**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ** **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** **«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ** **“СИНЕРГИЯ”»**

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

**Отчет по практической работе №1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **на тему** |  | **Тестирование программного обеспечения** | | |
|  |  | (наименование темы) | | |
|  |  |  | | |
| **по дисциплине** | | |  | **С#** |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | Воропаев Николай Сергеевич |  | Не требуется |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | Дкип-111прог |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  |  |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2024 г.**

**Отладка проекта**

**Цель занятия**: закрепление практических навыки работы с системой Visual Studio Code; научиться использовать инструментальные средства, помогающие провести отладку приложений. Научиться составлять техническое задание (ТЗ) на разработку программного продукта, применять отладочные классы в проекте.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования Visual Studio Code.

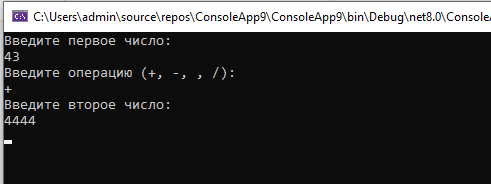
**Продолжительность занятия:**4 часа.

**Задание № 1**

Вариант 1

**Задание:**

1. Создание программы.
2. Проведите отладку программы всеми доступными вам средствами среды разработки.
3. Для выбранного программного продукта разработать техническое задание в соответствии с ГОСТ 19.201-78, предполагая, что сначала разрабатывается ТЗ, а затем будет написана программа для ТЗ.
4. Отчет по лабораторной работе должен содержать Скриншоты выполнения Задания 1, разделы технического задания, Задание№2 , Задание №3 или Задание №4



Калькулятор выполняет все базовые задачи!!

using System;

public class Calculator

{

public static void Main(string[] args)

{

Console.WriteLine("Введите первое число:");

double num1 = double.Parse(Console.ReadLine());

Console.WriteLine("Введите операцию (+, -, , /):");

string op = Console.ReadLine();

Console.WriteLine("Введите второе число:");

double num2 = double.Parse(Console.ReadLine());

double result = 0;

bool error = false;

switch (op)

{

case "+":

result = num1 + num2;

break;

case "-":

result = num1 - num2;

break;

case "":

result = num1 \* num2;

break;

case "/":

if (num2 == 0)

{

Console.WriteLine("Ошибка: Деление на ноль!");

error = true;

}

else

{

result = num1 / num2;

}

break;

default:

Console.WriteLine("Ошибка: Неверная операция!");

error = true;

break;

}

if (!error)

{

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

}

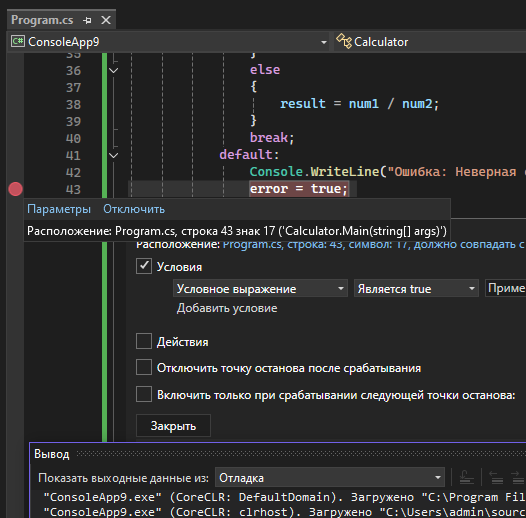
Console.ReadKey(); // Задерживает консоль, чтобы увидеть результат

