Лабораторная работа № 0 – ознакомительная.

Среда разработки MS Visual Studio. Консольные приложения.

Введение в C#. Консольный ввод и вывод.

Среда разработки MS Visual Studio. Первые программы.

(В лабораторной работе использованы задачи из учебного пособия Подбельский В.В. Язык C#. Решение задач. – М.: Финансы и статистика, 2014)

**Подготовительные мероприятия.**

Для хранения проектов создайте папку «Проекты на C#.<ФИО студента>» на рабочем диске (лучше хранить проекты на собственной флэшке).

**Порядок выполнения лабораторных работ.**

* Разобрать теоретический материал по теме
* Разобрать определения по теме
* Выполнить задания в соответствии с описанием
* Написать отчет по лабораторной работе
* Защитить отчет

**Тема 1. Среда разработки MS Visual Studio и консольные приложения.**

**Определения по теме**

* 1. ***Проект*** (***Project***) – это «контейнер» исходных кодов программы, результатов трансляции в исполнимый код, служебных файлов с описанием свойств проекта (Properties) и ссылок (References) на другие проекты
  2. ***Решение*** (***Solution***) – это средство объединения нескольких проектов в группу, «контейнер» для объединения проектов
  3. ***Обозреватель решений*** (***Solution Explorer) –*** окно среды разработки, содержащее перечень элементов решения. Отображается на экране командой «Вид\Обозреватель решений» (View\Solution Explorer).
  4. ***Консоль*** (Console, пульт управления)– это совокупность устройств, обеспечивающих взаимодействие компьютера и человека. В компьютерах консолью является комплект устройств ввода-вывода, подключенных к компьютеру. При вводе данных это клавиатура, мышь, при выводе данных – монитор.
  5. ***Трансляция программы –*** это преобразование программы, представленной на одном из языков программирования, в программу на другом языке. Виды трансляции: компиляция и интерпретация.
  6. ***Компиляция*** – это преобразование исходного кода программы с какого-либо языка программирования на машинный язык. Компилятор – транслятор, преобразующий исходный код с какого-либо языка программирования на машинный язык.
  7. ***Интерпретация*** – процесс чтения и последующего выполнения исходного кода. Прочитанный код (команда) может выполняться сразу или через создание промежуточного кода (байт-кода).
  8. ***Пространство имен*** (namespace)– способ логической группировки классов и других сущностей программы и позволяющий включать в состав проекта эти сущности оператором using. Решает также задачу дублирования имен, допуская одинаковые имена сущностей в разных пространствах имен.
  9. ***Идентификатор (identifier)–*** имя программного объекта, используемое для его идентификации и использования. В идентификаторе могут использоваться буквы, цифры и символ подчеркивания. Прописные и строчные буквы различаются. Первым символом идентификатора может быть буква или символ подчеркивания; длина может быть произвольной; пробелы внутри не допускаются.
  10. ***Правила задания идентификатора:***
* не должен совпадать с ключевыми словами;
* иметь смысловое значение (имя, отражающее сущность и назначение объекта программы);
* в дальнейшем будем использовать нотацию Camel, когда идентификатор начинается со строчной буквы, а каждое входящее в него слово с заглавной, например, maxLength.
  1. ***Ключевое слово (keyword) –*** зарезервированное слово, идентификатор, имеющий специальное значение для компилятора, например, class, const, for, …
  2. ***Тип данных (data type) –*** характеристика данных, хранимых в памяти компьютера, которая определяет:

1. Внутреннее представление данных в оперативной памяти компьютера
2. Набор допустимых операций над данными
3. Объем занимаемой данными памяти
4. Множество возможных значений (диапазоны возможных значений).
   1. ***Переменная (variable) –*** именованная область оперативной памяти, обозначаемая идентификатором (именем), предназначенная для хранения данных (значений), например, числа или текста. Переменная в процессе исполнения программы может принимать разные значения. Для использования переменной ее нужно объявить, то есть дать ей имя и указать, какого рода данные будут в ней храниться, т.е. указать тип переменной. Прежде чем использовать переменную, ей нужно задать значение (говорят «присвоить»). Для использования хранящегося значения достаточно указать соответствующее имя переменной.
   2. ***Константа или литерал (constant) –*** способ адресации к неизменяемым данным программы. Константы бывают именованными и неименованными. Неименованная константа задается своим значением, например, числом 10; логическим значением true; строковым значением «Здесь был Вася».

