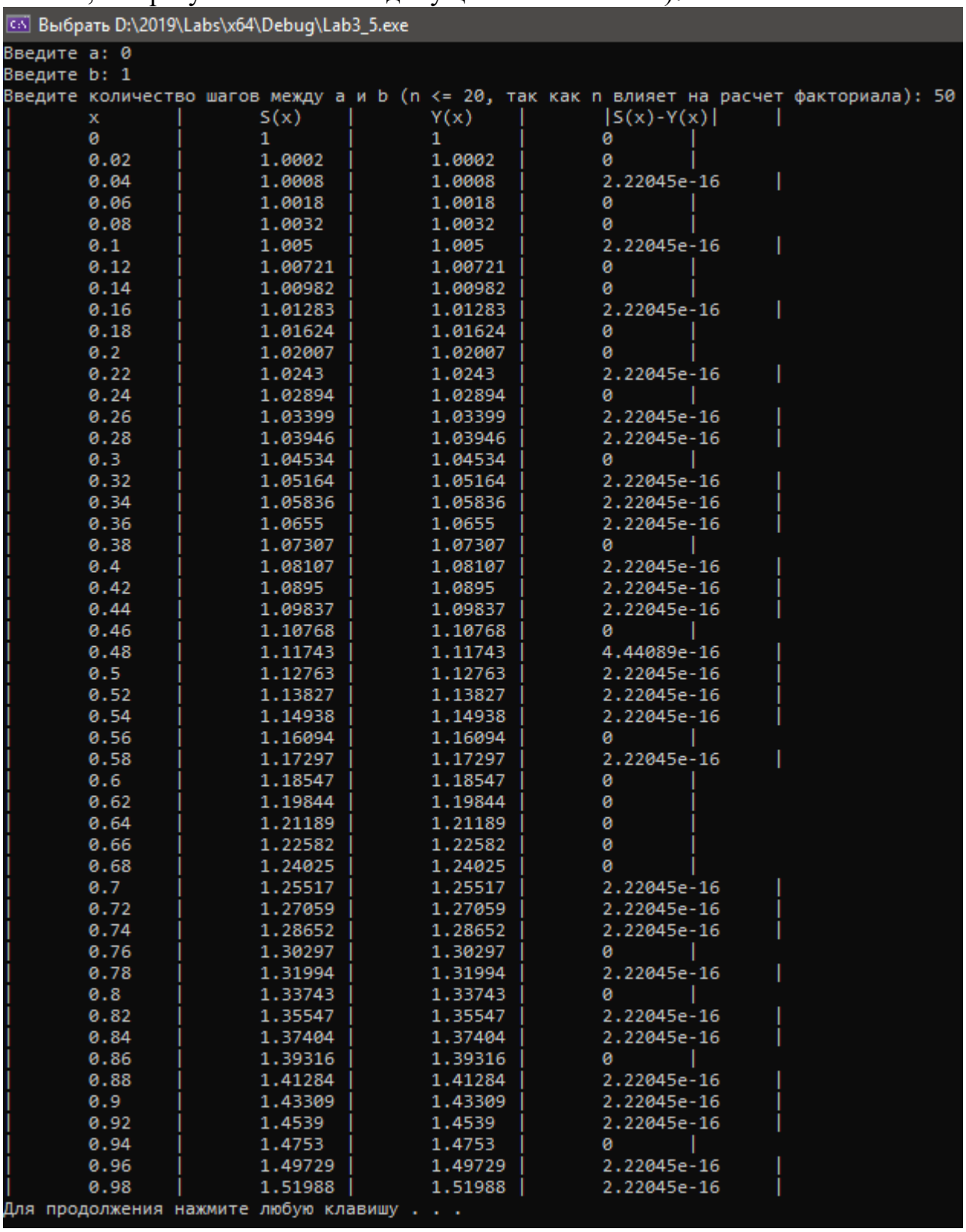
```
D:\2019\Labs\x64\Debug\Lab3_5.exe
Введите a: 0
Введите b: 2
Введите количество шагов между a и b (n <= 20, так как n влияет на расчет факториала): 20
|       x       |       S(x)    |       Y(x)    |       |S(x)-Y(x)|     |
|       0       |       1       |       1       |       0       |
|       0.1     |       1.005   |       1.005   |       2.22045e-16     |
|       0.2     |       1.02007 |       1.02007 |       0       |
|       0.3     |       1.04534 |       1.04534 |       0       |
|       0.4     |       1.08107 |       1.08107 |       0       |
|       0.5     |       1.12763 |       1.12763 |       0       |
|       0.6     |       1.18547 |       1.18547 |       0       |
|       0.7     |       1.25517 |       1.25517 |       2.22045e-16     |
|       0.8     |       1.33743 |       1.33743 |       2.22045e-16     |
|       0.9     |       1.43309 |       1.43309 |       0       |
|       1       |       1.54308 |       1.54308 |       2.22045e-16     |
|       1.1     |       1.66852 |       1.66852 |       4.44089e-16     |
|       1.2     |       1.81066 |       1.81066 |       2.22045e-16     |
|       1.3     |       1.97091 |       1.97091 |       0       |
|       1.4     |       2.1509  |       2.1509  |       8.88178e-16     |
|       1.5     |       2.35241 |       2.35241 |       0       |
|       1.6     |       2.57746 |       2.57746 |       8.88178e-16     |
|       1.7     |       2.82832 |       2.82832 |       0       |
|       1.8     |       3.10747 |       3.10747 |       4.44089e-16     |
|       1.9     |       3.41773 |       3.41773 |       1.33227e-15     |
Для продолжения нажмите любую клавишу . . .
```

Протестируем программу на других наборах данных:

