**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Физико-технический факультет

Кафедра компьютерных технологий

Дисциплина «Вычислительная математика»

Отчет к лабораторной работе № 1

«Вычисления с плавающей точкой»

Выполнил:

студент группы ИВТ-6

Давыденко Дмитрий Александрович

Принял:

асс. Пшеничный Кирилл Анатольевич

Донецк 2022

**ЗАДАНИЕ**

1. Найти границы начала машинного ноля и машинной бесконечности для вещественных чисел стандартной и повышенной точности. Соответствуют ли полученные значения действительности?
2. Ряд Тейлора для функции ошибок имеет вид:



Этот ряд сходится для всех *x*.

Вычислить erf(3) по указанной формуле с точно­стью до нулевого члена ряда для следующих случаев:

* Без преобразований исходной формулы;
* Сделать расчёт факториала с числом *n* целого типа,
* Сделать расчёт факториала сделать с числом *n* вещественного типа.
* Повторить расчеты, изменив формулу функции ошибок с вычислением очередного члена ряда через предыдущий член *an=kan*-1, т.е. найти *k* и таким образом избавиться от факториала.

Объяснить все четыре полученных результата. Каким знакам (цифрам) в результате erf(3) доверять нельзя? Была ли катастрофическая потеря верных знаков? В каких знаках делались округления?

**ХОД РАБОТЫ**

1. Найти границы начала машинного ноля и машинной бесконечности для вещественных чисел стандартной и повышенной точности. Соответствуют ли полученные значения действительности?

Репозиторий проекта:  
<https://github.com/uncleDimasik/computational-mathematics-22-23-laba-1>

Листинг программы:

<https://github.com/uncleDimasik/computational-mathematics-22-23-laba-1/blob/master/VMathLaba1.1/VMathLaba1.1/Program.cs>

**using** **System**;

**namespace** **VMathLaba1.\_1**

{

**class** **Program**

{

**static** **void** **Main**(**string**[] args)

{

Console.WriteLine("Расчеты");

Console.Write("\nМашинный эпсилон для типа float, полученный при расчетах: ");

PrintColoredText(**string**.Format("{0}\n", GetFloatMachineEpsilon()), ConsoleColor.Cyan);

Console.Write("\nВычисленный Машинный ноль для типа float в C#: ");

PrintColoredText(**string**.Format("{0}\n", GeFloatMachineZero()), ConsoleColor.Cyan);

Console.Write("\nМашинный эпсилон для типа double, полученный при расчетах: ");

PrintColoredText(**string**.Format("{0}\n", GetDoubleMachineEpsilon()), ConsoleColor.Cyan);

Console.Write("\nВычисленный Машинный ноль для типа double в C#: ");

PrintColoredText(**string**.Format("{0}\n", GetDoubleMachineZero()), ConsoleColor.Cyan);

Console.Write("\nМашинная бесконечность для типа float равна: ");

PrintColoredText(**string**.Format("{0}\n", GetFloatMachineInfinity()), ConsoleColor.Cyan);

Console.Write("\nМашинная бесконечность для типа double: ");

PrintColoredText(**string**.Format("{0}\n", GetDoubleMachineInfinity()), ConsoleColor.Cyan);

Console.WriteLine("\n\nРеференсные значения в .NET");

Console.Write("\nМашинный ноль для типа float в C#: ");

PrintColoredText(**string**.Format("{0}\n", **float**.Epsilon), ConsoleColor.Cyan);

Console.Write("\nМашинный ноль для типа double в C#: ");

PrintColoredText(**string**.Format("{0}\n", **double**.Epsilon), ConsoleColor.Cyan);

Console.ReadKey();

}

**private** **static** **void** **PrintColoredText**(**string** text, ConsoleColor color)

{

Console.ForegroundColor = color;

Console.Write(text);

Console.ResetColor();

}

**private** **static** **float** **GetFloatMachineEpsilon**()

{

**float** epsilon = **1f**;

**float** **value** = **1f**;

**while** (**value** + **1f** > **1f**)

{

**value** /= **2**;

**if** (**value** != **0**)

{

epsilon = **value**;

