Дисциплина «Программирование корпоративных систем»

Рабочая тетрадь 1

Основы языка C#. Разработка калькулятора

Теоретический материал

ПЕРЕМЕННЫЕ

Для хранения данных в программе применяются **переменные**.

Переменная представляет именованную область памяти, в которой хранится значение определенного типа. Переменная имеет тип, имя и значение. Тип определяет, какого рода информацию может хранить переменная.

Перед использованием любую переменную надо определить. Синтаксис определения переменной выглядит следующим образом:

```
тип имя_переменной;
int x;
```

ТИПЫ ДАННЫХ

В языке C# есть следующие базовые типы данных:

- **bool**: хранит значение true или false (логические литералы). Представлен системным типом **System.Boolean**
- **byte**: хранит целое число от 0 до 255 и занимает 1 байт. Представлен системным типом **System.Byte**
- **sbyte**: хранит целое число от -128 до 127 и занимает 1 байт. Представлен системным типом **System.SByte**
- **short**: хранит целое число от -32768 до 32767 и занимает 2 байта. Представлен системным типом System.Int16
- **ushort**: хранит целое число от 0 до 65535 и занимает 2 байта. Представлен системным типом **System.UInt16**
- **int**: хранит целое число от -2147483648 до 2147483647 и занимает 4 байта. Представлен системным типом **System.Int32.** Все целочисленные литералы по умолчанию представляют значения типа int:
- **uint**: хранит целое число от 0 до 4294967295 и занимает 4 байта. Представлен системным типом **System.UInt32**
- **long**: хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт. Представлен системным типом **System.Int64**

- **ulong**: хранит целое число от 0 до 18 446 744 073 709 551 615 и занимает 8 байт. Представлен системным типом **System.UInt64**
- **float**: хранит число с плавающей точкой от $-3.4*10^{38}$ до $3.4*10^{38}$ и

занимает 4 байта. Представлен системным типом **System.Single**

- **double**: хранит число с плавающей точкой от $\pm 5.0*10^{-324}$ до $\pm 1.7*10^{308}$ и занимает 8 байта. Представлен системным типом **System.Double**
- **decimal**: хранит десятичное дробное число. Если употребляется без десятичной запятой, имеет значение от $\pm 1.0*10^{-28}$ до $\pm 7.9228*10^{28}$, может хранить 28 знаков после запятой и занимает 16 байт. Представлен системным типом **System.Decimal □ char**: хранит одиночный символ в кодировке Unicode и занимает 2 байта. Представлен системным типом **System.Char.** Этому типу соответствуют символьные литералы:
- **string**: хранит набор символов Unicode. Представлен системным типом **System.String**. Этому типу соответствуют строковые литералы.
- **object**: может хранить значение любого типа данных и занимает 4 байта на 32-разрядной платформе и 8 байт на 64-разрядной платформе. Представлен системным типом **System.Object**, который является базовым для всех других типов и классов .NET.

КОНСОЛЬНЫЙ ВЫВОД

Для вывода информации на консоль мы уже использовали встроенный метод **Console.WriteLine**. То есть, если мы хотим вывести некоторую информацию на консоль, то нам надо передать ее в метод Console.WriteLine:

```
Console.WriteLine("Добро пожаловать в C#!");
```

Нередко возникает необходимость вывести на консоль в одной строке значения сразу нескольких переменных. В этом случае мы можем использовать прием, который называется **интерполяцией:**

```
1  string name = "Tom";
2  int age = 34;
3  double height = 1.7;
4  Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м");
```

Для встраивания отдельных значений в выводимую на консоль строку используются фигурные скобки, в которые заключается встраиваемое значение. Это можем значение переменной ({name}) или более сложное выражение (например, операция сложения {4 + 7}). А перед всей строкой ставится знак доллара $.

