**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Инженерная школа энергетики

Отделение электроэнергетики и электротехники

Направление: 09.04.03 Прикладная информатика

Дисциплина: Программирование и основы алгоритмизации

Отчет

по лабораторной работе №3

**Цикл с параметром, цикл - Пока**

Вариант 12

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. О-5КМ41 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Рустамов Р.А. |
| Проверил: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) | Гончаров А.С. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (балл) |  |

Томск-2025

# ХОД РАБОТЫ

1.Вычислить К — количество точек с целочисленными координатами, попадающих в круг радиуса R с центром в начале координат.

В теле метода Main организован бесконечный цикл while (true), внутри которого отображается меню с четырьмя пунктами. Пользователь вводит номер пункта, проверяется корректность ввода (целое число от 1 до 4). При ошибке сообщение выводится красным цветом.

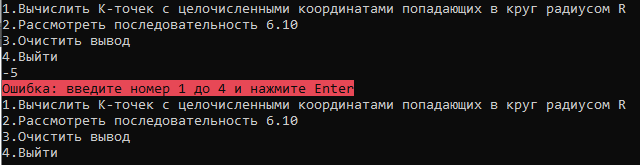


Рисунок –Вывод сообщения о неверном вводе

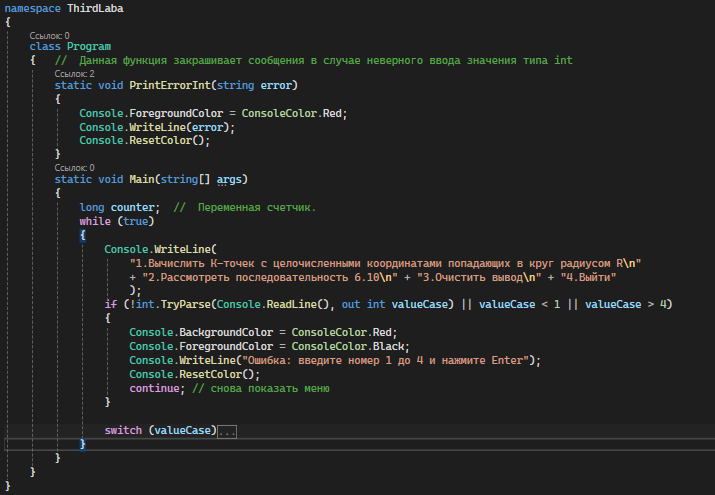


Рисунок – Основное меню

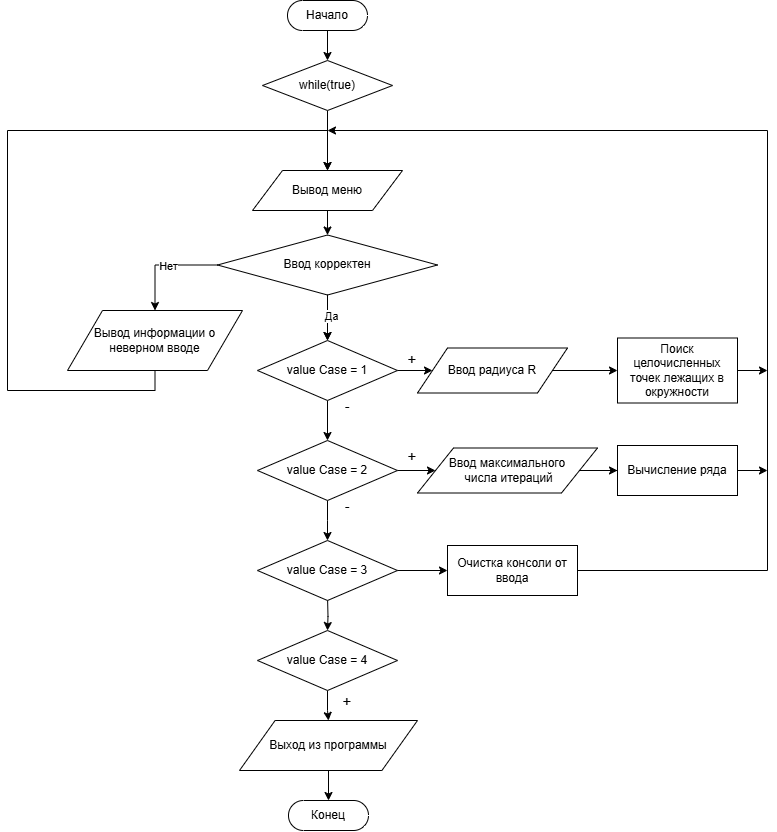


Рисунок – Блок схема функции Main

В зависимости от выбора выполняется один из четырёх сценариев:

* Case 1. Подсчёт точек в круге
  + Пользователь вводит радиус R.
  + Проверяется, что радиус — целое положительное число.
  + Программа перебирает все целочисленные координаты (x, y) в квадрате [-R, R] × [-R, R].
  + Для каждой точки проверяется условие принадлежности кругу:
  + Если условие выполняется, счётчик увеличивается на 1.
  + В конце выводится количество найденных точек.

|  |  |
| --- | --- |
|  |  |

Рисунок – Количество точек в круге радиусом R

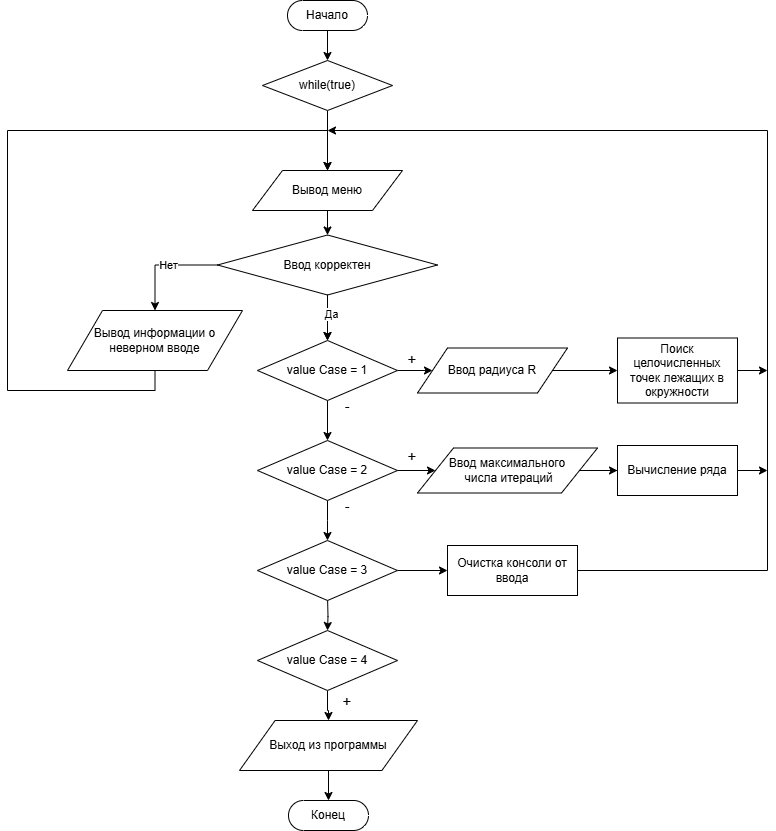


Рисунок – Блок схема расчета количества точек

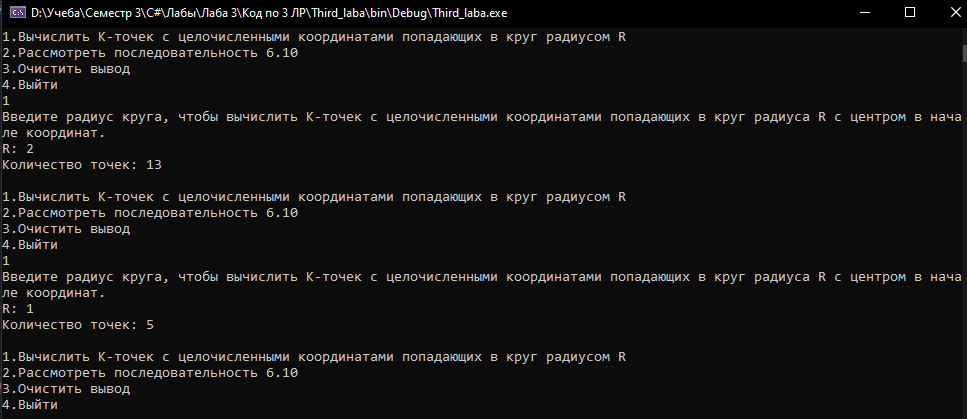


Рисунок –Пример расчета количества точек

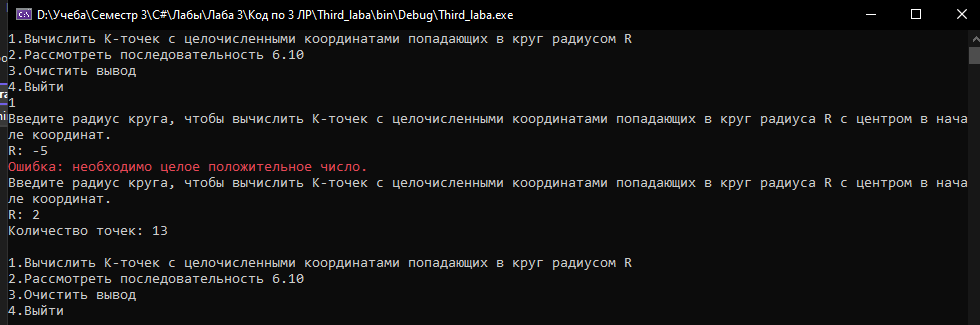


Рисунок – Пример проверки корректности ввода

* Case 2. Вычисление числовой последовательности (итерация Герона)