Компилятор относит константу к одному из типов данных по ее виду и значению. Иногда требуется явным образом задать тип константы. Для этого используются суффиксы (таблица 1)

Таблица 1 – Суффиксы целых и вещественных констант

|  |  |
| --- | --- |
| Суффикс | Значение |
| L, l | Длинное целое (long) |
| U, u | Беззнаковое целое (unsigned) |
| F, f | Вещественное с одинарной точностью (float) |
| D, d | Вещественное с двойной точностью (double) |
| M, m | Финансовое десятичного типа (decimal) |

* 1. ***Управляющая последовательность или простая Escape-последовательность (escape sequence) – определенный символ, предваряемый обратной косой чертой (таблица 2). Используется для представления:***
* кодов, не имеющих графического представления
* символов, имеющих специальное значение в строковых и символьных литералах.

Таблица 2 – Управляющие последовательности

|  |  |
| --- | --- |
| Вид | Наименование |
| \a | Звуковой сигнал |
| \b | Возврат на шаг |
| \f | Перевод страницы |
| \n | Перевод строки |
| \r | Возврат каретки |
| \t | Горизонтальная табуляция |
| \v | Вертикальная табуляция |
| \\ | Обратная косая черта |
| \’ | Апостроф |
| \” | Кавычка |
| \0 | Нуль-символ |
| \0x | Символ в шестнадцатеричном коде |
| \U (u)<шестнадцатеричный код> | Символ в кодировке Unicode |

**Лабораторная работа. Часть 1. Создание первых проектов.**

***Задание 1.1.*** Запустите программу MS Visual Studio. На начальной странице перейдите по ссылке «Создать проект».

***Далее в примерах в имени проекта и в имени решения используются буквы русского алфавита, пробелы и точки. Несмотря на это здесь рекомендуется использовать буквы английского алфавита без точек и пробелов (последние можно заменить знаком нижнего подчеркивания: «\_»)***

1. Введите данные проекта (рисунок 1):
   1. Выберите тип проекта «Visual C#»;
   2. Выберите вид приложения «Консольное приложение» (Console Application);
   3. В поле ввода «Имя» введите имя проекта (Name) «Тема 1. Задача 1»;
   4. В поле «Расположение» (Location) укажите полное имя папки (каталога), в которой будет сохранено решение «Проекты на C#.<ФИО студента>»;
   5. В поле «Имя решения» (Solution name) введите имя решения «Тема 1»;
   6. Установите флаг «Создать каталог для решения» («Create directory for solution);
   7. Завершите создание проекта кнопкой «ОК».

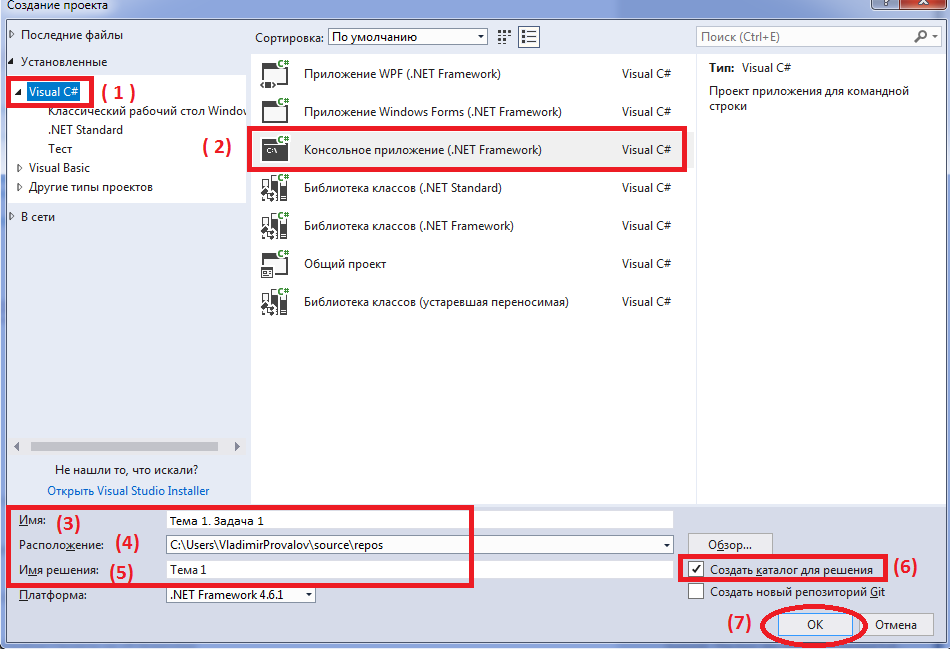


Рисунок 1. Создание первого проекта

1. Среда разработки сгенерирует шаблон консольного приложения (рисунок 2).