}

}

**return** epsilon;

}

**private** **static** **double** **GetDoubleMachineEpsilon**()

{

**double** epsilon = **1d**;

**double** **value** = **1d**;

**while** (**value** + **1d** > **1d**)

{

**value** /= **2d**;

**if** (**value** != **0**)

{

epsilon = **value**;

}

}

**return** epsilon;

}

**private** **static** **float** **GetFloatMachineInfinity**()

{

**float** infinity = **1f**;

**float** **value** = **1f**;

**while** (!**float**.IsInfinity(**value**))

{

infinity = **value**;

**value** \*= **2**;

}

**return** infinity;

}

**private** **static** **double** **GetDoubleMachineInfinity**()

{

**double** infinity = **1d**;

**double** **value** = **1d**;

**while** (!**double**.IsInfinity(**value**))

{

infinity = **value**;

**value** \*= **2**;

}

**return** infinity;

}

**private** **static** **double** **GetDoubleMachineZero**()

{

**double** zero = **0.1d**;

**double** **value** = **0.1d**;

**while** (**value** != **0**)

{

**if** ((**value** / **2**) <= **0d**)

{

zero = **value**;

}

**value** /= **2**;

}

**return** zero;

}

**private** **static** **double** **GeFloatMachineZero**()

{

**float** zero = **0.1f**;

**float** **value** = **0.1f**;

**while** (**value** != **0f**)

{

**if** ((**value** / **2**) <= **0f**)

{

zero = **value**;

}

**value** /= **2**;

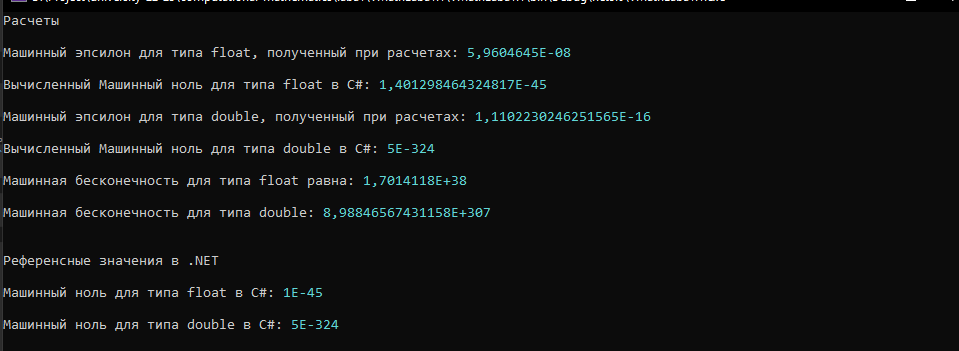
}

**return** zero;

}

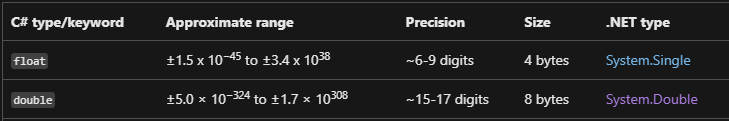
}

Результат работы программы:



**Соответствуют ли полученные значения действительности?**

Линк: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/floating-point-numeric-types>



Сравнив полученные значения с официальной документацией можно отметить что полученные значения соответствуют действительности.

1. Ряд Тейлора для функции ошибок имеет вид:



Этот ряд сходится для всех *x*.

Вычислить erf(3) по указанной формуле с точно­стью до нулевого члена ряда для следующих случаев:

* Без преобразований исходной формулы;
* Сделать расчёт факториала с числом *n* целого типа,
* Сделать расчёт факториала сделать с числом *n* вещественного типа.
* Повторить расчеты, изменив формулу функции ошибок с вычислением очередного члена ряда через предыдущий член *an=kan*-1, т.е. найти *k* и таким образом избавиться от факториала.

