**­­­­Московский Авиационный Институт**

(Национальный Исследовательский Университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект

по курсу «Информатика»

1 семестр

Задание 3.

Тема: «Вещественный тип. Приближённые вычисления. Табулирование

функций.»

|  |  |
| --- | --- |
| Студент: | Доянова В. А. |
| Группа: | М8О-111Б-22 |
| Преподаватель: | Аносова Н. П. |
| Подпись: |  |
| Оценка: |  |

Москва, 2022

**Содержание:**

**Задание3**

Формулировка3

Вариант задания3

**Работа4**

Код программы4

Описание алгоритма6

Протокол выполнения программы7

Вывод8

**Задание**

**Формулировка**

Составить программу на Си, которая печатает таблицу значений элементарной

функции, вычисленной двумя способами: по формуле Тейлора и с помощью

встроенных функций языка программирования. В качестве аргументов таблицы

взять точки разбиения отрезка [a, b] на n равных частей (n + 1 точка включая

концы отрезка), находящихся в рекомендованной области хорошей точности

формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в

сложностном смысле схеме с точностью ε\*k, где ε — машинное эпсилон

аппаратно реализованного вещественного типа для данной ЭВМ, а k —

экспериментально подбираемый коэффициент, обеспечивающий приемлемую

сходимость. Число итераций должно ограничиваться сверху числом порядка

100 Программа должна сама определять машинное ε и обеспечивать

корректные размеры генерируемой таблицы.

**Вариант задания**

Изображение выглядит как стол

Автоматически созданное описание

**Работа**

**Код программы**

#include <stdio.h>

#include <locale.h>

#include <math.h>

double st(double x, int s) {

double res = 1.0;

int i;

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

res = res \* x;

}

return res;

}

long double fact(int n) {

if (n < 0) { return 0; }

if (n == 0) { return 1; }

if (n > 0) { return n \* fact(n - 1); }

}

double eps(double x) {

double ep = x;

while (x + ep / 2.0 > x) {

ep /= 2.0;

}

return ep;

}

int main() {

setlocale(LC\_ALL, "rus");

double A = 0.0, B = 1.0;

double x, s, x0;

double sum;

double f;

int n = 0, steps = 0;

printf("Введите число разбиений отрезка\n");

scanf\_s("%d", &steps);

printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n| x |\t Ряд Тейлора\t\t |\t Функция\t | Число итераций |");

printf ("\n|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

for (x = A; x <= B; x += (B - A) / steps) {

x0 = 2;

s = x \* x;

sum = x \* x;

while (n <= 100) {

n += 1;

s = (-1) \* s \* ((x \* x \* 4) / ((x0 + 1) \* (x0 + 2)));

x0 += 2;

if (fabs(s) > eps(x)) {

sum += s;

}

else {

break;

}

}

f = st(sin(x), 2);

printf("| %.3f | \t%.18f\t | \t%.18f\t | \t %.d\t | \n", x, sum, f, n);

n = 0;

sum = 0;

}

printf("|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

return 0;

}

**Описание алгоритма**

Добавляем библиотеки ввода и вывода, математических функций.

Заводим переменные типа double:

* A – начало отрезка
* B – конец отрезка
* x – точки, на которые разбивается отрезок [A, B]
* sum – сумма членов ряда Тейлора
* s – слагаемое в ряде Тейлора
* f – элементарная функция sin²(x)

Создаем функции:

* st – возведение числа в степень
* fact – нахождение факториала
* eps – вычисление машинного эпсилон

Заводим переменные типа int:

* n – количество итераций
* steps – количество частей, на которые разбивается отрезок [A, B]