}

}

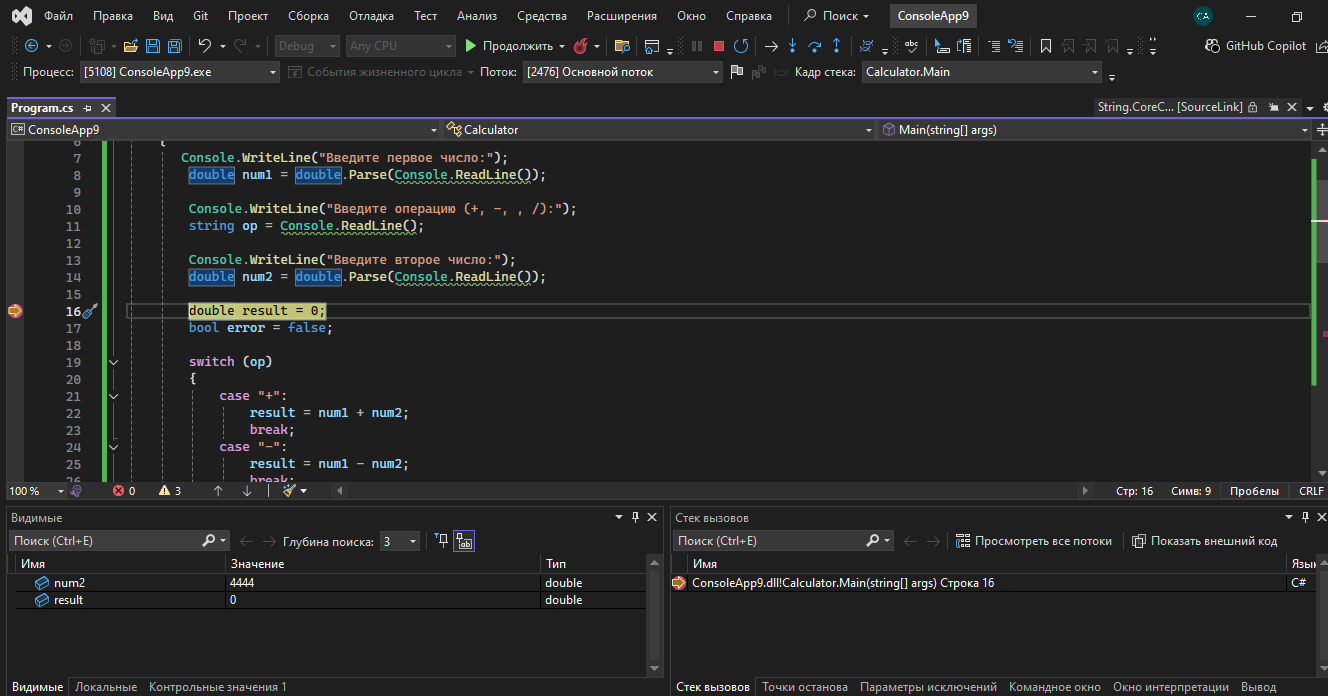


**Инструменты отладки в Visual Studio**



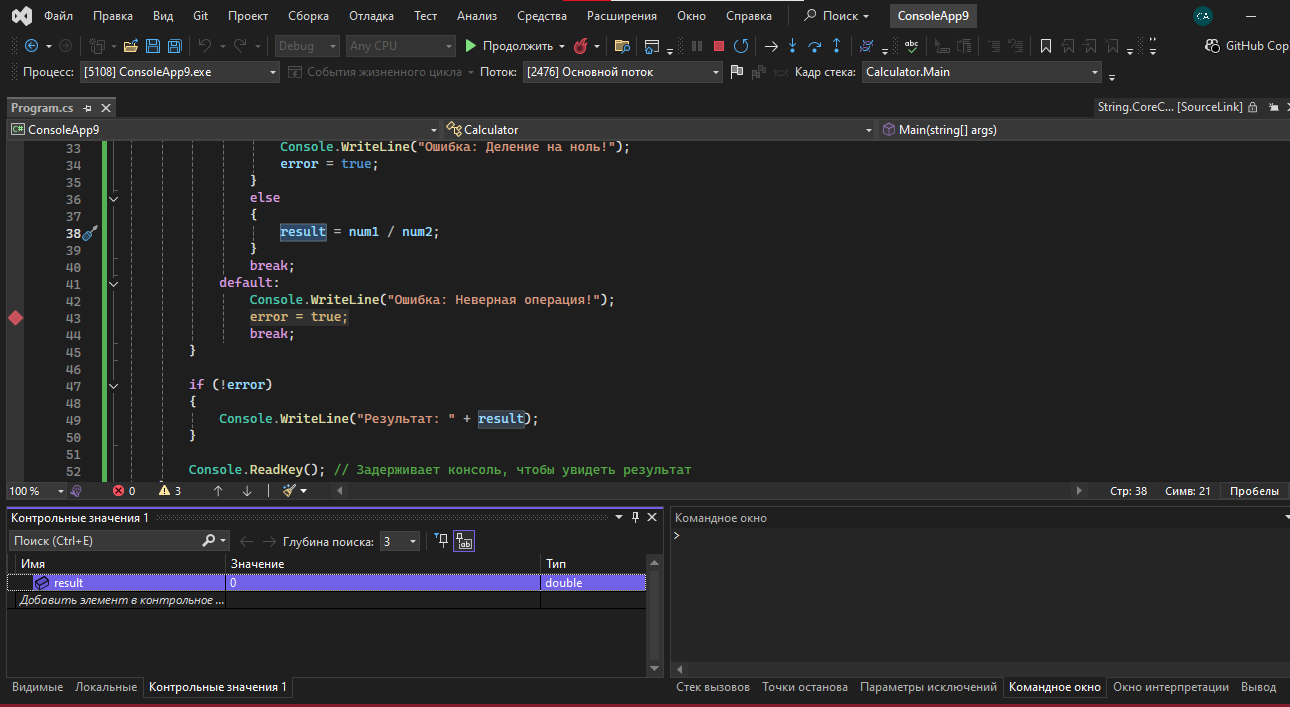
**Рис 1. Условные брейкпойтнты**

На С# панель тоже находиться внизу(там можно добавлять условия или действия)



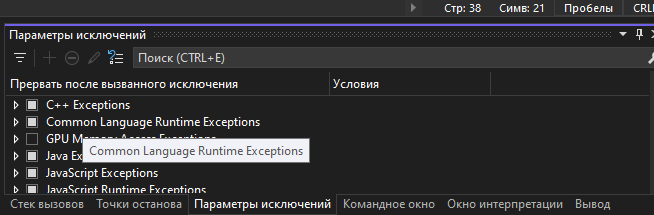
**Рис 2.** **Просмотр переменных и выражений**

Позволяет добавлять