```
D:\2019\Labs\x64\Debug\Lab3_5.exe
Введите a: 0
Введите b: 3
Введите количество шагов между a и b (n <= 20, так как n влияет на расчет факториала): 20
|       x       |       S(x)    |       Y(x)    |       |S(x)-Y(x)|     |
|       0       |       1       |       1       |       0       |
|       0.15    |       1.01127 |       1.01127 |       0       |
|       0.3     |       1.04534 |       1.04534 |       0       |
|       0.45    |       1.10297 |       1.10297 |       4.44089e-16     |
|       0.6     |       1.18547 |       1.18547 |       0       |
|       0.75    |       1.29468 |       1.29468 |       2.22045e-16     |
|       0.9     |       1.43309 |       1.43309 |       0       |
|       1.05    |       1.60379 |       1.60379 |       0       |
|       1.2     |       1.81066 |       1.81066 |       2.22045e-16     |
|       1.35    |       2.05833 |       2.05833 |       4.44089e-16     |
|       1.5     |       2.35241 |       2.35241 |       4.44089e-16     |
|       1.65    |       2.69951 |       2.69951 |       0       |
|       1.8     |       3.10747 |       3.10747 |       0       |
|       1.95    |       3.58548 |       3.58548 |       1.77636e-15     |
|       2.1     |       4.14431 |       4.14431 |       1.06581e-14     |
|       2.25    |       4.79657 |       4.79657 |       5.06262e-14     |
|       2.4     |       5.55695 |       5.55695 |       2.08722e-13     |
|       2.55    |       6.44259 |       6.44259 |       7.90479e-13     |
|       2.7     |       7.47347 |       7.47347 |       2.7871e-12      |
|       2.85    |       8.67281 |       8.67281 |       9.16778e-12     |
|       3       |       10.0677 |       10.0677 |       2.83809e-11     |
Для продолжения нажмите любую клавишу . . .
