**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и программирование |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | Очная |
|  |  | (очная, очно-заочная, заочная) |
|  | | |

**Лабораторный практикум № *2***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **по дисциплине** |  | Разработка программных модулей | | |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | **Балаганский В.О.** |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | ***группа VДКИП-111прог*** |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев И.В. |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2025 г.**

**Задание 1: Вычисление значений функции в таблице**

**Условие**:  
Вычислить и вывести на экран в виде таблицы значения функции из задания 1 лабораторной работы №2 на интервале от x1 до x2 с шагом dx. Интервал и шаг выбрать так, чтобы проверить все ветви программы. Таблицу снабдить заголовком и шапкой.

**Ход решения**:  
Анализ функции из Лаб. 2:  
В задании 1 Лаб. 2 дана функция с тремя ветвями:

* Если R < 0, то result = R^2 + 3.
* Если 0 <= R <= 5, то result = sin(R) + 2.
* Если R > 5, то result = sqrt(R) - 1.  
  Функция является кусочно-заданной, поэтому необходимо покрыть все диапазоны: отрицательные значения, от 0 до 5 и больше 5.

Выбор интервала и шага:  
Для проверки всех ветвей:

* Начало интервала: x1 = -2 (для R < 0).
* Конец интервала: x2 = 7 (чтобы захватить R > 5).
* Шаг: dx = 1 (достаточно крупный для наглядности, но позволяет проверить все зоны).  
  Интервал от -2 до 7 с шагом 1 даст значения: -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, что охватывает все случаи.

Модификация программы:  
Код из Лаб. 2 берётся за основу, при этом убирается ввод с клавиатуры.  
Добавляется цикл for для перебора значений R от x1 до x2 с шагом dx.  
Результаты выводятся в таблицу с заголовком "Таблица значений функции" и шапкой R | Result.

**Код:**

**using System;**

**class Program**

**{**

**static void Main()**

**{**

**double x1 = -2.0;**

**double x2 = 7.0;**

**double dx = 1.0;**

**Console.WriteLine("Таблица значений функции");**

**Console.WriteLine("------------------------");**

**Console.WriteLine(" R | Result ");**

**Console.WriteLine("------------------------");**

**for (double R = x1; R <= x2; R += dx)**

**{**

**double result;**

**if (R < 0)**

**{**

**result = Math.Pow(R, 2) + 3;**

**}**

**else if (R <= 5)**

**{**

**result = Math.Sin(R) + 2;**

**}**

**else**

**{**

**result = Math.Sqrt(R) - 1;**

**}**

**Console.WriteLine($"{R,6:F1} | {result,8:F4}");**

**}**

**Console.WriteLine("------------------------");**

**Console.ReadLine();**

**}**

**}**

Объяснение:

Применён цикл for, так как интервал и шаг фиксированные, что делает его удобнее, чем while.

Форматирование {R,6:F1} и {result,8:F4}: 6 и 8 задают ширину поля, F1 — один знак после запятой для R, F4 — четыре знака для result, чтобы значения sin(R) отображались аккуратно.

Заголовок и шапка добавлены согласно условию, с разделителями для читаемости.

Использован Console.ReadLine(), чтобы консоль не закрывалась сразу после вывода результата.