переменные и выражения, за которыми необходимо следить в процессе выполнения программы. Значения этих переменных и выражений автоматически обновляются при каждом шаге отладки.

****

**Рис 3. Консоль отладки**

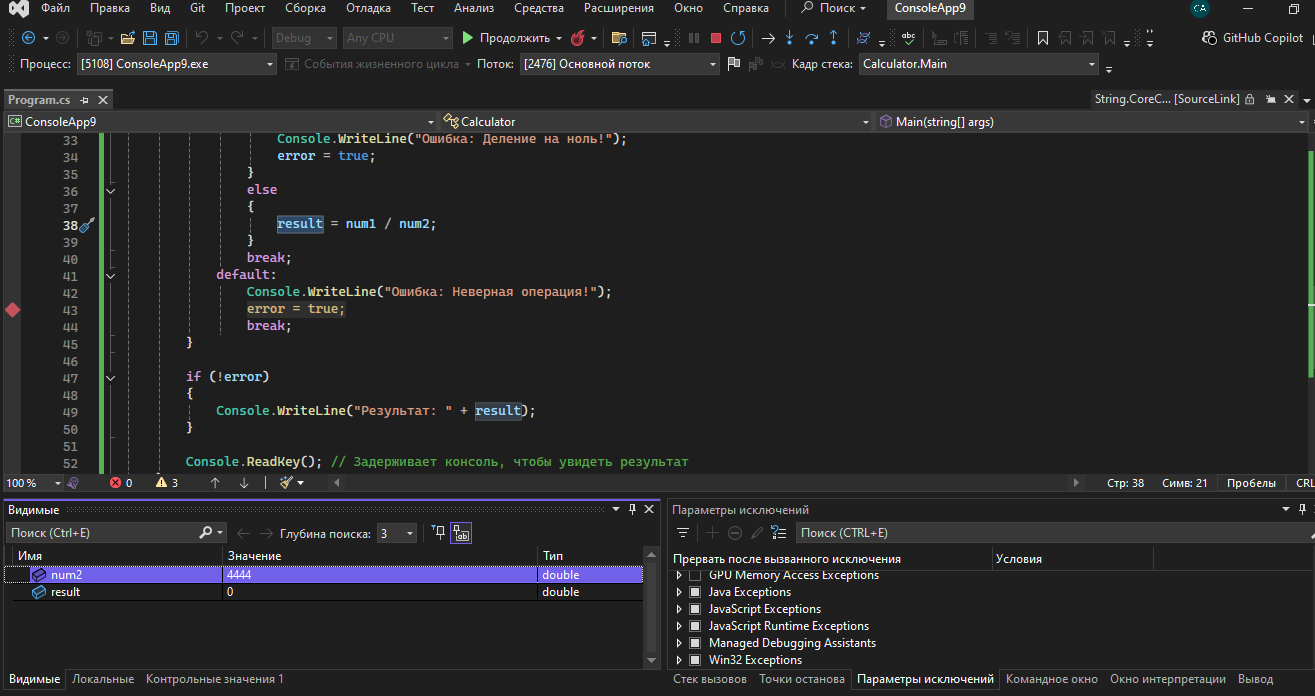
Позволяет вручную вводить команды и выражения для проверки их работы во время остановки программы.