При выводе на консоль вместо помещенных в фигурные скобки выражений будут выводиться их значения:

Есть другой способ вывода на консоль сразу нескольких значений:

```
1     string name = "Tom";
```

```
2    int age = 34;
3    double height = 1.7;
4    Console.WriteLine("Имя: {0}  Возраст: {2}  Рост: {1}м", name, height, age);
```

КОНСОЛЬНЫЙ ВВОД

Кроме вывода информации на консоль мы можем получать информацию с консоли. Для этого предназначен метод **Console.ReadLine()**. Он позволяет получить введенную строку.

```
1       Console.Write("Введите свое имя:
"); 2   string? name = Console.ReadLine();
3    Console.WriteLine($"Привет {name}");
```

В данном случае все, что вводит пользователь, с помощью метода Console.ReadLine() передается в переменную name.

Особенностью метода Console.ReadLine() является то, что он может считать информацию с консоли только в виде строки. Кроме того, возможная ситуация, когда для метода Console.ReadLine не окажется доступных для считывания строк, то есть когда ему нечего считывать, он возвращаает значение **null**, то есть, грубо говоря, фактически отсутствие значения. И чтобы отразить эту ситуацию мы определяем переменную name, в которую получаем ввод с консоли, как переменную типа **string?**. Здесь string указывает, что переменная может хранить значения типа string, то есть строки. А знак вопроса **?** указывает, что переменная также может хранить значение **null**, то есть по сути не иметь никакого значения.

Однако, может возникнуть вопрос, как нам быть, если, допустим, мы хотим ввести возраст в переменную типа int или другую информацию в переменные типа double или decimal? По умолчанию платформа .NET предоставляет ряд методов, которые позволяют преобразовать различные значения к типам int, double и т.д. Некоторые из этих методов:

- **Convert.ToInt32()** (преобразует к типу int)
- **Convert.ToDouble()** (преобразует к типу double)
- **Convert.ToDecimal()** (преобразует к типу decimal)

Задание 1.1

Задача:

Написать программу реализующую функционал классического калькулятора средствами языка C#, предусмотреть реализацию следующих операций:

+, -, *, /,%, 1/x, x^2, корень квадратный из x, M+, M-, MR.

В раздел решения приложить код решения и текстовое описание программного продукта по следующему плану:

1. Функционал;
2. Ограничения;
3. Возможные ошибки.

Решение:

```
class Calculator
{
    private static double MRY = 0.0;

    static void Main()
    {
        Console.WriteLine("Classic calculator");

        while (true)
        {
            Console.WriteLine("Доступные операции:");
            Console.WriteLine(" +   (сложение)");
            Console.WriteLine(" -   (вычитание)");
            Console.WriteLine(" *   (умножение)");
            Console.WriteLine(" /   (деление)");
            Console.WriteLine(" %   (деление на 100)");
            Console.WriteLine(" inv_value (1/x)");
            Console.WriteLine(" sqr (x^2)");
            Console.WriteLine(" sqrt (√x)");
            Console.WriteLine(" m+   (добавить в память)");
            Console.WriteLine(" m-   (удалить из памяти)");
            Console.WriteLine(" mr   (взять из памяти)");
            Console.WriteLine(" c    (очистить)");
            Console.WriteLine(" exit (выйти)");

            Console.Write("Выберите операцию: ");

            string operation =  Console.ReadLine().Trim().ToLower();

            if (operation == "exit")
            {
                break;
            }

            double a, b, result = 0;

            try
            {
```

```
                switch (operation)
                {
                    case "+":
                        a = ReadNumbers("Введите первое число: ");
                        b = ReadNumbers("Введите второе число: ");
                        result = a + b;
                        break;
                    case "-":
                        a = ReadNumbers("Введите первое число: ");
                        b = ReadNumbers("Введите второе число: ");
                        result = a - b;
                        break;
                    case "*":
                        a = ReadNumbers("Введите первое число: ");
                        b = ReadNumbers("Введите второе число: ");
                        result = a * b;
                        break;
                    case "/":
                        a = ReadNumbers("Введите первое число: ");
                        b = ReadNumbers("Введите второе число: ");

                        if (b == 0)
                        {
                            throw new DivideByZeroException();
                        }

                        result = a / b;
                        break;
                    case "%":
                        a = ReadNumbers("Введите делимое: ");
                        result = a / 100;
                        break;
                    case "inv_value":
                        a = ReadNumbers("Введите число: ");

                        if (a == 0)
                        {
                            throw new DivideByZeroException();
                        }
                        result = 1 / a;
                        break;
                    case "sqr":
                        a = ReadNumbers("Введите число: ");
                        result = a * a;
                        break;
                    case "sqrt":
                        a = ReadNumbers("Введите число: ");

                        if (a < 0)
                        {
                            throw new ArgumentException("Взять корень из
отрицательного числа невозможно!");
                        }
                        result = Math.Sqrt(a);
                        break;
                    case "m+":
                        a = ReadNumbers("Введите число: ");
                        MRY += a;
                        Console.WriteLine($"Память: {MRY}");
                        continue;
                    case "m-":
                        a = ReadNumbers("Введите число: ");
                        MRY -= a;
                        Console.WriteLine($"Память: {MRY}");
                        continue;
                    case "mr":
```

```
                        Console.WriteLine($"Память вызвана: {MRY}");
                        continue;
                    case "c":
                        MRY = 0.0;
                        Console.WriteLine($"Память очищена");
                        continue;
                    default:
                        Console.WriteLine("Неизвестная операция!");
                        continue;
                }

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

            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }

    static double ReadNumbers(string message)
    {
        Console.Write(message);
        string input = Console.ReadLine();
        input = input.Replace(".", ",");
        return double.Parse(input);
    }
}
```