  + Пользователь вводит максимально допустимое число итераций.
  + Начальное значение a1 = 10.
  + Итерационный процесс:

продолжается, пока не достигнута точность или не исчерпан лимит итераций.

* + Выводится количество итераций, финальное значение a и достигнутая точность.
  + Если точность не достигнута, выводится предупреждение.

|  |  |
| --- | --- |
|  |  |

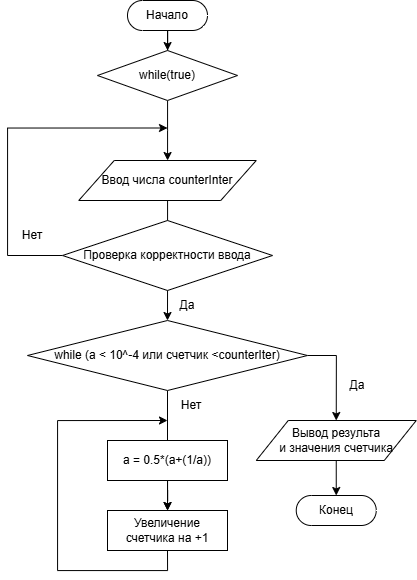


Рисунок – Блок схема расчета числового ряда Герона

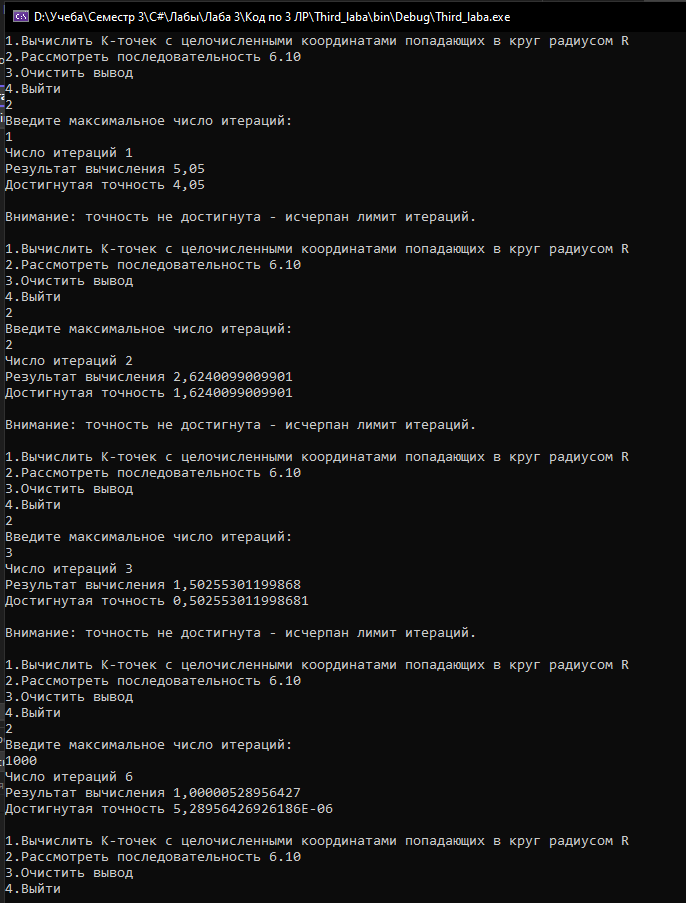


Рисунок – Пример расчета числового ряда

* Case 3. Очистка экрана
  + Выполняется команда Console.Clear().
* Case 4. Выход из программы
  + На экран выводится сообщение о завершении.
  + Программа завершается оператором return.