****

**Рис4. Исключения (Exceptions)**

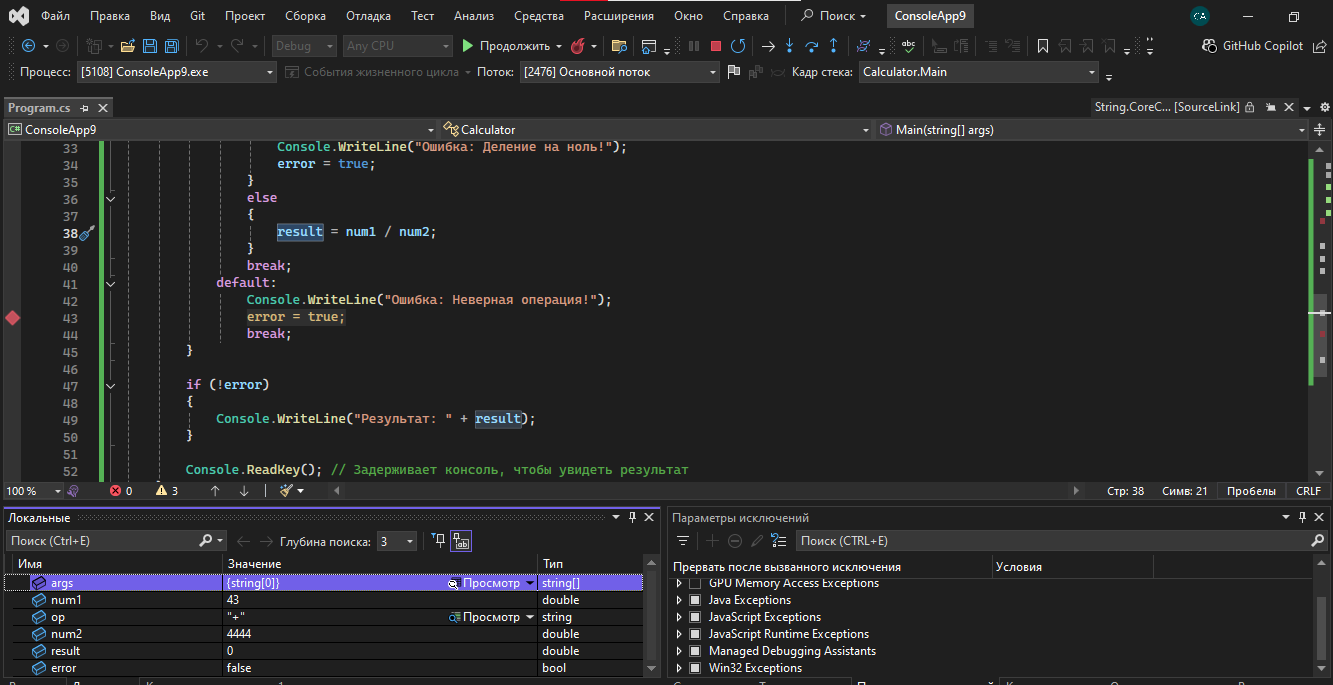
Исключение – это событие, которое возникает во время выполнения программы и сигнализирует о наличии ошибки или исключительной ситуации.

