МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ
Государственное автономное профессиональное образовательное учреждение
«ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»
(ГАПОУ ОКЭИ)

Практическое задание

ОКЭИ 09.02.07 4025 23 ПЗ

Тема: «*Разработка динамических библиотек DLL*»

Выполнил: Яценко Алексей Александрович

Оренбург 2025

Содержание

Введение

Данная практическая работа посвящена разработке динамической библиотеки (DLL) на языке C#, инкапсулирующей функционал консольного калькулятора. Основная цель проекта — освоение принципов создания и использования динамически подключаемых библиотек, а также закрепление навыков модульного программирования.

Библиотека будет предоставлять основные арифметические операции (сложение, вычитание, умножение, деление), а также более сложные математические функции: возведение в степень, вычисление факториала и решение квадратных уравнений. Реализация будет выполнена в виде консольного приложения, которое использует разработанную DLL, что наглядно демонстрирует преимущества разделения кода на логические модули и их повторного использования.

1 Ход работы

1.1 Создание DLL библиотеки

Данный код представляет собой динамическую библиотеку (DLL), содержащую класс Calculator с набором статических методов для выполнения математических операций. Библиотека включает как базовые арифметические функции, так и сложные математические вычисления.

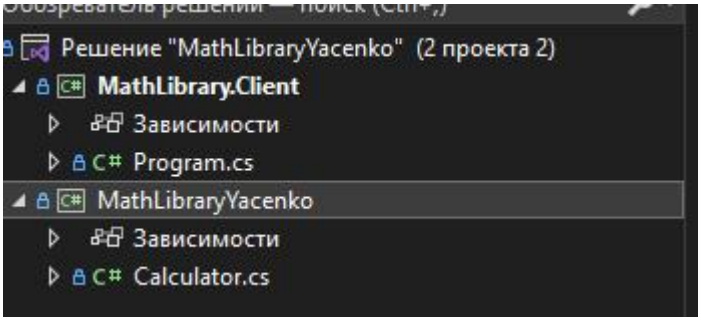


Рисунок 1 - Общая структура проекта в Solution Explorer

1.2 Базовые арифметические операции

Библиотека реализует стандартные арифметические операции:

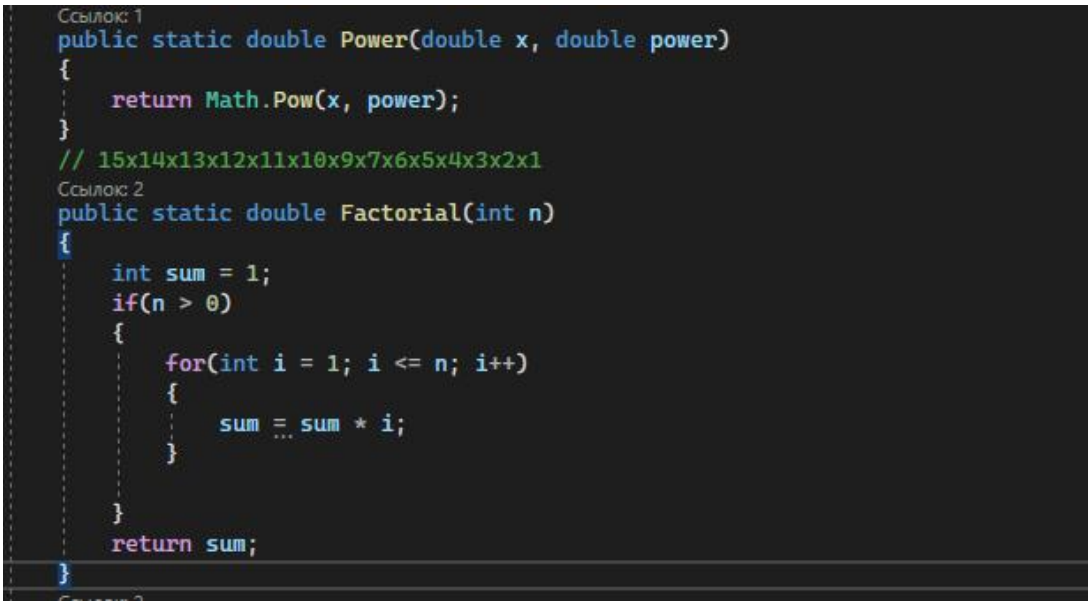
- Сложение (Add) - возвращает сумму двух чисел;
- Вычитание (Substract) - возвращает разность между вторым и первым аргументом;
- Умножение (Multiply) - возвращает произведение двух чисел;
- Деление (Divide) - включает проверку деления на ноль.

```
public static double Add(double x, double y)
{
    return x + y;
}


public static double Substract (double x, double y)
{
    return y - x;
}


public static double Multiply(double x, double y)
{
    return (x * y);
}


public static double Divide(double x, double y)
{
    if (y == 0)
    {
        Console.WriteLine("Ошибка: " + new DivideByZeroException("На ноль делить нельзя"));
    }

    return x / y;
}
```

Рисунок 2 - Базовые операции

1.3 Дополнительные математические функции

Метод Power использует встроенную функцию Math.Pow для вычисления степени числа. Метод Factorial вычисляет факториал целого числа с использованием цикла.