Текстовое описание программного продукта "Классический калькулятор"

1. Функционал

Программа представляет собой консольный калькулятор, реализующий следующие математические операции:

Базовые арифметические операции:
- Сложение (+) - суммирование двух чисел
- Вычитание (-) - нахождение разности двух чисел
- Умножение (*) - произведение двух чисел
- Деление (/) - частное двух чисел

Специальные математические операции:
- Процент (%) - деление числа на 100
- Обратное значение (1/x) - вычисление числа, обратного введенному
- Квадрат числа (x^2) - возведение числа во вторую степень
- Квадратный корень ($\sqrt{x}$) - извлечение корня второй степени

Операции с памятью:
- M+ - добавление числа к значению в памяти
- M- - вычитание числа из значения в памяти
- MR - отображение текущего значения из памяти
- C - очистка памяти (сброс в ноль)

Дополнительные функции:
- Поддержка дробных чисел (с разделителем точка или запятая)
- Обработка ошибок с выводом информативных сообщений

- Непрерывный режим работы до команды выхода

2. Ограничения

Технические ограничения:
- Работа только в консольном режиме (отсутствие графического интерфейса)
- Обработка только одного числа для унарных операций
- Обработка только двух чисел для бинарных операций
- Ограничения типа double (диапазон $\pm 5.0 \times 10^{-324}$ до $\pm 1.7 \times 10^{308}$, точность 15-16 знаков)

Функциональные ограничения:
- Отсутствие истории вычислений
- Невозможность использования предыдущего результата в следующей операции
- Отсутствие скобок и приоритетов операций
- Нет поддержки комплексных чисел