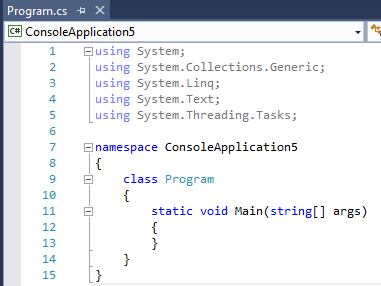


Рисунок 2. Шаблон консольного приложения

1. Запишите между внутренними фигурными скобками следующий текст (инструкции, команды)

Console.WriteLine("Это первая программа лабораторной работы!");

Console.WriteLine("Для завершения нажмите любую клавишу...");

Console.ReadKey();

Обратите внимание, как среда разработки выводит подсказки в виде списков с текстом команд, несоответствие текста команды синтаксису. Не забывайте в конце каждой инструкции указывать точку с запятой. Об этом тоже среда разработки предупреждает. Последняя инструкция позволяет задержать окно сообщения на экране при выполнении приложения до нажатия клавиши на клавиатуре.

1. Прокомментируем текст программы.
   1. В начале программы размещены операторы ***using***, обеспечивающие доступ к нескольким пространствам имен, содержащим стандартные классы платформы ***.NET Framework***. Для данной программы достаточно оставить подключение одного пространства имен ***System***. *Удалите из приложения все подключения пространства имен, кроме пространства System****.***
   2. Метод Main() принадлежит классу с именем Program {}, предложенным средой разработки. Имя может быть изменено. *Измените имя класса на* *MyProgram*.
   3. Класс *MyProgram с методом* Main() располагается в пространстве имен, заданным по умолчанию средой разработки и совпадающим с именем проекта «Тема 1. Задача 1». Это имя при необходимости можно изменить. *В данном проекте пространство имен задавать не обязательно. Удалите его вместе с парой скобок {}*.
   4. В заголовке метода Main() задан параметр ***string[] args,*** используемый для передачи параметров приложению при его запуске из командной строки. *В данном приложении использование параметра не предполагается, поэтому удалите его.*
   5. Метод ***WriteLine***() класса ***Console*** пространства имен ***System*** выводит на консоль (на устройство вывода), в качестве которой выступает монитор, текст, заданный параметром метода.
   6. Метод ReadKey() класса ***Console*** пространства имен ***System*** ожидает нажатия клавиши на клавиатуре (консоли, как устройства ввода), предотвращая закрытие окна вывода на экране до нажатия клавиши клавиатуры в процессе выполнения.

В результате выполненных преобразований получим следующий код программы (рисунок 3).

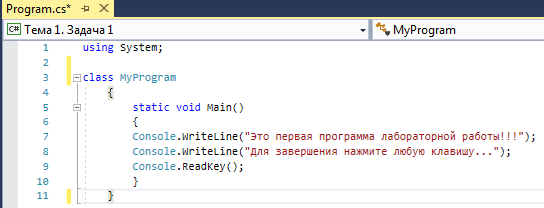


Рисунок 3. Код первой программы лабораторной работы №1

1. Скомпилируем и выполним приложение командой «***Отладка\Запуск без отладки***» (Start Without Debugging) линейки меню. В результате получите следующий результат выполнения консольного приложения (рисунок 4).

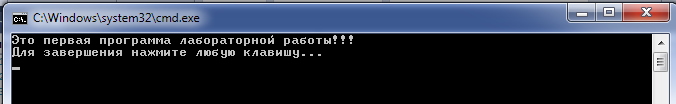


Рисунок 4. Результат выполнения программы

После нажатия любой клавиши на клавиатуре окно вывода результатов работы приложения закроется.