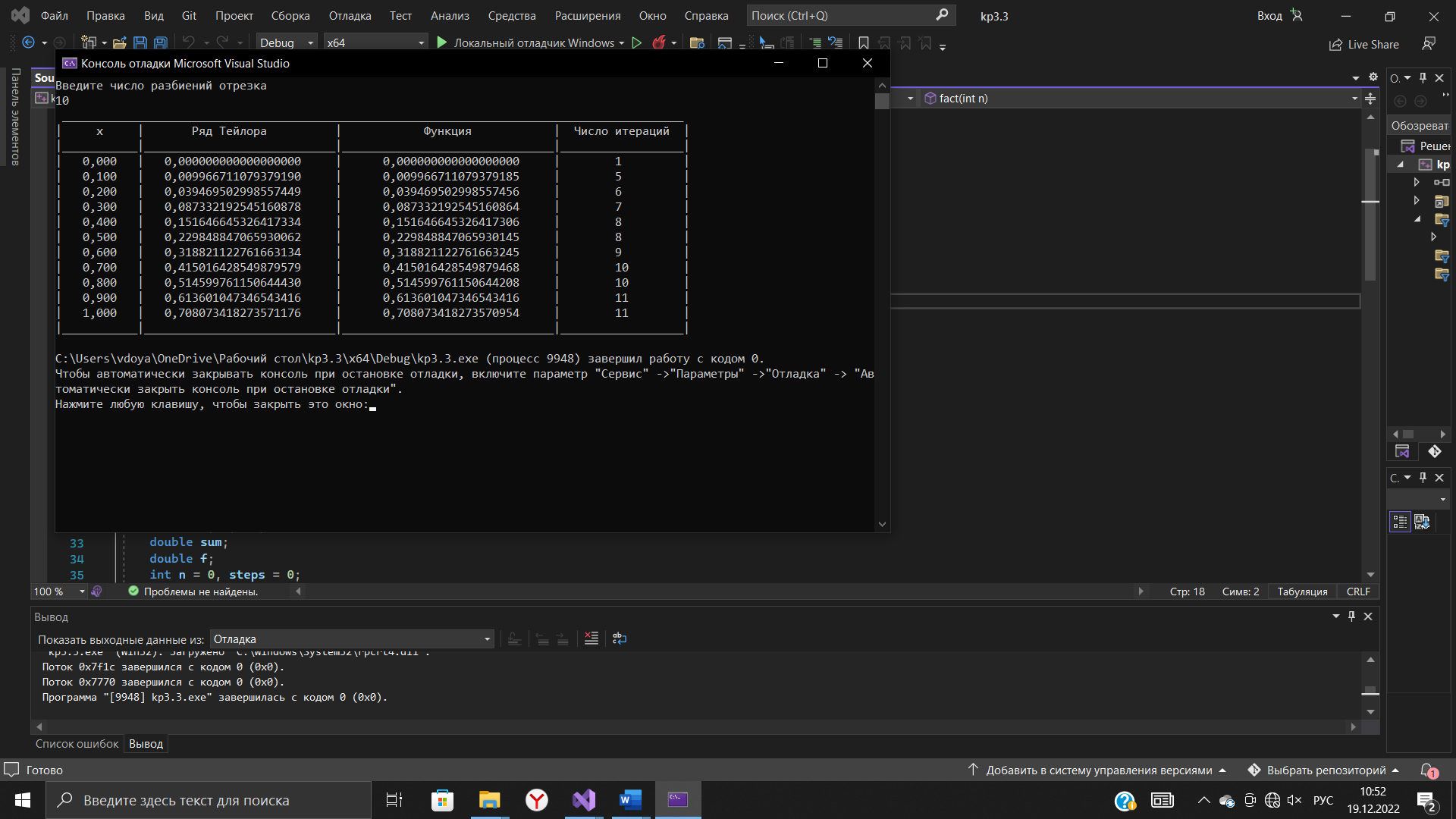
Пользователь вводит число разбиений отрезка.