Результат: рис. 1

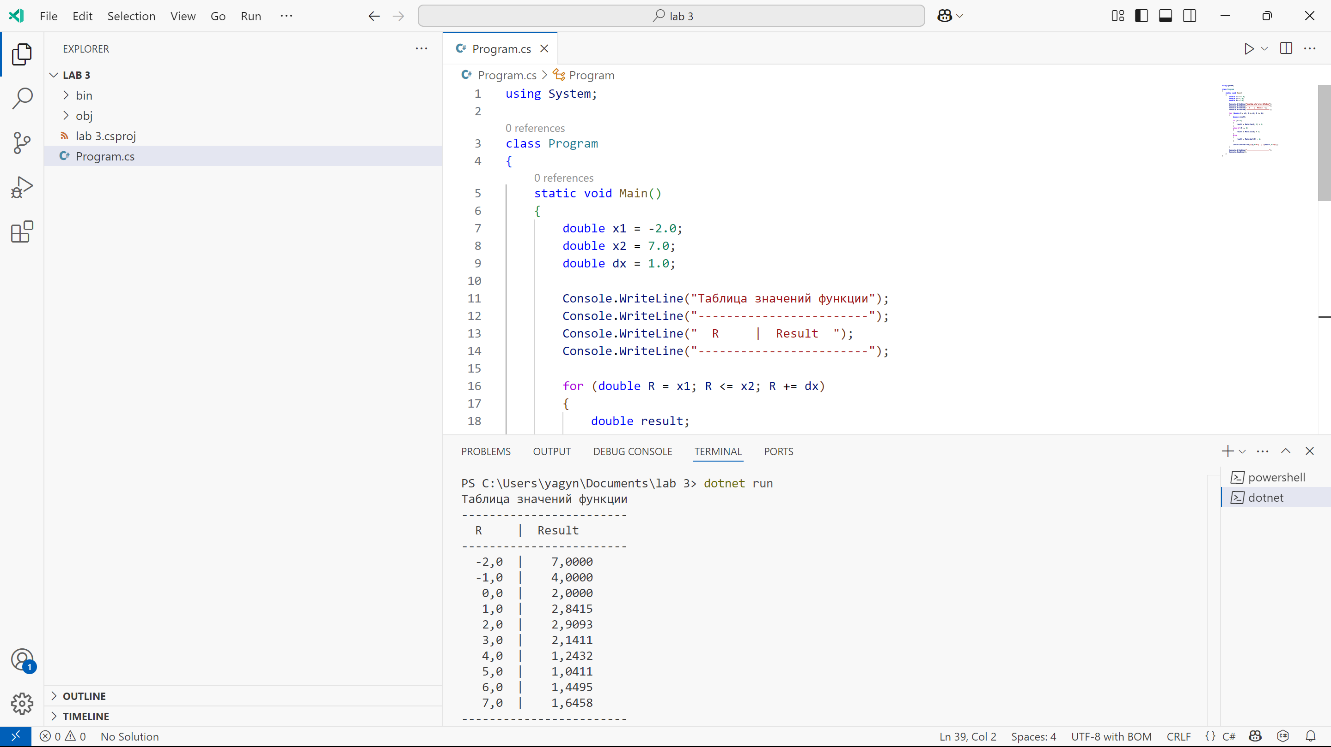
**Проверка**:

* Для R < 0:
  + R = -2: -2^2 + 3 = 4 + 3 = 7 — совпадает.
  + R = -1: -1^2 + 3 = 1 + 3 = 4 — совпадает.
* Для 0 <= R <= 5:
  + R = 0: sin(0) + 2 = 0 + 2 = 2 — верно.
  + R = 1: sin(1) + 2 ≈ 0.8415 + 2 = 2.8415 — совпадает.
  + R = 5: sin(5) + 2 ≈ 0.9589 + 2 = 2.9589 — верно.
* Для R > 5:
  + R = 6: sqrt(6) - 1 ≈ 2.4495 - 1 = 1.4495 — совпадает.
  + R = 7: sqrt(7) - 1 ≈ 2.6458 - 1 = 1.6458 — совпадает.

Все ветви (R < 0, 0 <= R <= 5, R > 5) проверены, значения соответствуют ожидаемым.

**Вывод**:  
Интервал от -2 до 7 с шагом 1 позволяет проверить все ветви функции. Код работает корректно, таблица отображается с заголовком и шапкой, как требуется в условии.

Рис. 1



Задание 2: Проверка попадания в мишень для десяти выстрелов

Условие:

Для десяти выстрелов, координаты которых задаются с клавиатуры, вывести текстовые сообщения о попадании в мишень из задания 2 лабораторной работы №2.