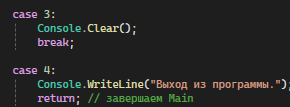


Рисунок – Очистка экрана и выход и программы

* PrintErrorInt(string error) — выводит сообщение об ошибке красным цветом, используется для валидации ввода.

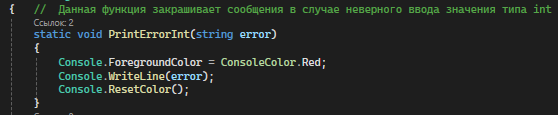


Рисунок – Вспомогательные функции

Особенности реализации программы следующие:

1. Для надёжности при подсчёте точек используется тип long, чтобы избежать переполнения счётчика;
2. Для проверки точности выбран критерий ;
3. Ввод данных защищён от ошибок с помощью int.TryParse() и long.TryParse();
4. Для удобства пользователя при некорректном вводе выводятся подсказки и программа не завершается, а возвращается к меню.

# Вывод

Разработанное приложение реализует меню с четырьмя пунктами. Программа корректно обрабатывает ввод пользователя, умеет вычислять количество точек с целыми координатами внутри круга, демонстрирует процесс сходимости последовательности по формуле Герона, очищает консоль и корректно завершает работу.

# ПРИЛОЖЕНИЕ а

using System;

namespace ThirdLaba

{

class Program

{ // Данная функция закрашивает сообщения в случае неверного ввода значения типа int

static void PrintErrorInt(string error)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(error);

Console.ResetColor();

}

static void Main(string[] args)

{

long counter; // Переменная счетчик.

while (true)

{

Console.WriteLine(

"1.Вычислить К-точек с целочисленными координатами попадающих в круг радиусом R\n"

+ "2.Рассмотреть последовательность 6.10\n" + "3.Очистить вывод\n" + "4.Выйти"

);

if (!int.TryParse(Console.ReadLine(), out int valueCase) || valueCase < 1 || valueCase > 4)

{

Console.BackgroundColor = ConsoleColor.Red;

Console.ForegroundColor = ConsoleColor.Black;

Console.WriteLine("Ошибка: введите номер 1 до 4 и нажмите Enter");

Console.ResetColor();

continue; // снова показать меню

}

switch (valueCase)

{

case 1: // Количество точек в круге радиусом R

long R;

counter = 0;

while (true)

{

Console.Write(

"Введите радиус круга, " +

"чтобы вычислить К-точек с целочисленными координатами" +

" попадающих в круг радиуса R с центром в начале координат.\n" + "R: ");

if (long.TryParse(Console.ReadLine(), out R) && R > 0)

break;

else

PrintErrorInt("Ошибка: необходимо целое положительное число.");

}

for (long x = -R; x <= R; x++)

{

for (long y = -R; y <= R; y++)

{

if (Math.Pow(x, 2) + Math.Pow(y, 2) <= Math.Pow(R, 2))

{

counter += 1;

}

}

}

Console.WriteLine($"Количество точек: {counter}\n");

break;

case 2: // Расчет числовой последовательности

int counterIter;

counter = 0;

double eps = 1e-4;

while (true)

{

Console.WriteLine("Введите максимальное число итераций:");

if (int.TryParse(Console.ReadLine(), out counterIter) && counterIter > 0)

break;

else

PrintErrorInt("Ошибка: необходимо целое положительное число.");

}

double a = 10.0;

while (eps < Math.Abs(1 - a) && counter < counterIter)

{

a = 0.5 \* (a + (1.0 / a));

counter++;

}

Console.WriteLine($"Число итераций {counter}\nРезультат вычисления {a}");

Console.WriteLine($"Достигнутая точность {Math.Abs(1 - a)}\n");

if (Math.Abs(1.0 - a) > eps)

Console.WriteLine("Внимание: точность не достигнута — исчерпан лимит итераций.\n");

break;

case 3:

Console.Clear();

break;

case 4:

Console.WriteLine("Выход из программы.");

return; // завершаем Main

}

}

}

}

}