Code отладчик может автоматически приостановить выполнение программы при возникновении исключения, позволяя быстро определить и устранить ошибку

****

**Рис 5. Видимые переменные**

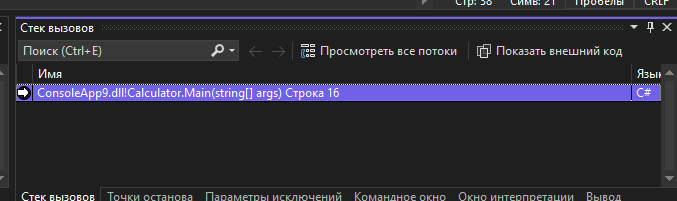
Показывает данные о переменных которые на данный момент активны



**Рис 6. Локальные**

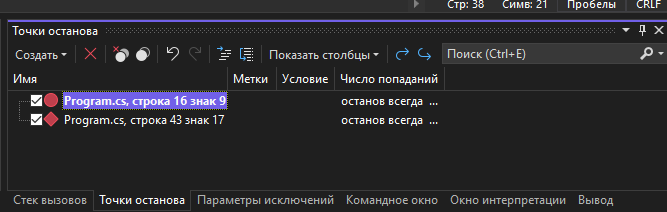
Находятся в рамках функции

Не могут выходить за пределы указанной функции!



**Рис 7. Стек вызовов**

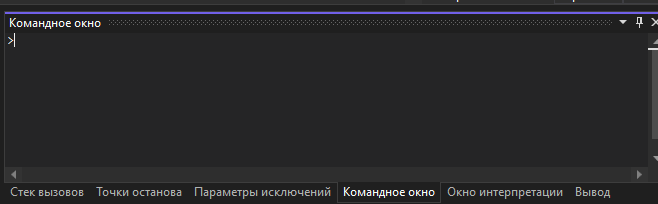
Переменные которые вызваны



**Рис 8. точки останова**

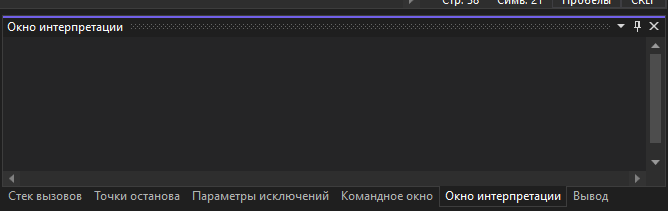
Приостановка программы

Приостанавливает программу для дальнейшего изучения!



**Рис 9. командное окно**

Служит для дополнения программы



**Рис 10. Окно интерпретации**

**Окно интерпретации** в Visual Studio используется для **отладки и вычисления выражений**, выполнения операторов и печати значений переменных

**Введение**

Этотдокумент описывает спецификацию для разработки простого калькулятора. Калькулятор будет выполнять базовые арифметические операции: сложение, вычитание, умножение и деление

1.1. Наименование программы:

“Программа "Калькулятор''

1.2. Назначение и область применения программы:

Программа предназначена для выполнения арифметических операций сложения, вычитания, умножения и деления над двумя вещественными числами. Программа ориентирована на использование отдельными пользователями на персональных компьютерах.

**2. Требования к программе**

Калькулятор будет выполнять базовые арифметические операции: сложение, вычитание, умножение и деление

2.1. Требования к функциональным характеристикам программы:

Программа должна запрашивать у пользователя два вещественных числа.

Программа должна запрашивать у пользователя арифметическую операцию (+, -, , /).

Программа должна выполнять выбранную арифметическую операцию над введенными числами.

