офессиональ Федеральное государственное автономное образовательное

учреждение высшего пр ного образования

Институт транспортной на тему техники и систем управления

Кафедра «Управление и защита информации»

задание №4

по дисциплине:

«Программирование и основы алгоритмизации»

Выполнил:

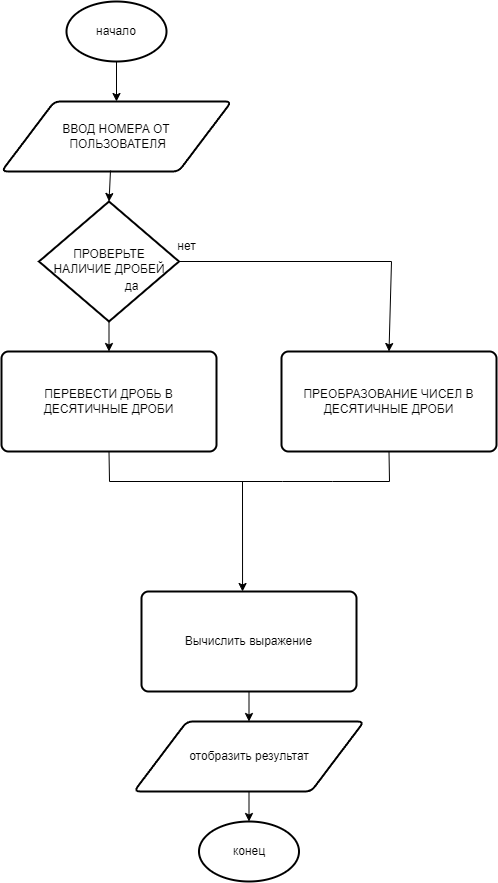
Студент группы ТЭИ-111

ДЕНГ ДЕНГ ГАК ЛУАЛ

Вариант 13

сафонов Антон игоревич

Москва 2024



1. Начало:

\* Символ: Овал

\* Пояснение: Этот символ указывает на начало выполнения программы.

2. Подсказка для ввода:

\* Символ: Параллелограмм

\* Пояснение: Этот символ представляет операцию ввода/вывода.

\* Пример кода:

Console.Write("Введите число или дробь: ");

3. Чтение входных данных:

\* Символ: Параллелограмм

\* Пояснение: Этот символ представляет операцию ввода/вывода.

\* Пример кода:

string input = Console.ReadLine();

4. Проверьте наличие фракции:

\* Символ: Бриллиант

\* Пояснение: Этот символ обозначает точку принятия решения.

\* Пример кода:

if (вход. Содержит("/"))

{

Это дробь

}

еще

{

Это десятичное число

}

5. Если дробь:

\* Символ: Прямоугольник

\* Пояснение: Этот символ представляет собой процесс или действие.

\* Пример кода:

Разнесите дробь

string[] parts = ввод. Split('/');

int numerator = int. Parse(parts[0]);

int denominator = int. Parse(parts[1]);

double value = (double)числитель / знаменатель;

6. Если не дробь:

\* Символ: Прямоугольник

\* Пояснение: Этот символ представляет собой процесс или действие.

\* Пример кода:

Разбор десятичного числа

Значение double = double. Parse(ввод);

7. Вычислить выражение:

\* Символ: Прямоугольник

\* Пояснение: Этот символ представляет собой процесс или действие.

\* Пример кода:

двойной результат = (a - b) \* c - d / (e - f) / g;

// ... другие расчеты...

8. Отображение результата:

\* Символ: Параллелограмм

\* Пояснение: Этот символ представляет операцию ввода/вывода.

\* Пример кода:

Console.WriteLine("Результат: " + результат);

9. Окончание:

\* Символ: Овал

\* Пояснение: Этот символ указывает на окончание выполнения программы.

Не забудьте адаптировать примеры кода к вашим конкретным требованиям и сложности вычислений. Эта блок-схема и объяснение кода обеспечивают общую основу для понимания хода и реализации программы

using System;

using System.Text.RegularExpressions;

public class FormulaCalculator

{

public static void Main(string[] args)

{

double a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p;

try

{

// Get user input, handling fractions

a = GetFractionInput("a (2.7): ");

b = GetFractionInput("b (0.8): ");

c = GetFractionInput("c (2): ");

d = GetFractionInput("d (1/3): ");

e = GetFractionInput("e (5.2): ");

f = GetFractionInput("f (1.4): ");

g = GetFractionInput("g (7): ");

h = GetFractionInput("h (2.625): ");

i = GetFractionInput("i (8): ");

j = GetFractionInput("j (9): ");

k = GetFractionInput("k (11): ");

l = GetFractionInput("l (95): ");

m = GetFractionInput("m (7): ");

n = GetFractionInput("n (30): ");

o = GetFractionInput("o (93): ");

p = GetFractionInput("p (5/18): ");

// Calculate the expression

double result = CalculateExpression(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p);

Console.WriteLine($"Result: {result}");

}

catch (FormatException)

{

Console.WriteLine("Invalid input format. Please use numbers or fractions (numerator/denominator).");

}

catch (Exception ex)

{

Console.WriteLine($"An error occurred: {ex.Message}");

}

}

// Helper function to parse fractions

static double GetFractionInput(string prompt)

{

Console.Write(prompt);

string input = Console.ReadLine();

// Check for mixed fractions (e.g., 1 2/3)

Match matchMixed = Regex.Match(input, @"(\d+)\s+(\d+)/(\d+)");

if (matchMixed.Success)

{

int whole = int.Parse(matchMixed.Groups[1].Value);

int num = int.Parse(matchMixed.Groups[2].Value);

int den = int.Parse(matchMixed.Groups[3].Value);

return whole + (double)num / den;

}

// Check for simple fractions (e.g., 1/3)

Match matchSimple = Regex.Match(input, @"(\d+)/(\d+)");

if (matchSimple.Success)

{

int num = int.Parse(matchSimple.Groups[1].Value);

int den = int.Parse(matchSimple.Groups[2].Value);

return (double)num / den;

}

// If not a fraction, parse as a double

return double.Parse(input);

}

// Calculate the expression, combining the best aspects of both methods

static double CalculateExpression(

double a, double b, double c, double d, double e, double f, double g, double h, double i, double j, double k, double l, double m, double n, double o, double p)

{

// Break down the expression into smaller sub-expressions for readability and potential optimization

double numerator = (a - b) \* c - d;

double denominator = (e - f) / g;

double sqrt1 = Math.Sqrt(numerator / denominator);

double sqrt2 = Math.Sqrt(h);

double product = i \* j / k;

double division = m / (n - o - p);

double sum = l + division;

double subtraction = sum \* 2 - 1.0 / 4.0;

double sqrt3 = Math.Sqrt(6 \* Math.PI);

return 5 \* sqrt1 - sqrt2 + product - subtraction + sqrt3;

}

}

заключение по проделанной работе:

Код представляет собой функциональное и хорошо структурированное решение для вычисления конкретной формулы с дробным вводом. Однако улучшения в обработке ошибок, гибкость (возможность создания формул, определяемых пользователем) и упрощение кода (уменьшение количества переменных) сделали бы его более надежным, удобным для пользователя и сопровождаемым. Добавление таких функций, как обработка приоритета операторов, сделало бы его значительно более мощным в качестве универсального калькулятора формул.