***Создание второго проекта в текущем решении.***

***Задание 1.2.*** Программа должна «спросить» у пользователя его имя и «поздороваться» с ним по имени.

1. В окне обозревателя решений вызовите контекстное меню строки ***Решение «Тема 1»*** и выполните команду «Добавить\Создать проект».
2. В диалоговом окне задайте параметры нового проекта:
   1. Тип проекта «Visual C#»;
   2. Вид приложения «Консольное приложение»;
   3. Имя проекта «Тема 1. Задача 2»;
3. Имя папки (каталога) проекта оставьте прежним. Обратите внимание, система ***на запрашивает имя и необходимость создания нового каталога для решения!!!***
4. Введите текст программы и приведите проект к виду, представленному на рисунке 5.

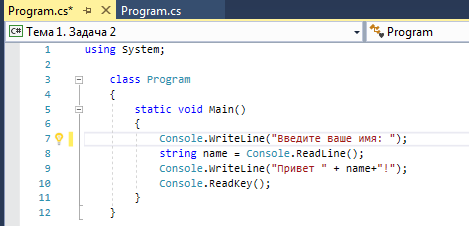


Рисунок 5. Решение задачи 2

1. Прокомментируем приведенный код.
   1. В программе используется переменная ***name*** строкового типа (string).
   2. Метод ***ReadLine***() класса ***Console*** пространства имен ***System*** считывает из входного консольного потока (клавиатуры) строку символов, введенных пользователем (по нажатию клавиши Enter) и помещает ее в переменную name.
   3. Параметр метода ***WriteLine***() предпоследней строки представляет собой конкатенацию строковых литералов и переменной name.
2. Перед компиляцией и выполнением кода второго проекта необходимо сделать его «стартовым». Для этого в обозревателе решений вызовите контекстное меню второго проекта и выполните команду «Назначить автозагружаемым проектом» (Set as StartUp Project).
3. Запустите проект на выполнение «***Отладка\Запуск без отладки***» (Start Without Debugging). Должен быть получен следующий результат (рисунок 6).

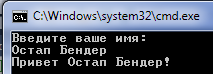


Рисунок 6. Результат выполнения программы 2.

***Задание 1.3. Файлы решения и его проектов.***

Рассмотрим структуру каталогов решения «Тема 1» в папке «Проекты на C#.<ФИО студента>».

В папке «Тема 1» расположены две папки проектов решения Тема 1. Задача 1 и Тема 1. Задача 2, а также файл Тема 1.SLN. С помощью последнего файла можно запустить новый экземпляр Visual studio c загруженным решением.

Каталоги проектов имеют следующую структуру:

Тема 1. Задача 1

[bin]

[obj]

[Properties]

Program.cs

Тема 1. Задача 1.csproj

Текстовый файл ***Program.cs*** содержит код программы проекта, который можно отредактировать в любом текстовом редакторе. *Откройте этот файл в каком-либо текстовом редакторе, например, в блокноте.*

Далее рассмотрим содержимое каталога ***Bin\Debug***. Он содержит файл ***Тема\_1. Задача\_1.exe*** с типом Приложение, который содержит исполняемый кодом проекта. Этот файл содержит код программы на промежуточном языке и выполняется средствами среды исполнения ***.NET Framework.*** Этот файл может быть выполнен вне среды разработки Visual Studio но при установленной на компьютере версии платформы ***.NET Framework.***

00000000Выполните следующие задания:

* Скопируйте исполняемый файл ***Тема\_1. Задача\_2.exe*** в корневой каталог доступного диска или флэшки.
* Запустите консольное окно (командную строку) Windows (кнопка с флагом windows + R; наберите команду cmd).
* В консольном окне укажите полное имя файла <Диск>:\ ***Тема\_1. Задача\_2.exe***

***Задание 1.4. Самостоятельное задание***

1. Создайте новый проект в решении Тема 1.
2. Напишите программу, которая просит ввести с клавиатуры ваше имя, название учебной группы и название учебного заведения, в котором вы учитесь, и выводит это на экран в виде одной строки.

**Лабораторная работа. Часть 2. Консольный вывод**

***Задание 2.1.*** Используя в строках управляющие символы (escape-последовательности) вывести в виде таблицы из двух колонок суффиксы и названия типов числовых констант (таблица 3).

Суффикс Значение Суффикс Значение

<пусто> int <пусто> double

L, l long F, f float

U, u uint M, m decimal

Ul, ul ulong D, d double

1. Создайте новый проект Задача 1 в решении Тема 2 и введите текст программы, представленный на рисунке 1. Решение задачи представлено на рисунке 1.