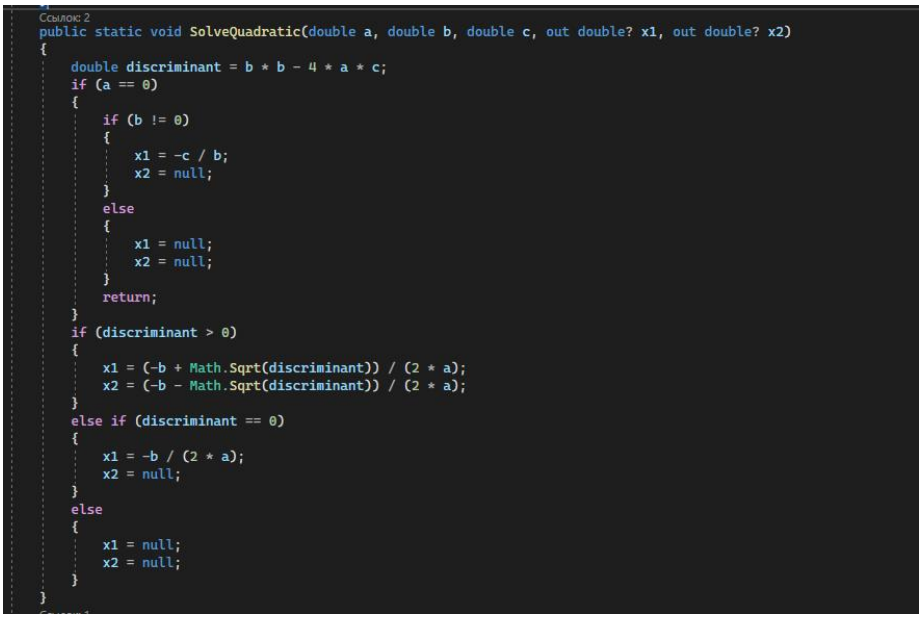


```
Ссылок: 1
public static double Power(double x, double power)
{
    return Math.Pow(x, power);
}
// 15x14x13x12x11x10x9x7x6x5x4x3x2x1
Ссылок: 2
public static double Factorial(int n)
{
    int sum = 1;
    if(n > 0)
    {
        for(int i = 1; i <= n; i++)
        {
            sum = sum * i;
        }

    }
    return sum;
}
```

Рисунок 3 - Функции Power и Factorial.

Метод SolveQuadratic решает квадратные уравнения вида $ax^2 + bx + c = 0$:

- Вычисляет дискриминант;
- Обрабатывает особые случаи (когда a = 0);
- Возвращает корни через выходные параметры;
- Использует nullable типы для обозначения отсутствия корней.

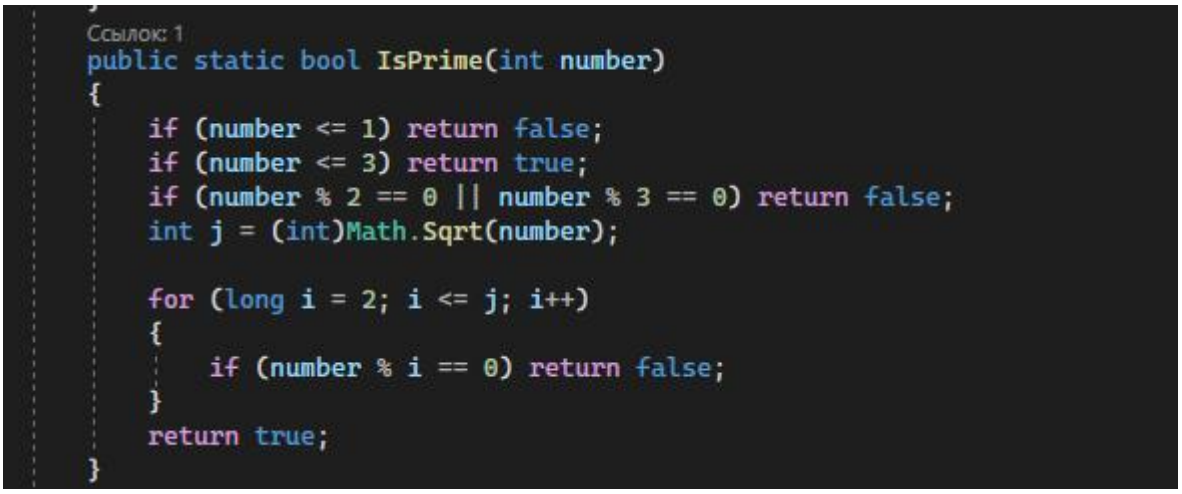


```
Ссылок: 2
public static void SolveQuadratic(double a, double b, double c, out double? x1, out double? x2)
{
    double discriminant = b * b - 4 * a * c;
    if (a == 0)
    {
        if (b != 0)
        {
            x1 = -c / b;
            x2 = null;
        }
        else
        {
            x1 = null;
            x2 = null;
        }
        return;
    }
    if (discriminant > 0)
    {
        x1 = (-b + Math.Sqrt(discriminant)) / (2 * a);
        x2 = (-b - Math.Sqrt(discriminant)) / (2 * a);
    }
    else if (discriminant == 0)
    {
        x1 = -b / (2 * a);
        x2 = null;
    }
    else
    {
        x1 = null;
        x2 = null;
    }
}
```

