# Дисциплина «Программирование корпоративных систем» Рабочая тетрадь 1

**Основы языка С#. Разработка калькулятора**

|  |
| --- |
| **Теоретический материал** |
| **ПЕРЕМЕННЫЕ**  Для хранения данных в программе применяются **переменные**.  Переменная представляет именнованную 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\*1038 до 3.4\*1038 и |

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

* **double**: хранит число с плавающей точкой от ±5.0\*10-324 до ±1.7\*10308 и занимает 8 байта. Представлен системным типом **System.Double**
* **decimal**: хранит десятичное дробное число. Если употребляется без десятичной запятой, имеет значение от ±1.0\*10-28 до ±7.9228\*1028, 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:*** | |
|  | using System;  class Program  {  static void Main()  {  while (true)  {  Console.WriteLine("Введите первое число");  double num\_1 = int.Parse(Console.ReadLine() ?? "");  double num\_2 = 0;  double memory = 0;  Console.WriteLine("Введите операцию");  string operation = Console.ReadLine() ?? "";  switch (operation)  {  case "+":  Console.WriteLine("Введите второе число");  num\_2 = double.Parse(Console.ReadLine() ?? "");  Console.WriteLine($"Сумма {num\_1} и {num\_2} = {num\_1 + num\_2}");  break;  case "-":  Console.WriteLine("Введите второе число");  num\_2 = double.Parse(Console.ReadLine() ?? "");  Console.WriteLine($"Разность {num\_1} и {num\_2} = {num\_1 - num\_2}");  break;  case "\*":  Console.WriteLine("Введите второе число");  num\_2 = double.Parse(Console.ReadLine() ?? "");  Console.WriteLine($"Умножение {num\_1} и {num\_2} = {num\_1 \* num\_2}");  break;  case "/":  Console.WriteLine("Введите второе число");  num\_2 = double.Parse(Console.ReadLine() ?? "");  Console.WriteLine($"Деление {num\_1} и {num\_2} = {num\_1 / num\_2}");  break;  case "%":  Console.WriteLine("Введите второе число");  num\_2 = double.Parse(Console.ReadLine() ?? "");  Console.WriteLine($"Отсаток от деления {num\_1} на {num\_2} = {num\_1 % num\_2}");  break;  case "1/x":  Console.WriteLine($"1/{num\_1} = {1 / num\_1}");  break;  case "X^2":  Console.WriteLine($"{num\_1}^2 = {Math.Pow(num\_1, 2)}");  break;  case "V":  Console.WriteLine($"sqrt({num\_1}) = {Math.Sqrt(num\_1)}");  break;  case "M+":  Console.WriteLine("Введите число чтобы записать его в сумму");  memory += double.Parse(Console.ReadLine() ?? "");  Console.WriteLine("Число успешно прибавлено");  break;  case "M-":  Console.WriteLine("Введите число чтобы вычисти его из памяти");  memory -= double.Parse(Console.ReadLine() ?? "");  Console.WriteLine("Число успешно вычтено из памяти");  break;  case "MR":  Console.WriteLine($"число в памяти = {memory}");  break;  case "ESC":  return;  default:  break ;  }  }  }  }   1. **Функционал:**   Простой калькулятор, позволяет пользователю вписать первое значение, далее выбрать нужную ему операцию, и в зависимость от операции, ему может понадобиться добавить второе значение, либо же сразу получить ответ на свой вопрос.   1. **Ограничения:**   Этот калькулятор позволяет делить на ноль, но при этом результат будет “?”.  Так же диапазон вычислений ограничен ограничениями типа “double” в c#, то есть: от +- 5.0^10-324 до +- 1.7^10308. А также +-15 знаков после запятой.   1. **Возможные ошибки:**   При попытке вписать вместо чисел любые другие символы произойдет ошибка, код расчитан на то, что пользователь будет корректно вводить данные. Однако при попытке вставить необрабатываемый символ при подстановке операции, программа вернет пользователя к вводу первого числа. |
| ***Ответ:*** | |
|  |  |