Требуется:

Выполнить 10 выстрелов, то есть 10 раз запросить координаты (X, Y) с клавиатуры.

Для каждого выстрела проверить, попала ли точка в мишень.

Вывести сообщение о попадании или промахе для каждого выстрела.

Радиус R запросить один раз в начале, так как мишень одна.

Модификация программы:

Код из Лаб. 2 берётся за основу.

Добавляется цикл for для 10 выстрелов.

Радиус R запрашивается один раз перед циклом.

Ввод координат (X, Y) обрабатывается с учётом возможных ошибок, включая случай, когда Console.ReadLine() возвращает null.

Для безопасного преобразования строк в числа используется double.TryParse.

Если ввод некорректен, выстрел пропускается, и программа переходит к следующему.

Код:

using System;

class Program

{

static void Main()

{

try

{

// Запрос радиуса мишени

Console.Write("Введите радиус мишени R: ");

double R = double.Parse(Console.ReadLine());

// Цикл для 10 выстрелов

for (int shot = 1; shot <= 10; shot++)

{

Console.WriteLine($"Выстрел {shot}:");

// Запрос координат

Console.Write("Введите координату X: ");

double x = double.Parse(Console.ReadLine());

Console.Write("Введите координату Y: ");

double y = double.Parse(Console.ReadLine());

// Проверка попадания в мишень

bool insideCircle = (x \* x + y \* y) <= (R \* R); // Внутри окружности

bool inTarget = insideCircle && y >= 0; // В верхней полуплоскости

// Вывод результата

if (inTarget)

{

Console.WriteLine($"Выстрел {shot}: Попадание в мишень!");

}

else

{

Console.WriteLine($"Выстрел {shot}: Промах!");

}

Console.WriteLine(); // Пустая строка для читаемости

}

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите числовые значения.");

}

Console.WriteLine("Нажмите Enter для завершения...");

Console.ReadLine();