Объяснить все четыре полученных результата. Каким знакам (цифрам) в результате erf(3) доверять нельзя? Была ли катастрофическая потеря верных знаков? В каких знаках делались округления?

Листинг программы:

<https://github.com/uncleDimasik/computational-mathematics-22-23-laba-1/blob/master/VMathLaba1.1/VMathLaba1.1/Program.cs>

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Numerics**;

**namespace** **VMathLaba1.\_2**

{

**class** **Program**

{

**static** **void** Main(string[] args)

{

Console.WriteLine("Без изменения формулы {0}", IntErrorFucn(3));

Console.WriteLine("Расчёт факториала с числом целого типа {0}", IntErrorFucn(3));

Console.WriteLine("Расчёт факториала с числом вещественного типа {0}", DoubleErrorFucn(3));

Console.WriteLine("Расчёт формулы через предыдущий член ряда {0}", ErrorFuncModify(3));

Console.ReadKey();

}

**private** **static** int Factorial(int n)

{

int factorial = 1;

**if** (n == 0)

{

**return** 1;

}

**for** (int i = 1; i <= n; i++)

{

factorial = **checked**(factorial \* i);

}

**return** factorial;

}

**private** **static** double Factorial(double n)

{

double factorial = 1;

**if** (n == 0)

{

**return** 1;

}

**for** (double i = 1; i <= n; i++)

{

factorial \*= i;

}

**return** factorial;

}

**public** **static** double DoubleErrorFucn(int x)

{

double sum = 0d;

**for** (double n = 0; ; n++)

{

double res = 0d;

var fact = Factorial(n);

**if** (fact <= 0 || double.IsInfinity(fact))

{

**break**;

}

res = Math.Pow(-1, n) \* Math.Pow(x, 2 \* n + 1) / (fact \* (2 \* n + 1));

**if** (double.IsInfinity(res) || res == 0d)

{

**break**;

}

sum += res;

}

**return** (2 / Math.Sqrt(Math.PI)) \* sum;

}

**public** **static** double IntErrorFucn(int x)

{

double sum = 0d;

**for** (int n = 0; ; n++)

{

double res = 0d;

var fact = Factorial(n);

**if** (fact >= int.MaxValue)

{

**break**;

}

**try**

{

res = **checked**(Math.Pow(-1, n) \* Math.Pow(x, 2 \* n + 1) / (fact \* (2 \* n + 1)));

}

**catch** (OverflowException)

{

**break**;

}

**if** (double.IsInfinity(res) || res == 0d)

{

**break**;

}

sum += res;

}

**return** (2 / Math.Sqrt(Math.PI)) \* sum;

}

**public** **static** double ErrorFuncModify(double x)

{

double sum = x, mult = 1, memb;

int n = 0;

*//https://ru.wikipedia.org/wiki/Функция\_ошибок*

*//https://math.stackexchange.com/questions/3694975/evaluating-erfx-using-taylors-series*

**do**

{

n++;

mult \*= -x \* x / n;

memb = x / (2 \* n + 1) \* mult;

**if** (memb == 0d)

{

**break**;

}

sum += memb;

} **while** (**true**);

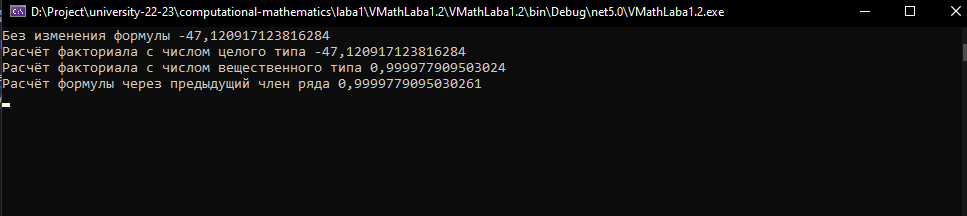
**return** (2 / Math.Sqrt(Math.PI)) \* sum;