```

Протестируем программу на других наборах данных (тут введено значение, которое заставляет программу рассчитывать факториалы чисел до 50, но сужением

диапазона для x от 0 до 1 удается получить приемлемые результаты, хотя одинаковая разность свидетельствует о выходе за допустимый диапазон для рассчитанных факториалов, но числа в делителе (то есть знаменателе дроби) все равно так велики, что результат лоялен к допущенным ошибкам):



```
Выбрать D:\2019\Labs\x64\Debug\Lab3_5.exe
Введите a: 0
Введите b: 1
Введите количество шагов между a и b (n <= 20, так как n влияет на расчет факториала): 50
|       x       |       S(x)    |       Y(x)    |       |S(x)-Y(x)|     |
|       0       |       1       |       1       |       0       |
|       0.02    |       1.0002  |       1.0002  |       0       |
|       0.04    |       1.0008  |       1.0008  |       2.22045e-16     |
|       0.06    |       1.0018  |       1.0018  |       0       |
|       0.08    |       1.0032  |       1.0032  |       0       |
|       0.1     |       1.005   |       1.005   |       2.22045e-16     |
|       0.12    |       1.00721 |       1.00721 |       0       |
|       0.14    |       1.00982 |       1.00982 |       0       |
|       0.16    |       1.01283 |       1.01283 |       2.22045e-16     |
|       0.18    |       1.01624 |       1.01624 |       0       |
|       0.2     |       1.02007 |       1.02007 |       0       |
|       0.22    |       1.0243  |       1.0243  |       2.22045e-16     |
|       0.24    |       1.02894 |       1.02894 |       0       |
|       0.26    |       1.03399 |       1.03399 |       2.22045e-16     |
|       0.28    |       1.03946 |       1.03946 |       2.22045e-16     |
|       0.3     |       1.04534 |       1.04534 |       0       |
|       0.32    |       1.05164 |       1.05164 |       2.22045e-16     |
|       0.34    |       1.05836 |       1.05836 |       2.22045e-16     |
|       0.36    |       1.0655  |       1.0655  |       2.22045e-16     |
|       0.38    |       1.07307 |       1.07307 |       0       |
|       0.4     |       1.08107 |       1.08107 |       2.22045e-16     |
|       0.42    |       1.0895  |       1.0895  |       2.22045e-16     |
|       0.44    |       1.09837 |       1.09837 |       2.22045e-16     |
|       0.46    |       1.10768 |       1.10768 |       0       |
|       0.48    |       1.11743 |       1.11743 |       4.44089e-16     |
|       0.5     |       1.12763 |       1.12763 |       2.22045e-16     |
|       0.52    |       1.13827 |       1.13827 |       2.22045e-16     |
|       0.54    |       1.14938 |       1.14938 |       2.22045e-16     |
|       0.56    |       1.16094 |       1.16094 |       0       |
|       0.58    |       1.17297 |       1.17297 |       2.22045e-16     |
|       0.6     |       1.18547 |       1.18547 |       0       |
|       0.62    |       1.19844 |       1.19844 |       0       |
|       0.64    |       1.21189 |       1.21189 |       0       |
|       0.66    |       1.22582 |       1.22582 |       0       |
|       0.68    |       1.24025 |       1.24025 |       0       |
|       0.7     |       1.25517 |       1.25517 |       2.22045e-16     |
|       0.72    |       1.27059 |       1.27059 |       2.22045e-16     |
|       0.74    |       1.28652 |       1.28652 |       2.22045e-16     |
|       0.76    |       1.30297 |       1.30297 |       0       |
|       0.78    |       1.31994 |       1.31994 |       2.22045e-16     |
|       0.8     |       1.33743 |       1.33743 |       0       |
|       0.82    |       1.35547 |       1.35547 |       2.22045e-16     |
|       0.84    |       1.37404 |       1.37404 |       2.22045e-16     |
|       0.86    |       1.39316 |       1.39316 |       0       |
|       0.88    |       1.41284 |       1.41284 |       2.22045e-16     |
|       0.9     |       1.43309 |       1.43309 |       2.22045e-16     |
|       0.92    |       1.4539  |       1.4539  |       2.22045e-16     |
|       0.94    |       1.4753  |       1.4753  |       0       |
|       0.96    |       1.49729 |       1.49729 |       2.22045e-16     |
|       0.98    |       1.51988 |       1.51988 |       2.22045e-16     |
Для продолжения нажмите любую клавишу . . .
```

2. Пример выполнения программы

Вычислить рекуррентное выражение $\sum_{к=1}^{20} -1^{к} \frac{x^{к}}{к!}$.