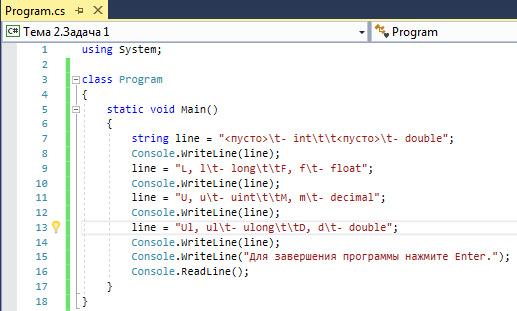


Рисунок 1. Текст программы задачи 2.1

1. Выполните программу. Результат работы программы представлен на рисунке 2.



Рисунок 2. Результат решения задачи 2.1

***Задание 2.2.*** Вывести значения выражений типов bool, int и double.

1. Создайте новый проект Задача 2 в решении Тема 2 и введите текст программы, представленный на рисунке 3.

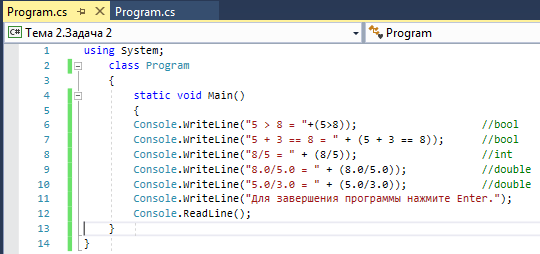


Рисунок 3. Текст программы задачи 2.2

1. *Выполните программу*. Результат работы программы представлен на рисунке 4.

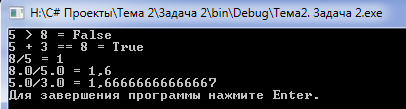


Рисунок 4. Результаты решения задачи 2.2

1. *Объясните полученные результаты.*
   1. *Почему нет ошибки при конкатенации строковой константы и выражения не строкового типа?*
   2. *Что означает операция ==?*
   3. *Почему результат деления 8/5 = 1, а 8.0/5.0 =1,6*
   4. *Почему в результате деления 5.0/3.0 такое количество цифр после запятой, в то время, как в предыдущем примере одна?*

**Европейский и американский форматы вывода вещественных чисел.**

В европейском (немецкий, французский, русский, …) формате представления вещественных чисел для разделения целой и дробной части используется запятая, в то время, как в американском (американский, английский) формате – точка. Национальный формат устанавливается в настройках операционной системы, которые могут быть различными. Это может приводить к нежелательным последствиям при использовании программы.

Среда .NET позволяет устанавливать требуемый формат вывода вещественных чисел независимо от настроек операционной системы.

***Задание 2.3.*** Вывести значение числа в формате, установленном в операционной системе, затем в американском, затем в европейском форматах. Число доступно как константа класса Math.

1. Создайте новый проект Задача 3 в решении Тема 2 и введите текст программы, представленный на рисунке 5.

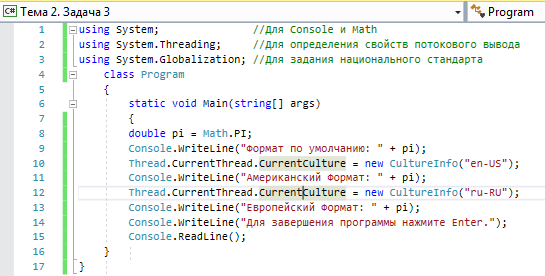


Рисунок 5. Текст программы задачи 2.3

1. Результат выполнения программы представлен на рисунке 6.

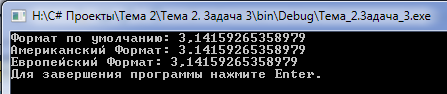


Рисунок 6. Результат решения задачи 2.3

1. Объясните полученные результаты.

**Форматирование вывода чисел. Спецификаторы форматов вывода.**

Метод Console.WriteLine() имеет множество перегрузок для вывода различных типов данных. Одним из способов вывода различных типов данных, в том числе и числовых, является вывод текстового представления выводимых заданных объектов, за которыми следует текущий признак конца строки, в стандартный выходной поток с использованием заданных сведений о форматировании. Формат метода [***WriteLine(String, Object, Object)***](https://docs.microsoft.com/ru-ru/dotnet/api/system.console.writeline?view=netframework-4.8#System_Console_WriteLine_System_String_System_Object_System_Object_).