Программа должна выводить результат операции на экран.

Программа должна обрабатывать ошибку деления на ноль, вывода сообщения об ошибке и прекращения вычисления.

Программа должна обрабатывать ошибку ввода некорректной операции, выводя сообщение об ошибке.

**2.2. Требования к надежности программы:**

2.2.1. Требования к обеспечению надежного функционирования программы:

Программа должна корректно работать при вводе корректных данных. Должна быть устойчива к ошибкам ввода некорректных данных в пределах допустимых типов данных.

2.2.2. Время восстановления программы после отказа:

Не применимо. Отсутствие отказов при корректном вводе.

2.2.3. Отказы программы из-за некоректных действий оператора:

Программа должна обрабатывать некорректный ввод чисел (например, букв вместо чисел) с выводом сообщения об ошибке и запросом на повторный ввод. Некорректный ввод операции должен также обрабатываться с выводом сообщения об ошибке.

**3. Условия эксплуатации программы**

3.1. Климатические условия эксплуатации программы:

Не предъявляются специфические требования. Программа предназначена для работы в стандартных условиях эксплуатации персонального компьютера.

3.2. Требования к квалификации и численности персонала:

Пользователь должен обладать базовыми навыками работы с персональным компьютером.

3.3. Требования к составу и параметрам технических средств:

Персональный компьютер с операционной системой, поддерживающей среду выполнения .NET Framework.

3.4. Требования к информационной совместимости:

3.4.1. Требования к информационным структурам и методам решения:

Программа использует вещественные числа в качестве входных и выходных данных. Алгоритмы вычислений - стандартные арифметические операции.

3.4.2. Требования к исходным кодам и языкам программирования:

Язык программирования C#.

3.4.3. Требования к программным средствам, используемым

программой: Среда разработки .NET Framework.

3.4.4. Требования к защите информации и программ:

Не предъявляются специфические требования.

3.5. Специальные требования:

Программа должна быть простой в использовании и иметь интуитивно понятный интерфейс командной строки.

**4. Требования к программной документации:**

* 1. Предварительный состав программной документации:

Техническое задание, исходный код программы, руководство пользователя.

**5. Технико-экономические показатели**

5.1. Экономические преимущества разработки программы:

Низкая стоимость разработки и простота в использовании.

**6. Стадии и этапы разработки программы**

6.1. Стадии разработки программы:

Установка и настройка среды разработки

Определение алгоритма действий

Добавление функции

Запуск программы

6.2. Этапы разработки программы:

Этапы разработки будут соответствовать стандартному циклу разработки программного обеспечения (водопадная модель).

6.3. Содержание работ по этапам: На каждом этапе будут выполняться соответствующие работы по проектированию, кодированию, тестированию и документации.