Текст программы, которая без рекуррентной оптимизации вычисляет данное выражение, рассчитывая каждый член ряда, для вычисления которого надо вычислять факториал (поэтому k вводим не больше 20):

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    double x, sum = 0.0;
    cout << "x: ";
    cin >> x;
    int k0, k1;
    cout << "Начальное k: ";
    cin >> k0;//k0 = 1;
    cout << "Конечное k: ";
    cin >> k1;//k1 = 20;//факториал нарастает очень быстро, не вводите числа больше 20
    if (k0 > k1)
    {
        int t = k0;
        k0 = k1;
        k1 = t;
        cout << "\nЗначения поменялись местами.\n";
    }
    for (int k = k0; k < k1 + 1; k++)
    {
        unsigned long long int f = 1;//факториал расчитывается для каждой "большой" итерации заново
        for (int i = 1; i < k + 1; i++)
        {
            f = f * i;
        }
        cout << "Факториал " << k << ": " << f << endl;//для проверки корректности расчета промежуточных значений
        sum = sum + (pow(-1, k)*(pow(x, k)/f));//к значению суммы предыдущей итерации прибавить значение, расчитанное в этой итерации
        cout << "Промежуточная сумма: " << sum << endl;//для проверки корректности расчета промежуточных значений
    }
    cout << "\nИтоговая сумма: " << sum << endl;
    system("pause");
    return 0;
}
```

Тестируем:

C:\Users\Student416\source\repos\Project2\Debug\Project2.exe

```
x: 2.98
Начальное k: 1
Конечное k: 20
Факториал 1: 1
Промежуточная сумма: -2.98
Факториал 2: 2
Промежуточная сумма: 1.4602
Факториал 3: 6
Промежуточная сумма: -2.9504
Факториал 4: 24
Промежуточная сумма: 0.335497
Факториал 5: 120
Промежуточная сумма: -1.6229
Факториал 6: 720
Промежуточная сумма: -0.650228
Факториал 7: 5040
Промежуточная сумма: -1.06431
Факториал 8: 40320
Промежуточная сумма: -0.910062
Факториал 9: 362880
Промежуточная сумма: -0.961134
Факториал 10: 3628800
Промежуточная сумма: -0.945915
Факториал 11: 39916800
Промежуточная сумма: -0.950038
Факториал 12: 479001600
Промежуточная сумма: -0.949014
Факториал 13: 6227020800
Промежуточная сумма: -0.949249
Факториал 14: 87178291200
Промежуточная сумма: -0.949199
Факториал 15: 1307674368000
Промежуточная сумма: -0.949209
Факториал 16: 20922789888000
Промежуточная сумма: -0.949207
Факториал 17: 355687428096000
Промежуточная сумма: -0.949207
Факториал 18: 6402373705728000
Промежуточная сумма: -0.949207
Факториал 19: 121645100408832000
Промежуточная сумма: -0.949207
Факториал 20: 2432902008176640000
Промежуточная сумма: -0.949207

Итоговая сумма: -0.949207
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Student416\source\repos\Project2\Debug\Project2.exe
x: -9.8765
Начальное k: 20
Конечное k: 1

Значения поменялись местами.
Факториал 1: 1
Промежуточная сумма: 9.8765
Факториал 2: 2
Промежуточная сумма: 58.6491
Факториал 3: 6
Промежуточная сумма: 219.217
Факториал 4: 24
Промежуточная сумма: 615.678
Факториал 5: 120
Промежуточная сумма: 1398.81
Факториал 6: 720
Промежуточная сумма: 2687.91
Факториал 7: 5040
Промежуточная сумма: 4506.73
Факториал 8: 40320
Промежуточная сумма: 6752.18
Факториал 9: 362880
Промежуточная сумма: 9216.32
Факториал 10: 3628800
Промежуточная сумма: 11650
Факториал 11: 39916800
Промежуточная сумма: 13835.2
Факториал 12: 479001600
Промежуточная сумма: 15633.6
Факториал 13: 6227020800
Промежуточная сумма: 17000
Факториал 14: 87178291200
Промежуточная сумма: 17963.9
Факториал 15: 1307674368000
Промежуточная сумма: 18598.5
Факториал 16: 20922789888000
Промежуточная сумма: 18990.3
Факториал 17: 355687428096000
Промежуточная сумма: 19217.9
Факториал 18: 6402373705728000
Промежуточная сумма: 19342.8
Факториал 19: 121645100408832000
Промежуточная сумма: 19407.7
Факториал 20: 2432902008176640000
Промежуточная сумма: 19439.8

Итоговая сумма: 19439.8
Для продолжения нажмите любую клавишу . . .