Строка вывода текстового представления включает в себя неизменяемые символы и специальные конструкции, называемые полями подстановок. Структура поля подстановки имеет вид {N[,[-]W][:S[R]}, где N – номер аргумента; W – ширина поля; S – спецификатор формата; R – спецификатор точности (число от 0 до 99). Если перед величиной ширины поля стоит знак минус, выравнивание по левому краю, иначе - по правому краю. Спецификаторы формата S и точности R представлены в таблице 3.

Таблица 3 – Спецификаторы формата S и точности R для числовых данных

|  |  |  |
| --- | --- | --- |
| Спецификатор | Формат представления | Роль спецификатора точности |
| C или с | Валютный | Количество десятичных разрядов |
| D или d | Целочисленный | Минимальное число цифр |
| E или e | Экспоненциальный | Число разрядов после точки |
| F или f | С фиксированной точкой | Число разрядов после точки |
| G или g | Короткий из E или F | Подобен E или F |
| X или x | Шестнадцатеричный | Минимальное число цифр |
| N или n | С разделителями тирад | Количество знаков после точки |
| P или p | Процент | Количество знаков после точки |

***Задание 2.4.*** Вывести строковые представления чисел 10761.937554 и 8395 с использованием спецификаторов формата.

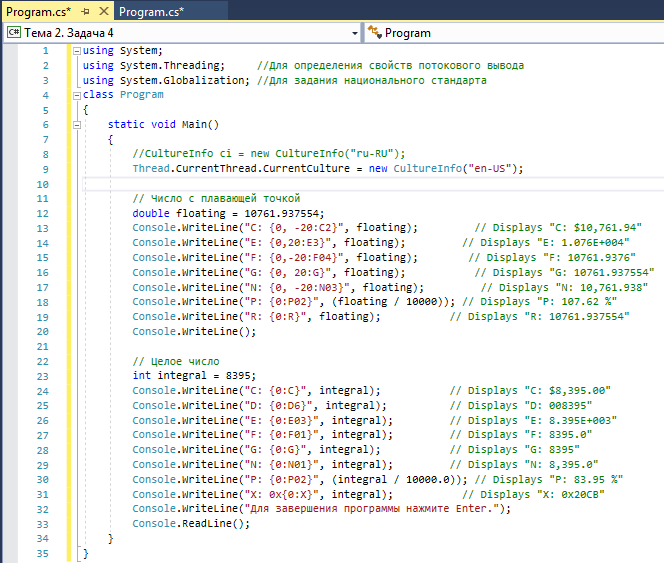
1. Создайте новый проект Задача 4 в решении Тема 2 и введите текст программы, представленный на рисунке 7.

Рисунок 7. Текст программы задачи 2.4.

1. Результат выполнения программы представлен на рисунке 7.

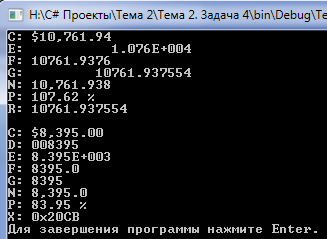


Рисунок 8. Результат решения задачи 2.4.

1. Объясните полученный результат.

**Форматирование вывода символьного представления арифметических значений.**

Чтобы определить способ форматирования числовых данных, можно создать строку настраиваемого (пользовательского) числового формата, состоящую из одного или нескольких описателей настраиваемого формата (таблица 4). Описатель формата указывается вместо спецификатора формата, использованного в предыдущем примере.

Кроме того, для приведения числовых данных к требуемому виду можно использовать метод ***ToString***(), унаследованный всеми типами данных от общего базового класса ***object***. ***Аргументом метода является формат символьного представления данных.***

Таблица 4 – Описатели настраиваемого формата

|  |  |  |
| --- | --- | --- |
| Спецификатор | Обозначение | Примеры |
| 0 | Любая цифра и нуль | 5.6 и «00.00» =>05.60  .27 и «0.0» =>0.3  26.5 и «00» =>27 |
| # | Любая цифра | 26.5 и «##» =>27  5.6 и «##.##» =>5.6 |
| . | Десятичная точка |  |
| , | Разделитель триад | 9876544321 и «00,00» =>9,876,544,321 |
| % | Процент | 0,005432 и «#0.##%» =>0.54% |
| E0 или e0 | Экспоненциальная нотация | 1234500 и «0.##E+00» =>1.23E+06 |
| ‘ABC’ | Литеральная подстановка строки | 30 и «0 ‘ руб.’ » => 30 руб. |
|  |  |  |

***Задание 2.5.*** Определить переменную типа Double, инициируя ее тем приближенным значением числа которое предоставляет класса System.Math. Вывести строковые представления значения переменной и результатов ее умножения и деления на 10, Применить в методе ToString() строки формата «0.000» и «0.000E0».