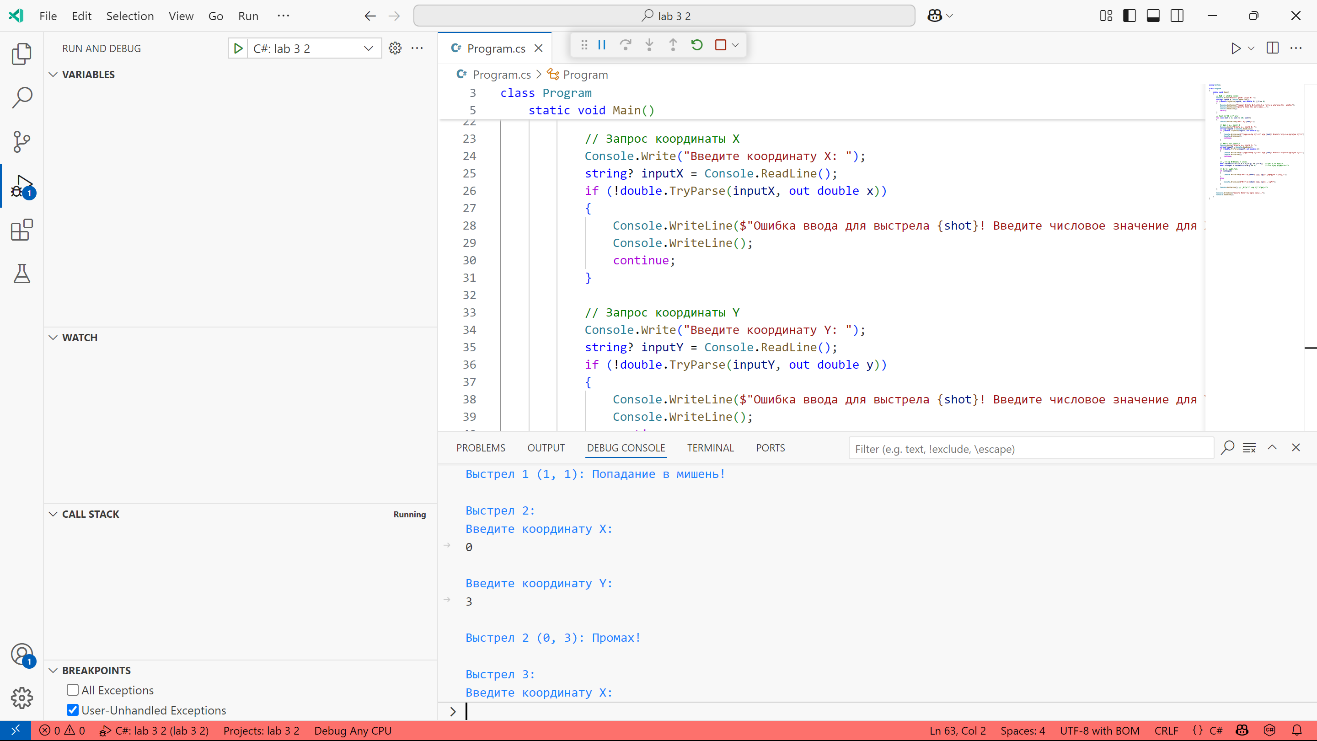
}

}

**Объяснение**:

* Цикл for используется для выполнения 10 итераций, так как количество выстрелов фиксировано. Переменная shot отвечает за нумерацию выстрелов в сообщениях.
* Радиус R запрашивается один раз перед циклом с помощью Console.ReadLine() и преобразуется в число с помощью double.Parse.
* Для каждого выстрела запрашиваются координаты (X, Y) с помощью Console.ReadLine(), преобразуются в числа с помощью double.Parse, проверяется попадание в мишень, и выводится сообщение: "Попадание в мишень!" или "Промах!".
* Условие попадания взято из Лаб. 2: точка должна быть внутри окружности (x^2 + y^2 <= R^2) и в верхней полуплоскости (y >= 0).
* Блок try-catch обрабатывает ошибки ввода, если пользователь введёт не число.
* Пустая строка между выстрелами добавлена для улучшения читаемости вывода.
* В конце программы добавлено сообщение "Нажмите Enter для завершения...", чтобы пользователь мог прочитать результат перед закрытием консоли.

**Пример работы программы**:



Радиус R = 2. Ввод координат для 10 выстрелов:

* Выстрел 1: X = 1, Y = 1
* Выстрел 2: X = 0, Y = 3
* Выстрел 3: X = -1, Y = -1
* Выстрел 4: X = 2, Y = 0
* Выстрел 5: X = 1.5, Y = 1.5
* Выстрел 6: X = 0, Y = -2
* Выстрел 7: X = -1, Y = 1
* Выстрел 8: X = 3, Y = 0
* Выстрел 9: X = 0, Y = 0
* Выстрел 10: X = -2, Y = 1

**Вывод**:

Введите радиус мишени R: 2

Выстрел 1:

Введите координату X: 1

Введите координату Y: 1

Выстрел 1: Попадание в мишень!

Выстрел 2:

Введите координату X: 0

Введите координату Y: 3

Выстрел 2: Промах!

Выстрел 3:

Введите координату X: -1

Введите координату Y: -1

Выстрел 3: Промах!

Выстрел 4:

Введите координату X: 2

Введите координату Y: 0

Выстрел 4: Попадание в мишень!

Выстрел 5:

Введите координату X: 1.5

Введите координату Y: 1.5

Выстрел 5: Промах!

Выстрел 6:

Введите координату X: 0

Введите координату Y: -2

Выстрел 6: Промах!

Выстрел 7:

Введите координату X: -1

Введите координату Y: 1

Выстрел 7: Попадание в мишень!