```

3. Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализуйте алгоритм решения задачи. Вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ для x, изменяющегося от a до b с шагом $h = (b - a)/n$; где a=0.1, b=1, n=100, $Y(x)=e^{2x}$, $S(x) = 1 + \frac{2x}{1!} + \cdots + \frac{(2x)^n}{n!}$.

3. Разработать на языке C++ программу вывода на экран решения задачи в соответствии с вариантом индивидуального задания, указанным преподавателем.
4. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
5. Ответить на контрольные вопросы.

4. Задания по вариантам:

4.1 Задание. Везде, где это возможно, входные данные в программу вводит пользователь с клавиатуры

№ вар.	Задание
1	Напечатать таблицу перевода расстояний из дюймов в сантиметры для значений длин от 1 до 20 дюймов 1 дюйм = 2,54 см .
2	Вывести все четные числа кратные пяти в интервале от 2 до 100 включительно.
3	Даны натуральные числа от -500 до 500. Найти все трехзначные числа, у которых четные сотни.
4	Определить сумму модулей всех нечетных, отрицательных чисел от -99 до 99.
5	Даны натуральные числа от 0 до 700. Найти все трехзначные числа, у которых нечетные сотни.
6	Получить в порядке убывания все делители данного числа.
7	Составьте программу определения наибольшего общего делителя двух натуральных чисел .
8	Составьте программу определения наименьшего общего кратного двух натуральных чисел .
9	Составьте программу, подсчитывающую количество цифр вводимого вами целого неотрицательного числа . Можно использовать операцию целочисленного деления.
10	Даны числа от 1 до 1000 и число m. Вывести результат умножение куба нечетных сотен на число m.
11	Даны числа от 1 до 1000 и число m. Вывести результат деления квадрата сотен кратных 5 на число m.
12	Дано число n от 1 до 1000 и число m. Вывести результат квадрат разности числа n и число m.
13	Вычислить: $1+2+4+8+\ldots+2^{10}$ и $(1+2)*(1+2+3)*\ldots*(1+2+\ldots+10)$.
14	Даны числа от 1 до 1000 и число m. Вывести результат целочисленного деления нечетных сотен на число m.
15	Билет называют «счастливым», если в его номере сумма первых трех цифр равна сумме последних трех. Подсчитать число тех «счастливых» билетов, у которых сумма трех цифр равна 13. Номер билета может быть от 000000 до 999999 .
16	Дано число n от 1 до 1000 и число m. Вывести результат квадрат целочисленного деления n на m.
17	В ЭВМ вводятся по очереди данные о росте N учащихся класса. Определить средний рост учащихся в классе.
18	Составьте программу, суммирующую штрафное время команд при игре в хоккей. Выводить на экран суммарное штрафное время обеих команд после любого его изменения. После окончания игры выдать итоговое сообщение.
19	Дано натуральное число n (n<9999). Найти предпоследнюю цифру числа (в предположении, что n>10).
20	Даны числа от 1 до 1000 и число m. Вывести все остатки от деления четных сотен на число m.

№ вар.	Задание
21	Для заданного числа *N* составьте программу вычисления суммы S=1+1/2+1/3+1/4+…+1/*N*, где *N* – натуральное число.
22	Каждая бактерия делится на две в течение одной минуты. В начальный момент имеется одна бактерия. Составьте программу, которая рассчитывает количество бактерий на заданное вами целое значение момента времени (15 минут, 7 минут и т.п.) .