3. Возможные ошибки

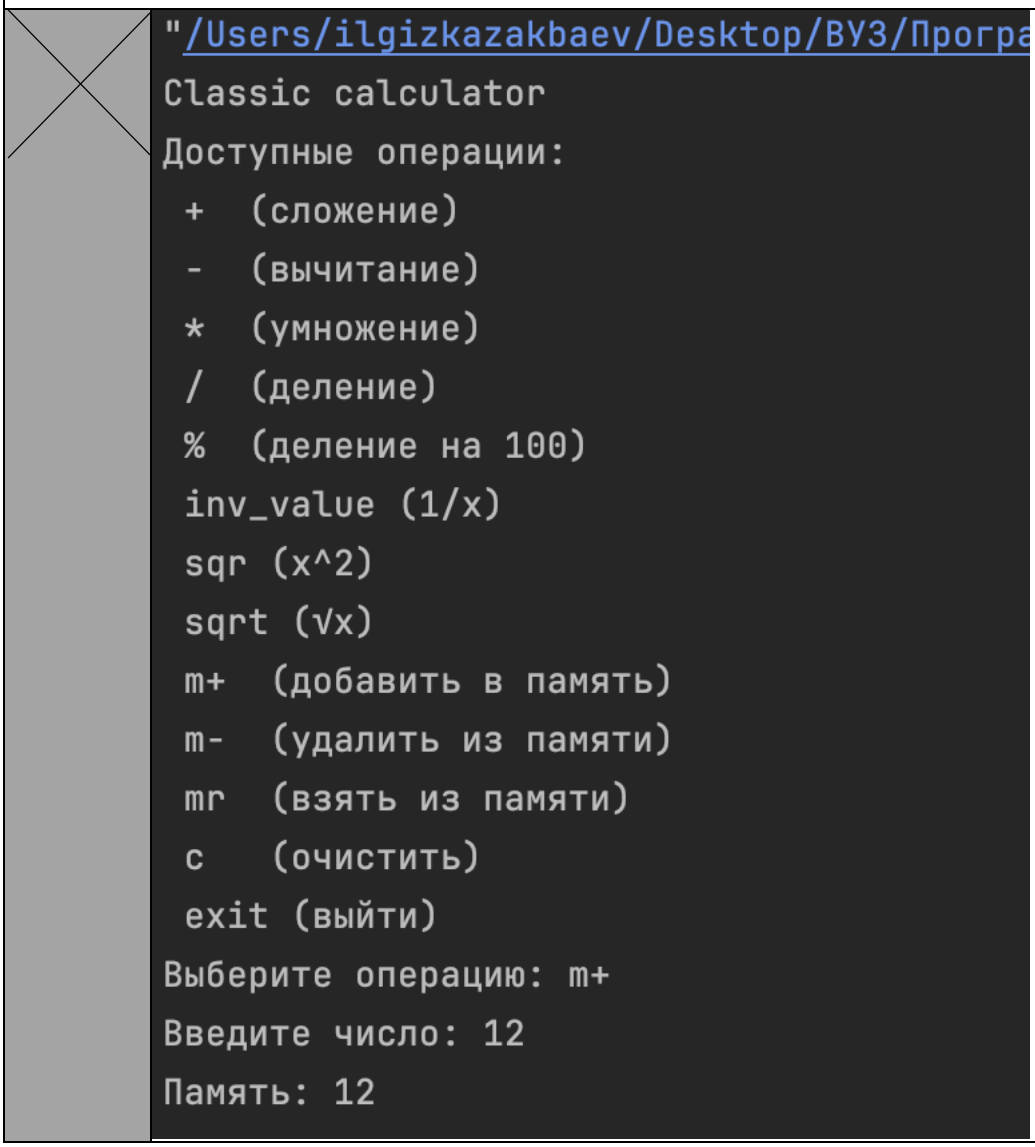
Ошибки ввода:
- Ввод нечисловых значений при запросе чисел
- Использование некорректного формата чисел
- Ввод пустой строки вместо числа

Математические ошибки:
- Деление на ноль (для операций / и 1/x)
- Извлечение квадратного корня из отрицательного числа
- Переполнение при очень больших числах
- Потеря точности при операциях с крайними значениями

Логические ошибки:
- Использование памяти до её инициализации (начальное значение 0.0)
- Накопление ошибок округления при многократных операциях
- Отсутствие проверки на допустимость операций с специальными значениями (NaN, Infinity)

Обработка ошибок:
Программа использует блоки try-catch для обработки исключений:
- DivideByZeroException - при делении на ноль
- ArgumentException - при некорректных аргументах
- FormatException - при неверном формате числа
- Все исключения выводят понятные сообщения об ошибках

Ответ:


"/Users/ilgizkazakbaev/Desktop/ВУЗ/Програ
Classic calculator
Доступные операции:
+ (сложение)
- (вычитание)
* (умножение)
/ (деление)
% (деление на 100)
inv_value (1/x)
sqr (x^2)
sqrt (√x)
m+ (добавить в память)
m- (удалить из памяти)
mr (взять из памяти)
c (очистить)
exit (выйти)
Выберите операцию: m+
Введите число: 12
Память: 12