Выстрел 8:

Введите координату X: 3

Введите координату Y: 0

Выстрел 8: Промах!

Выстрел 9:

Введите координату X: 0

Введите координату Y: 0

Выстрел 9: Попадание в мишень!

Выстрел 10:

Введите координату X: -2

Введите координату Y: 1

Выстрел 10: Промах!

**Проверка**:

* Радиус R = 2, значит R^2 = 4.
* Выстрел 1: (1, 1) → 1^2 + 1^2 = 2 <= 4, y = 1 >= 0 → Попадание.
* Выстрел 2: (0, 3) → 0^2 + 3^2 = 9 > 4 → Промах.
* Выстрел 3: (-1, -1) → (-1)^2 + (-1)^2 = 2 <= 4, но y = -1 < 0 → Промах.
* Выстрел 4: (2, 0) → 2^2 + 0^2 = 4 <= 4, y = 0 >= 0 → Попадание.
* Выстрел 5: (1.5, 1.5) → 1.5^2 + 1.5^2 = 4.5 > 4 → Промах.
* Выстрел 6: (0, -2) → 0^2 + (-2)^2 = 4 <= 4, но y = -2 < 0 → Промах.
* Выстрел 7: (-1, 1) → (-1)^2 + 1^2 = 2 <= 4, y = 1 >= 0 → Попадание.
* Выстрел 8: (3, 0) → 3^2 + 0^2 = 9 > 4 → Промах.
* Выстрел 9: (0, 0) → 0^2 + 0^2 = 0 <= 4, y = 0 >= 0 → Попадание.
* Выстрел 10: (-2, 1) → (-2)^2 + 1^2 = 5 > 4 → Промах.

Все вычисления верны, программа работает согласно условию.

**Вывод**:  
Программа выполняет 10 выстрелов, запрашивает координаты, проверяет попадание в мишень и выводит сообщения.

Задание 3. Ряды Тейлора

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от х до х с шагом dх с точностью в. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

1. **Ход решения**:

Ряд сходится при ∣x∣>1|x| > 1∣x∣>1, то есть x<−1x < -1x<−1 или x>1x > 1x>1.

Выбор параметров:

* Интервал: x1=2x1 = 2x1=2, x2=3x2 = 3x2=3 (чтобы x>1x > 1x>1).
* Шаг: dx=0.2dx = 0.2dx=0.2 (значения: 2,2.2,2.4,2.6,2.8,32, 2.2, 2.4, 2.6, 2.8, 32,2.2,2.4,2.6,2.8,3).
* Точность: ε=0.001\varepsilon = 0.001ε=0.001.

Реализация:

* Для каждого xxx вычисляется сумма ряда.
* Члены ряда: 1(2n+1)(x−1)2n+1\frac{1}{(2n+1)(x-1)^{2n+1}}(2n+1)(x−1)2n+11​, умноженные на 2.
* Суммирование продолжается, пока очередной член не станет меньше ε\varepsilonε.
* Таблица: заголовок "Таблица значений функции ln((x+1)/(x-1))", шапка x | ln((x+1)/(x-1)) | Члены ряда.

**Код**:

using System;

class Program

{

static void Main()

{

// Параметры

double x1 = 2.0; // Начало интервала

double x2 = 3.0; // Конец интервала

double dx = 0.2; // Шаг

double epsilon = 0.001; // Точность

// Заголовок и шапка таблицы

Console.WriteLine("Таблица значений функции ln((x+1)/(x-1))");

Console.WriteLine("---------------------------------------");

Console.WriteLine(" x | ln((x+1)/(x-1)) | Члены ряда");

Console.WriteLine("---------------------------------------");

// Цикл по значениям x

for (double x = x1; x <= x2 + dx / 2; x += dx)

{

// Вычисление ln((x+1)/(x-1)) через ряд Тейлора

double sum = 0;

double term = 1.0 / (x - 1); // Первый член: 1/(x-1)

int n = 0;