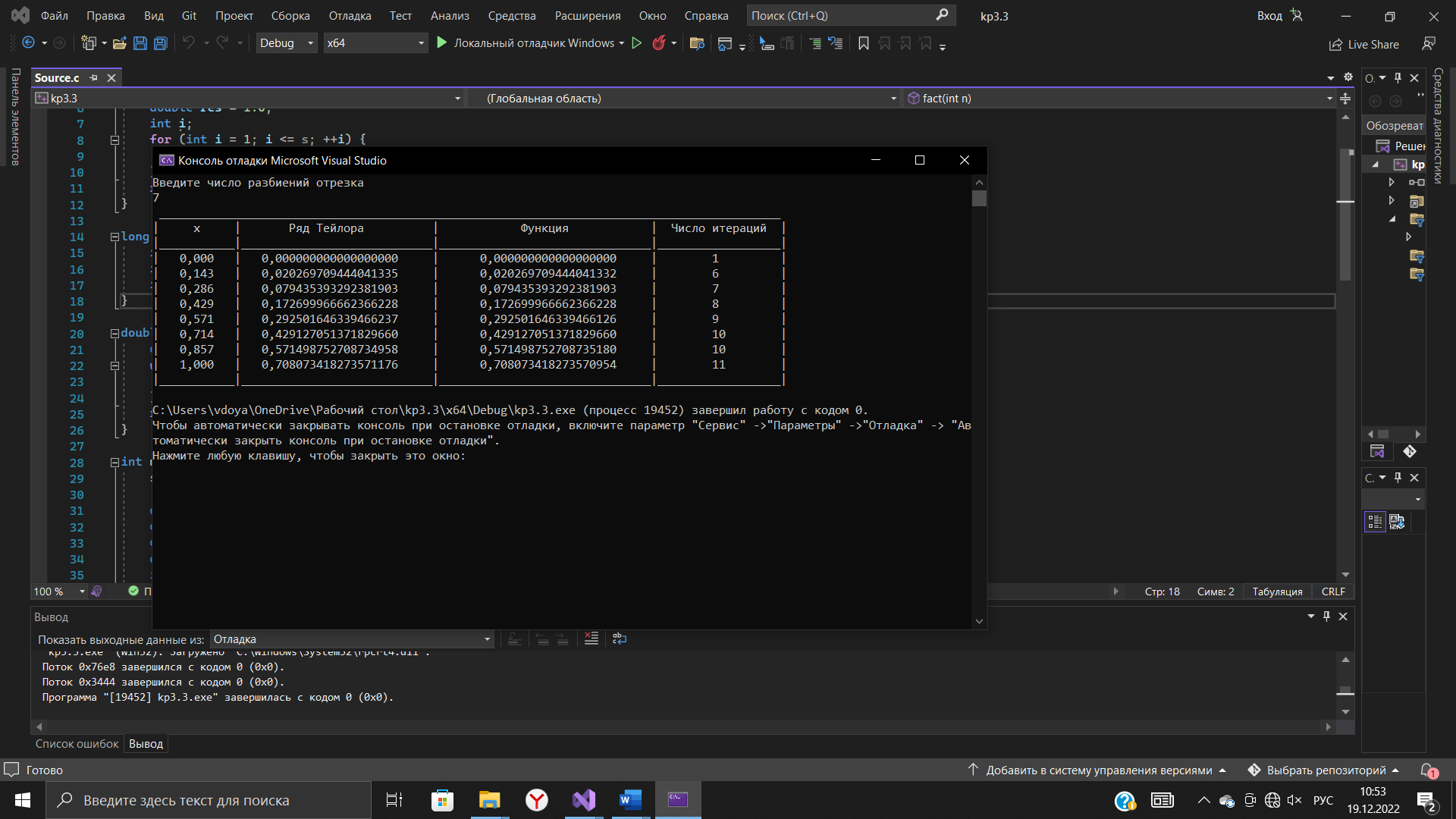
Вычисляем машинное эпсилон: находим значение eps для каждого x через функци. Проверяем результат деления eps на x, если оно оказывается больше нуля, то продолжаем деление. В противном случае останавливаем цикл.