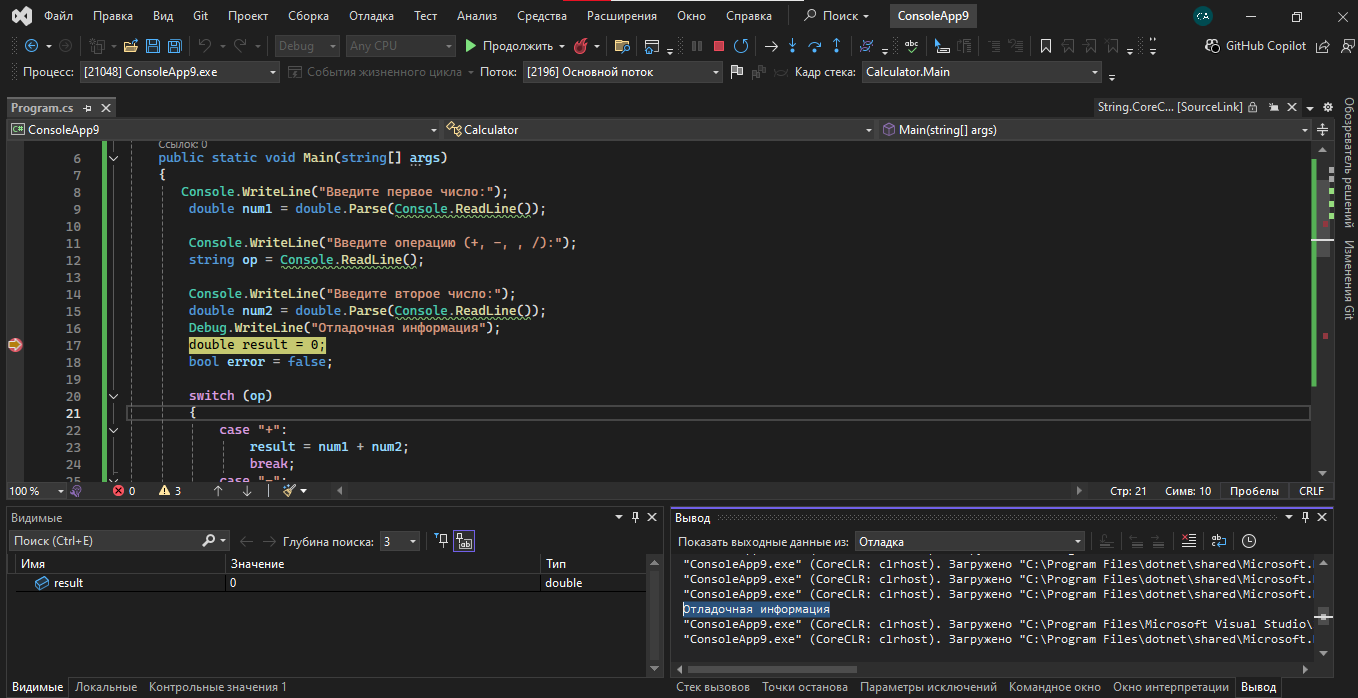
**7. Порядок контроля и приемки**

7.1. Виды испытаний:

Модульное тестирование, интеграционное тестирование, приемочные испытания.

7.2. Общие требования к приемке работы: Программа должна соответствовать требованиям, указанным в настоящем техническом задании. Результаты тестирования должны подтвердить корректность работы программы.

## Отладочные классы в C#:



**Рис 11. отладочная информация**

Выявление ошибки в логике программы

**Контрольные вопросы:**

1. **Что такое отладка?**

Процесс устранение ошибок

1. **Какие инструменты отладки вам известны?**

Инструменты отладки сильно зависят от используемого языка программирования и среды разработки. Вот некоторые примеры:

Интегрированные отладчики (IDE debuggers)

3. **Методы отладки.**

Пошаговое выполнение (Stepping):

Выполнение кода построчно, позволяющее отслеживать изменения значений переменных и контролировать поток выполнения.

Точки останова (Breakpoints):

Устанавливаются в определённых местах кода, чтобы приостановить выполнение программы и исследовать состояние в этом месте.

1. **Назначение технического задания?**

Техническое задание (ТЗ) — это документ, который описывает цели, требования и ограничения для разработки программного обеспечения или другого технического продукта. Он служит основой для дальнейшей разработки, обеспечивает понимание между заказчиком и разработчиком и позволяет оценить затраты, сроки и ресурсы.

1. **Кто составляет и утверждает ТЗ?**

ТЗ составляется, как правило, заказчиком (или его представителем) при участии разработчика. Утверждается ТЗ, обычно, обеими сторонами — заказчиком и разработчиком, подписью уполномоченных лиц.

1. **На каком этапе разработки программного изделия составляется ТЗ?**

ТЗ составляется на начальном этапе разработки программного обеспечения, до начала проектирования. Его наличие необходимо для планирования работ, оценки ресурсов и согласования требований

1. **Какими документами регламентируется написание ТЗ?**

Нет единого универсального документа, регламентирующего написание ТЗ. Формат и содержание ТЗ могут варьироваться в зависимости от организации, проекта и используемых стандартов. Однако, существуют общие рекомендации и стандарты, например, ГОСТы (для России) или IEEE (для международного сообщества), которые могут быть использованы в качестве ориентира. Часто компании разрабатывают собственные шаблоны ТЗ, основанные на лучших практиках.