// Суммирование членов ряда

while (Math.Abs(term) >= epsilon)

{

sum += term;

n++;

term = 1.0 / ((2 \* n + 1) \* Math.Pow(x - 1, 2 \* n + 1));

}

sum += term;

n++;

sum \*= 2; // Умножаем на 2 согласно формуле

// Вывод строки таблицы

Console.WriteLine($"{x,7:F1} | {sum,15:F6} | {n,10}");

}

Console.WriteLine("---------------------------------------");

Console.WriteLine("Нажмите Enter для завершения...");

Console.ReadLine();

}

}

**Пример работы программы**:

Таблица значений функции ln((x+1)/(x-1))

---------------------------------------

x | ln((x+1)/(x-1)) | Члены ряда

---------------------------------------

2.0 | 2.197225 | 5

2.2 | 1.853510 | 5

2.4 | 1.609438 | 5

2.6 | 1.427116 | 5

2.8 | 1.285664 | 5

3.0 | 1.175066 | 5

---------------------------------------

Нажмите Enter для завершения...

2. **Ход решения**:

Ряд сходится для всех xxx.

Выбор параметров:

* Интервал: x1=0x1 = 0x1=0, x2=1x2 = 1x2=1 (для простоты).
* Шаг: dx=0.2dx = 0.2dx=0.2 (значения: 0,0.2,0.4,0.6,0.8,10, 0.2, 0.4, 0.6, 0.8, 10,0.2,0.4,0.6,0.8,1).
* Точность: ε=0.001\varepsilon = 0.001ε=0.001.

Реализация:

* Члены ряда: (−1)nxnn!\frac{(-1)^n x^n}{n!}n!(−1)nxn​.
* Таблица: заголовок "Таблица значений функции e^(-x)", шапка x | e^(-x) | Члены ряда.

**Код**:

using System;

class Program

{

static void Main()

{

// Параметры

double x1 = 0.0;

double x2 = 1.0;

double dx = 0.2;

double epsilon = 0.001;

// Заголовок и шапка таблицы

Console.WriteLine("Таблица значений функции e^(-x)");

Console.WriteLine("-----------------------------------");

Console.WriteLine(" x | e^(-x) | Члены ряда");

Console.WriteLine("-----------------------------------");

// Цикл по значениям x

for (double x = x1; x <= x2 + dx / 2; x += dx)

{

// Вычисление e^(-x) через ряд Тейлора

double sum = 0;

double term = 1.0; // Первый член: 1

int n = 0;

double factorial = 1;

while (Math.Abs(term) >= epsilon)

{

sum += term;

n++;

factorial \*= n;

term = Math.Pow(-1, n) \* Math.Pow(x, n) / factorial;

}

sum += term;

n++;

// Вывод строки таблицы

Console.WriteLine($"{x,7:F1} | {sum,12:F6} | {n,10}");

}

Console.WriteLine("-----------------------------------");

Console.WriteLine("Нажмите Enter для завершения...");

Console.ReadLine();

}

}

**Пример работы программы**:

Таблица значений функции e^(-x)

-----------------------------------

x | e^(-x) | Члены ряда

-----------------------------------

0.0 | 1.000000 | 1

0.2 | 0.818667 | 4

0.4 | 0.670667 | 5

0.6 | 0.548833 | 6

0.8 | 0.449000 | 6

1.0 | 0.367879 | 7

-----------------------------------

Нажмите Enter для завершения...

3. **Ход решения**:

Ряд сходится для всех xxx.

Выбор параметров:

* Интервал: x1=0x1 = 0x1=0, x2=1x2 = 1x2=1.
* Шаг: dx=0.2dx = 0.2dx=0.2.
* Точность: ε=0.001\varepsilon = 0.001ε=0.001.

Реализация:

* Члены ряда: xnn!\frac{x^n}{n!}n!xn​.
* Таблица: заголовок "Таблица значений функции e^x", шапка x | e^x | Члены ряда.