Рисунок 4 - Функция SolveQuadratic

Метод IsPrime определяет, является ли число простым:

- Использует оптимизации: проверка делимости на 2 и 3;
- Проверяет делители только до квадратного корня числа;
- Возвращает логическое значение.



```
Ссылок: 1
public static bool IsPrime(int number)
{
    if (number <= 1) return false;
    if (number <= 3) return true;
    if (number % 2 == 0 || number % 3 == 0) return false;
    int j = (int)Math.Sqrt(number);

    for (long i = 2; i <= j; i++)
    {
        if (number % i == 0) return false;
    }
    return true;
}
```

Рисунок 5 - Функция проверки простых чисел.

1.4 Архитектурные особенности

У проекта есть следующие архитектурные особенности:

- Все методы объявлены как static, что позволяет вызывать их без создания экземпляра класса;
- Использование nullable типов для возврата optional значений;
- Обработка исключительных ситуаций (деление на ноль);

2 Демонстрация работы проекта

На рисунках 6,7 представлен код Program.cs, который поочередно вызывает все методы и проверяет их работоспособность с выводом текстовых сообщений.

```
Console.WriteLine("Демонстрация работы MathLibrary.dll");

double x = 10, y = 4;

Console.Write($"Сложение чисел {x} + {y} = ");
Console.WriteLine(Calculator.Add(x, y));

Console.Write($"Вычитание второго числа из первого {y} - {x} = ");
Console.WriteLine(Calculator.Substract(x, y));

Console.Write($"Умножение чисел {x} * {y} = ");
Console.WriteLine(Calculator.Multiply(x, y));

Console.Write($"Деление чисел без ошибки {x} / {y} = ");
Console.WriteLine(Calculator.Divide(x, y));

double a = 2, b = 0;

Console.Write($"Деление чисел без ошибки {a} / {b} = ");
Console.WriteLine(Calculator.Divide(a, b));

Console.WriteLine("Проверка чисел на простоту");

int[] numbersToCheck = { 1, 2, 3, 4, 17, 25, 97, 997};

for (int i = 0; i < numbersToCheck.Length; i++)
{
    int num = numbersToCheck[i];
    bool isPrime = Calculator.IsPrime(num);
    Console.WriteLine($"Число {num} является простым? -> {isPrime}");
}
```

Рисунок 6 - Первая часть методов

```
int z = 6, power = 3;
Console.Write($"Возведение {x} в {power} степень числа = ");
Console.WriteLine(Calculator.Power(z, power));

int q = 12;
Console.Write($"Факториал {q} = ");
Console.WriteLine(Calculator.Factorial(q));

Console.Write($"Факториал {q} = ");
Console.WriteLine(Calculator.Factorial(q));

double? x1, x2;
Calculator.SolveQuadratic(1, -4, 4, out x1, out x2);
Console.WriteLine($"\nУравнение: x² - 4x + 4 = 0");
Console.WriteLine($"Корни: x1 = {x1}, x2 = {x2}");
// Вывод: Корни: x1 = 2, x2 = null


Calculator.SolveQuadratic(1, -3, 2, out x1, out x2);
Console.WriteLine($"Уравнение: x² - 3x + 2 = 0");
Console.WriteLine($"Корни: x1 = {x1}, x2 = {x2}");
// Вывод: Корни: x1 = 2, x2 = 1
```

Рисунок 7 – Вторая часть методов

На рисунке 7 представлена демонстрация работы запущенного проекта со всеми пояснительными сообщениями.

```
Демонстрация работы MathLibrary.dll
Сложение чисел 10 + 4 = 14
Вычитание второго числа из первого 4 - 10 = -6
Умножение чисел 10 * 4 = 40
Деление чисел без ошибки 10 / 4 = 2,5
Деление чисел без ошибки 2 / 0 = Ошибка: System.DivideByZeroException: На ноль делить нельзя
?
Проверка чисел на простоту
Число 1 является простым? -> False
Число 2 является простым? -> True
Число 3 является простым? -> True
Число 4 является простым? -> False
Число 17 является простым? -> True
Число 25 является простым? -> False
Число 97 является простым? -> True
Число 997 является простым? -> True
Возведение 10 в 3 степень числа = 216
Факториал 12 = 479001600
Факториал 12 = 479001600

Уравнение: x? - 4x + 4 = 0
Корни: x1 = 2, x2 =
Уравнение: x? - 3x + 2 = 0
Корни: x1 = 2, x2 = 1

D:\Users\yatsenko.aa2201\source\repos\MathLibraryYacenko\MathLibrary.Client\bin\Debug\net6.0\
цесс 14124) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Па
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 8 - Демонстрация работы проекта