23	Составьте программу вывода на экран всех простых чисел, не превосходящих заданного *N*. Простым называется натуральное число больше единицы, имеющее только два делителя: единицу и само это число .
24	В 1202г итальянский математик Леонард Пизанский (Фибоначчи) предложил такую задачу: пара кроликов каждый месяц дает приплод – двух кроликов (самца и самку), от которых через два месяца уже получается новый приплод. Сколько кроликов будет через год, если в начале года имелась одна пара? Согласно условию задачи числа, соответствующие количеству кроликов, которые появляются через каждый месяц, составляют последовательность 1, 1, 2, 3, 5, 8, 13, 21, 37, … Составьте программу, позволяющую найти все числа Фибоначчи, меньшие заданного числа *N*.
25	Для чисел от 1 до 1000, найти сотни в которых есть внутренние повторение (например 122, 133, 144, 677 и т.д.)
26	Для чисел от 1 до 1000. Найти количество трехзначных чисел, все цифры которых одинаковы.
27	Для чисел от 1 до 1000 . Найти все нечетные сотни в которой есть повторение чисел.
28	Для чисел от 1 до 1000, возвести в куб каждый третий десяток каждой второй сотни.
29	Дано натуральное число *n* (*n*>999). Определить число сотен в нём
30	Даны натуральные числа от 0 до *n* (*n*<99) и число *m*. И найти квадрат первого числа больше *m* .

4.2 Задание:

Данная задача предполагает получение (печать на консоль) таблицы данных: **| x | S(x) | Y(x) | Y(x)-S(x) |**, где значение аргумента x изменяется от a до b с шагом h.

Для каждого x найти значения функции Y(x), суммы S(x) и модуля разности |Y(x)-S(x)|. Результаты расчетов вывести в виде таблицы на консоль. Значения a, b, h и n вводятся с клавиатуры пользователем. Так как значение S(x) является рядом разложения функции Y(x), то значения S и Y для текущего аргумента x должны совпадать в целой части и в двух-четырех позициях после десятичной точки. Близость значений S(x) и Y(x) во всем диапазоне значений x указывает на правильность их вычисления.

Работу программы проверить для (набор тестовых данных для ввода пользователем с клавиатуры) a=0,1; b=1,0; h=0,1; n выбрать самостоятельно.

Знак «сигма» $\sum$ означает сумму всех значений выражения, которое записано справа от этого знака. В формулу выражения подставляется текущее значение **x** и это выражение вычисляется для всех k начиная от того значения, которое пишется снизу под $\sum$ до того, которое пишется сверху над $\sum$. Например, в первом варианте k начинается с нуля, далее увеличивается на единицу, пока не достигнет значения n, которое было заранее определено пользователем с клавиатуры. То есть для одного значения **x** нужно в цикле вычислить выражения, в первом из которых k будет равно 0, во втором k=1, в третьем – k=2 и так далее, пока в последнем выражении k получит значение n. Все вычисленные значения этих выражений нужно просуммировать. Это делается в цикле. Но и сам **x** изменяется от **a** до **b** с шагом **h** – это делается во внешнем цикле. Полученные во внешнем цикле значения выражений тоже суммируются.

Восклицательный знак «!» означает факториал. Например, 5! = 1*2*3*4*5 = 120. Символ **e** – это экспонента.

Данные, вводимые пользователем сразу проверяйте на корректность. Например, n должно быть целым положительным; b должно быть больше либо равно a; h должно быть меньше разности между b и a.