}

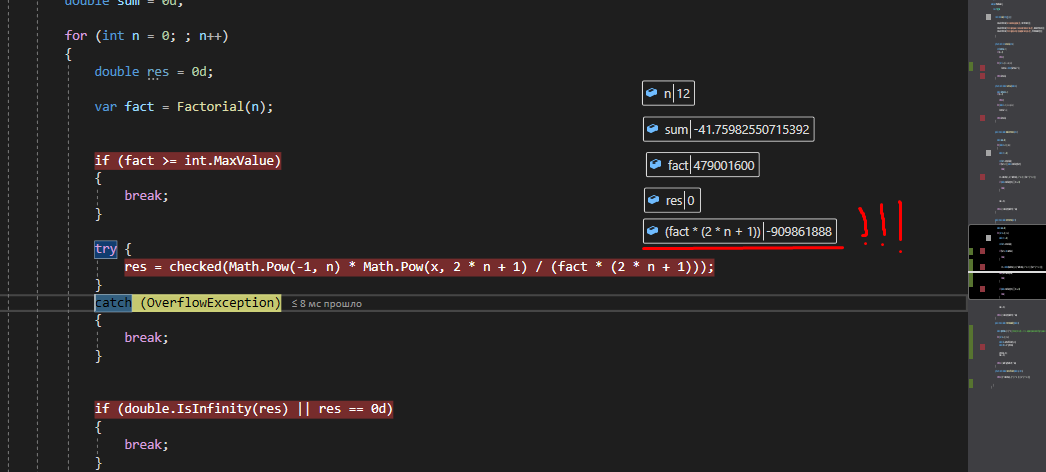
}

}

Результат работы программы:

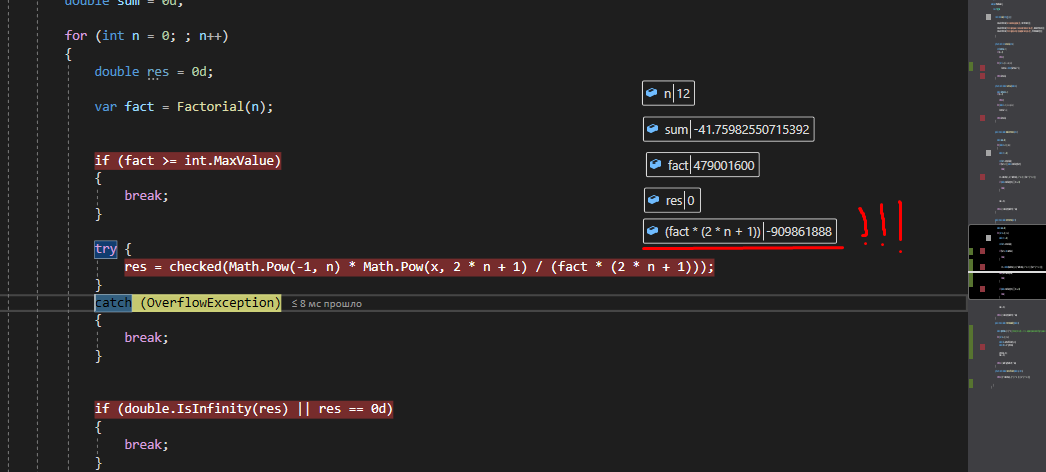


* **Без преобразований исходной формулы**



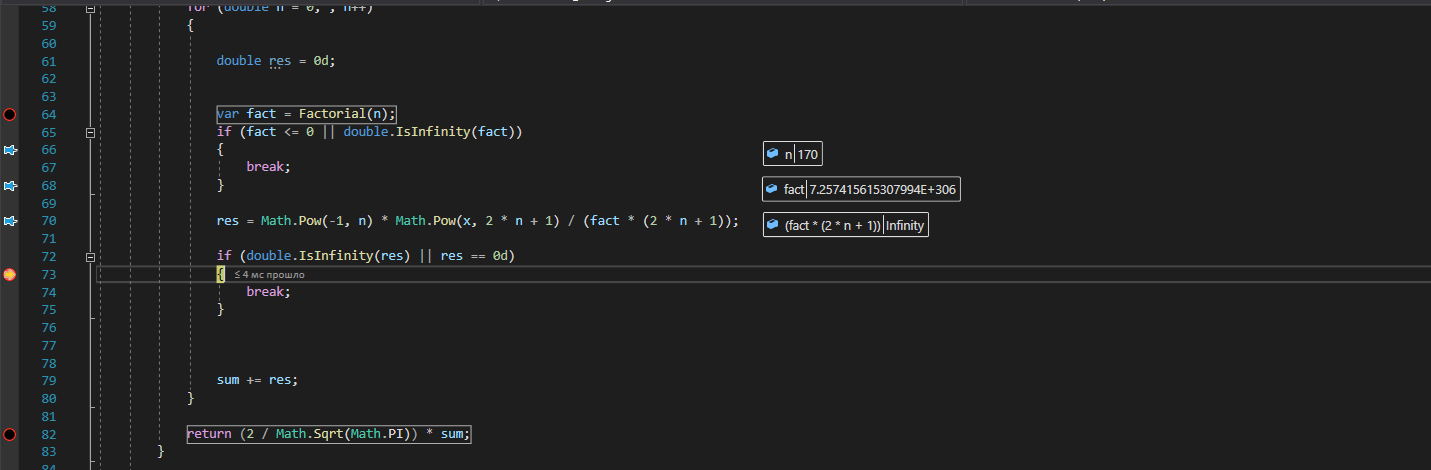
При n = 12, происходит переполнение int при вычислении знаменателя. Таким образом на n = 12 заканчивается ряд.

* **Сделать расчёт факториала с числом *n* целого типа**



При n = 12, происходит переполнение int при вычислении знаменателя. Таким образом на n = 12 заканчивается ряд.

**• Сделать расчёт факториала сделать с числом n вещественного типа**



При n = 170, значение факториала = 7.257415615307994E+306, а это близко к верхнему значению double, т.е к машинной бесконечности.



Получается что в знаменателе бесконечность, т.е мы делим на машинную бесконечность. Переменная double становится 0.

* **Повторить расчеты, изменив формулу функции ошибок с вычислением очередного члена ряда через предыдущий член *an=kan*-1, т.е. найти *k* и таким образом избавиться от факториала.**



При n = 296, значение предыдущего элемента, а это близко к машинному «0» double.



На 139-й строке программы мы делаем умножение на 0. Это значит что мы не можем считать ряд дальше. Т.к предыдущий эл. равен 7.4109846876187E-323 ≈ «0»

***Контрольные вопросы***

1. Какова причина возникновения округлений при работе с вещественными числами?

* Под их хранение выделяется ограниченное число бит. Поэтому при вычислениях с вещественными числами ошибки, связанные с неточностью представления

накапливаются. Причем, чем меньше места выделено, тем больше будет эта ошибка.

1. Что такое «катастрофическая потеря верных знаков»?

* Связана с тем, что наибольшие слагаемые по абсолютной величине на несколько порядков больше конечной суммы.

1. Что такое и как проявляют себя исчезновение и переполнение вещественных чисел?

* Когда числа по модулю подходят к своим границам. Для double на .net это



1. Почему величина дискретности вещественных чисел компьютера разная для разных порядков чисел?

* Выделяется разное кол-во байт.

1. Что такое накопление вычислительных погрешностей? Напишите пример алгоритма, накапливающего вычислительные погрешности.

* Выполнение вычислений на ЭВМ сопровождается появлением вычислительной погрешности, связанной в первую очередь с необходимостью округления результата каждой арифметической операции. Даже если разрядность ЭВМ велика, существует реальная опасность, что выполнение большого числа операций приведет к накоплению погрешности, способной значительно или даже полностью исказить вычисляемый результат.

Пример:

image1

1. Что такое чувствительность решения задачи к её параметрам, и к чему может приводить высокая чувствительность отдельных параметров?

* Когда параметры должны иметь большое кол-во значащих знаков. Это может приводить к погрешностям связанных с округлением ЭВМ.