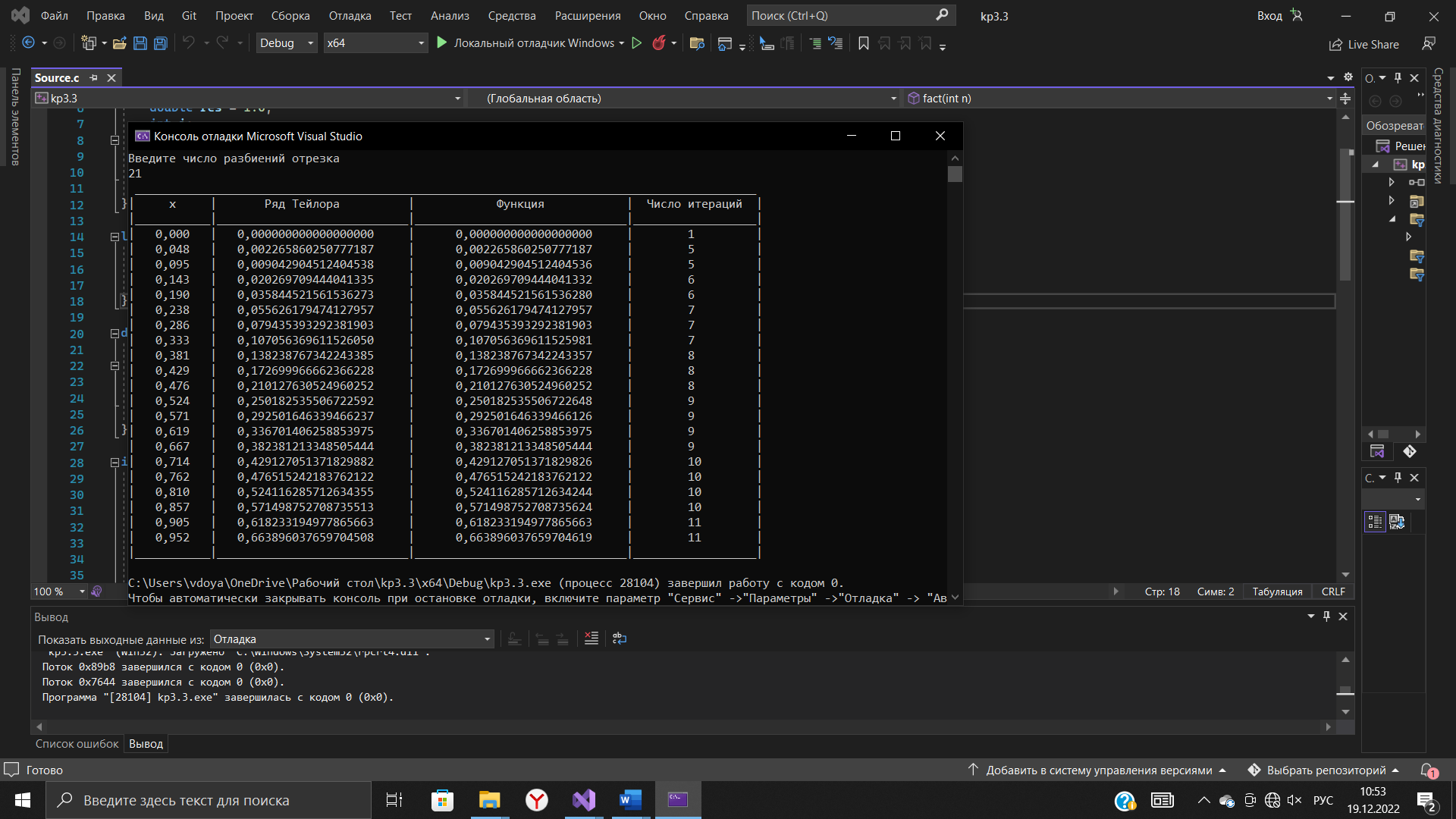
Выводим верхнюю часть таблицы.

Начинаем цикл for. Присваиваем x текущее значение точки (начиная с начала отрезка). Запоминаем начальное значение в x0 и начальное значение суммы. Начинаем цикл while, пока n меньше 100 (каждый раз увеличиваем переменную на 1). Рассчитываем текущее значение s. Если модуль s больше eps, то прибавляем s к sum, иначе заканчиваем цикл. Рассчитываем значение f. Выводим основную часть таблицы, обнуляем n, sum. Выводим нижнюю часть таблицы.

**Протокол выполнения программы**







**Вывод**

Таблица показывает, что значения ряда Тейлора имеют отличия от встроенной функции примерно после 16 знака после запятой. Это значит, что такой метод задания функции не является идеальным при необходимости получения точности более 15 знаков после запятой.

В ходе выполнения данного задания курсового проекта я научилась реализовывать программную версию вычисления значений функции, пользуясь рядом Тейлора, определять машинное эпсилон. Я научилась выводить данный в виде таблицы, оптимизировать вывод значений переменных с плавающей точкой.