ВАРИАНТЫ:

1. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k+1}}{(2k+1)!},\quad Y(x)=\sin(x).$

2. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{\sin(kx)}{k},\quad Y(x)=\frac{x}{2}.$

3. $S(x)=\sum_{k=0}^{n}\frac{\cos(\frac{k\pi}{4})}{k!}x^k,\quad Y(x)=e^{x\cos\frac{\pi}{4}}\cos(x\sin(\frac{\pi}{4})).$

4. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k}}{(2k)!},\quad Y(x)=\cos(x).$

5. $S(x)=\sum_{k=0}^{n}\frac{\cos(kx)}{k!},\quad Y(x)=e^{\cos x}\cos(\sin(x)).$

6. $S(x)=\sum_{k=0}^{n}\frac{2k+1}{k!}x^{2k},\quad Y(x)=(1+2x^2)e^{x^2}.$

7. $S(x)=\sum_{k=1}^{n}\frac{x^k\cos\frac{k\pi}{3}}{k},\quad Y(x)=-\frac{1}{2}\ln(1-2x\cos\frac{\pi}{3}+x^2).$

8. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{\cos(kx)}{k^2},\quad Y(x)=\frac{1}{4}\left(x^2-\pi^2/3\right).$

9. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k+1}}{4k^2-1},\quad Y(x)=\frac{1+x^2}{2}arctg(x)-x/2.$

10. $S(x)=\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!},\quad Y(x)=\frac{e^x+e^{-x}}{2}.$

11. $S(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!}(x/2)^k, \quad Y(x)=(\frac{x^2}{4}+\frac{x}{2}+1)e^{\frac{x}{2}}$.

12. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{2k^2+1}{(2k)!}x^{2k}, \quad Y(x)=(1-\frac{x^2}{2})\cos(x)-\frac{x}{2}\sin(x)$.

13. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}, \quad Y(x)=2(\cos^2 x-1)$.

14. $S(x)=\sum_{k=0}^{n}\frac{x^{2k+1}}{(2k+1)!}, \quad Y(x)=\frac{e^x-e^{-x}}{2}$.

15. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}, \quad Y(x)=x\,arctg(x)-\ln\sqrt{1+x^2}$.

16. $S(x)=\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!}, \quad Y(x)=\frac{e^x+e^{-x}}{2}$.

4.3 Задание. Сделайте логическую схему алгоритма для вашей программы из задания 4.2.

Если в вашей программе три цикла (один вложен в другой), то можно использовать специальные парные графические фигуры для обозначения цикла.

5. Контрольные вопросы

1.В каких случаях используется оператор цикла с параметром? Как он оформляется? Как он работает (что происходит при его выполнении)? Нарисуйте графическую схему выполнения.

2. Может ли тело оператора цикла с параметром не выполниться ни разу?

3. Как должен быть оформлен оператор цикла с параметром, чтобы тело цикла выполнялось при уменьшающихся значениях параметра цикла? Как он будет работать (что будет происходить при его выполнении)? Нарисуйте графическую схему выполнения.

4. Можно ли в теле оператора цикла с параметром не использовать величину- параметр цикла?

5. В каких случаях используется оператор цикла с предусловием? Как он оформляется? Как он работает (что происходит при его выполнении)? Нарисовать графическую схему выполнения.

6. В каких случаях используется оператор цикла с постусловием? Как он оформляется? Как он работает (что происходит при его выполнении)? Нарисовать графическую схему выполнения.

7. Может ли тело оператора цикла с постусловием:

а) не выполниться ни разу?

б) выполняться бесконечное число раз (или до тех пор, когда пользователь прервет его выполнение)?

8. Всегда ли можно вместо оператора цикла с параметром использовать оператор цикла с предусловием? А наоборот?

Домашнее задание

Прочитать страницы 116 – 136 книги Дейтел, Х. Как программировать на C++ / Х.Дейтел, П.Дейтел (путь на сервере: s1 / Предметы / ОАиП_Шаляпин / Дейтел Харви - Как программировать на C++.pdf).

Литература

Страуструп, Б. Язык программирования C++ / Б. Страуструп. – СПб.: БИНОМ, 1999.

Павловская, Т. А. C++. Объектно-ориентированное программирование: практикум / Павловская, Т. А., Щупак. – СПб. : Питер, 2004

Преподаватель Белокопыцкая Ю.А.

Рассмотрено на заседании цикловой комиссии……..
Протокол № ____от «___» ____2016
Председатель ЦК