1. Создайте новый проект Задача 5 в решении Тема 2 и введите текст программы, представленный на рисунке 9.

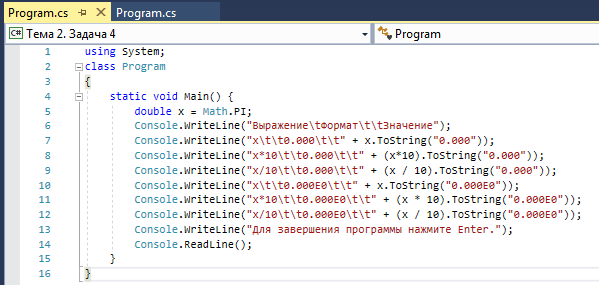


Рисунок 9. Текст программы задачи 2.5

1. Скомпилируйте и выполните программу. Результат выполнения программы представлен на рисунке 10.

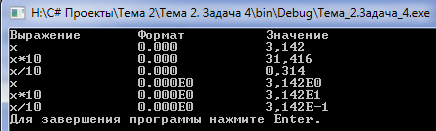


Рисунок 10. Результат решения задачи 2.5

# Интерполяция строк на C#

Еще одним из способов вывода различных типов данных, в том числе и числовых, является предварительное формирование текстового представления выводимых заданных объектов в виде строки методом [Format(String, Object, Object, Object)](https://docs.microsoft.com/ru-ru/dotnet/api/system.string.format?view=netframework-4.8#System_String_Format_System_String_System_Object_System_Object_System_Object_) и последующим выводом ее в консоль. Например,

*decimal temp = 20.4m;*

*string s = String.Format("The temperature is {0}°C.", temp);*

*Console.WriteLine(s);*

В этом примере сначала формируется форматная строковый литерал, помещаемый в переменную s. Обратите внимание, что форматная строка не содержит параметра, содержащего значение для вывода. Вывод в консоль осуществляется методом *Console.WriteLine(s)*.

Более удобный синтаксис предоставляет функция интерполяции строк для включения форматированных результатов выражения в строку результатов.

Для определения строкового литерала в качестве интерполированной строки к ее началу добавл

яется символ $. Например,

*double a = 3;*

*double b = 4;*

*string s = $" Площадь прямоугольного треугольника со сторонами {a} и {b} = {0.5 \* a \* b}";*

*Console.WriteLine(s);*

В подстановках интерполированной строки ставится аргумент, а не его номер, как в обычной форматной строке.

Однако при использовании обоих способов форматной строки (со знаком $ и без) возникают некоторые трудности вывода Esc-последовательностей (таблица 1). В этом случае, в интерполированную строку, начинающуюся с символа $ следует добавить за ним символ @.

***Задание 2.6.*** Вывести на экран строки, содержащие ESC-последовательность «C:\Users\Jane\Documents» с использованием обоих видов форматной строки.

1. Создайте новый проект Задача 6 в решении Тема 2 и введите текст программы, представленный на рисунке 11.

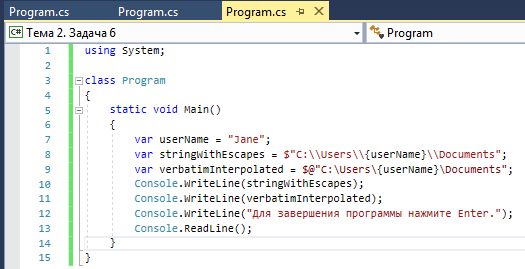


Рисунок 11. Текст программы задачи 2.6

1. Скомпилируйте и выполните программу. Результат выполнения программы представлен на рисунке 12.

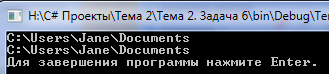


Рисунок 12. Результаты решения задачи 2.6