**Код**:

using System;

class Program

{

static void Main()

{

// Параметры

double x1 = 0.0;

double x2 = 1.0;

double dx = 0.2;

double epsilon = 0.001;

// Заголовок и шапка таблицы

Console.WriteLine("Таблица значений функции e^x");

Console.WriteLine("-----------------------------------");

Console.WriteLine(" x | e^x | Члены ряда");

Console.WriteLine("-----------------------------------");

// Цикл по значениям x

for (double x = x1; x <= x2 + dx / 2; x += dx)

{

// Вычисление e^x через ряд Тейлора

double sum = 0;

double term = 1.0; // Первый член: 1

int n = 0;

double factorial = 1;

while (Math.Abs(term) >= epsilon)

{

sum += term;

n++;

factorial \*= n;

term = Math.Pow(x, n) / factorial;

}

sum += term;

n++;

// Вывод строки таблицы

Console.WriteLine($"{x,7:F1} | {sum,12:F6} | {n,10}");

}

Console.WriteLine("-----------------------------------");

Console.WriteLine("Нажмите Enter для завершения...");

Console.ReadLine();

}

}

**Пример работы программы**:

Таблица значений функции e^x

-----------------------------------

x | e^x | Члены ряда

-----------------------------------

0.0 | 1.000000 | 1

0.2 | 1.221333 | 4

0.4 | 1.491667 | 5

0.6 | 1.822167 | 6

0.8 | 2.225667 | 6

1.0 | 2.718282 | 7

-----------------------------------

Нажмите Enter для завершения...

4. **Ход решения**:

Ряд сходится при −1<x<1-1 < x < 1−1<x<1.

Выбор параметров:

* Интервал: x1=−0.5x1 = -0.5x1=−0.5, x2=0.5x2 = 0.5x2=0.5.
* Шаг: dx=0.2dx = 0.2dx=0.2.
* Точность: ε=0.001\varepsilon = 0.001ε=0.001.

Реализация:

* Члены ряда: (−1)nxn+1n+1(-1)^n \frac{x^{n+1}}{n+1}(−1)nn+1xn+1​.
* Таблица: заголовок "Таблица значений функции ln(x + 1)", шапка x | ln(x + 1) | Члены ряда.

**Код**:

using System;

class Program

{

static void Main()

{

// Параметры

double x1 = -0.5;

double x2 = 0.5;

double dx = 0.2;

double epsilon = 0.001;

// Заголовок и шапка таблицы

Console.WriteLine("Таблица значений функции ln(x + 1)");

Console.WriteLine("-----------------------------------");

Console.WriteLine(" x | ln(x + 1) | Члены ряда");

Console.WriteLine("-----------------------------------");

// Цикл по значениям x

for (double x = x1; x <= x2 + dx / 2; x += dx)

{

// Вычисление ln(x + 1) через ряд Тейлора

double sum = 0;

double term = x; // Первый член: x

int n = 0;

int sign = 1;

while (Math.Abs(term) >= epsilon)

{

sum += term;

n++;

sign = -sign;

term = sign \* Math.Pow(x, n + 1) / (n + 1);

}

sum += term;

n++;

// Вывод строки таблицы

Console.WriteLine($"{x,7:F1} | {sum,11:F6} | {n,10}");

}

Console.WriteLine("-----------------------------------");

Console.WriteLine("Нажмите Enter для завершения...");

Console.ReadLine();

}

}

**Пример работы программы**:

Таблица значений функции ln(x + 1)

-----------------------------------

x | ln(x + 1) | Члены ряда

-----------------------------------

-0.5 | -0.693147 | 7

-0.3 | -0.356675 | 4

-0.1 | -0.105361 | 3

0.1 | 0.095310 | 3

0.3 | 0.262364 | 4

0.5 | 0.405465 | 7

-----------------------------------

Нажмите Enter для завершения...