***Знакомство со средой разработки Visual C#***

## Указания к работе:

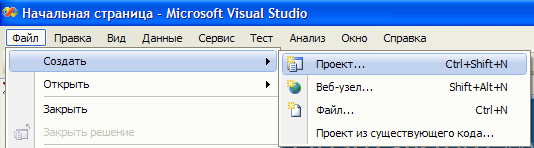
1. Запустите Visual Studio из главного меню путем выполнения следующей последовательности команд:

*Пуск → Все Программы → Microsoft Visual Studio → Microsoft Visual Studio*

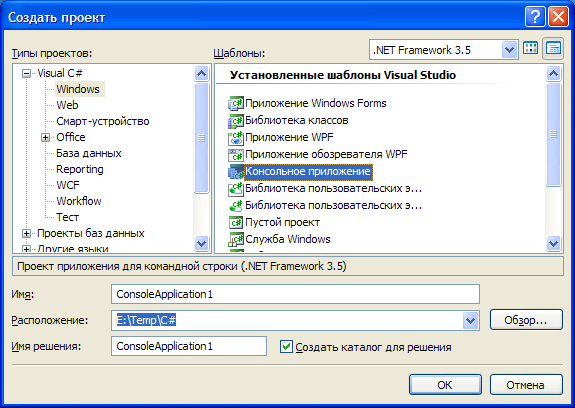
1. В открывшемся окне программы выберите команду

*Файл → Создать → Проект*

Как показано на рисунке:



1. В открывшемся диалоге создания нового проекта выберите язык программирования *Visual C#*, тип приложения *Windows* шаблон приложения *Консольное приложение*:



Поменяйте имя проекта и путь для его сохранения в соответствии с указаниями преподавателя.

1. *Visual Studio* автоматически создаст проект и добавит в него начальный исходный текст.

Простейшая программа на языке C#, выводящая на экран строку текста, выглядит следующим образом:

using System;

namespace MyAppC

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

Console.ReadLine();

}

}

}

1. Сохраните проект в отдельный каталог на диск, открытый для записи (обычно это диск F:). Откройте каталог с помощью проводника Windows или файлового коммандера и найдите в этом каталоге файл решения (с расширением SLN) и папку проекта. Откройте папку проекта и найдите файл проекта (с расширением CSPROJ) и файл с исходным текстом программы (он имеет расширение CS).
2. Первый шаг, необходимый для выполнения программы – ***компиляция исходного файла в промежуточный код, готовый для выполнения***. Выполните команду *Построение* → *Построить решение* (F6).

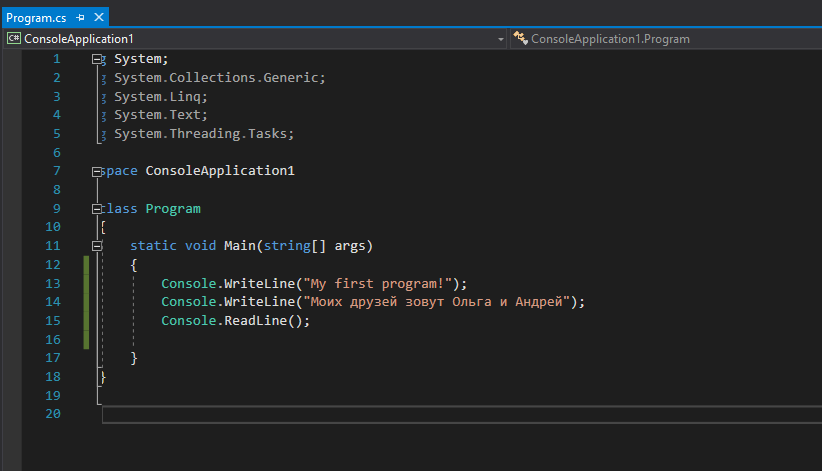
После выполнения компиляции откройте папку BIN и найдите в ней:

* Исполнимый файл программы (с расширением EXE);
* Файл с отладочной информацией программы (с расширением PDB);
* Служебный файл, используемый интегрированной средой для запуска программы (с расширением VSHOST.EXE);
* Файл манифеста с описанием сборки (с расширением MANIFEST), который является текстовым и его можно просматривать.

1. Для запуска программы на выполнение следует выполнить команду *Отладка* → *Запуск без отладки* или нажать CTRL+F5.

## Задание на работу:

1. Ознакомиться со средой разработки *Visual Studio*.
2. Создать новое консольное приложение.
3. Добавить в него код для вывода на экран фразы ”My first program!”, скомпилировать и выполнить программу.
4. Модифицировать программный код таким образом, чтобы программа при запуске выводила на экран имена Ваших друзей.



## Арифметические выражения

Данная работа основана на программировании вычисления арифметических выражений. Для примера рассмотрим выражение вида:



Для программирования вычисления этого выражения на С# необходимо:

1. Записать его в виде, пригодном для программирования, заменяя:
   * дроби на операции деления;
   * операцию возведения в степень - операциями последовательного умножения;
   * стандартные математические функции - их обозначениями в С# (с использованием справочной системы если необходимо).

Для данного выражения можно использовать два идентичных способа записи:

а) a=((b+c)\*(b+c)-1.25)/(c-2)/Math.Sqrt(b-5);

б) a=((b+c)\*(b+c)-1.25)/((c-2)\*Math.Sqrt(b-5));

1. Позаботиться об описании исходных данных и их вводе. В данном выражении для вычисления a необходимы значения переменных b и c, которые могут быть как целыми (int), так и действительными (double). В любом случае для ввода b и c необходимо использовать стандартный метод ***ReadLine()*** класса ***Console***. Этот метод вводит символьную строку, которую затем необходимо преобразовать в значение нужного типа, воспользовавшись методами ***ToInt32()*** или ***ToDouble()*** класса ***Convert***:

int b = Convert.ToInt32(Console.ReadLine());

double c = Convert.ToDouble(Console.ReadLine());

Таким образом, для описания и ввода целочисленных переменных b и c может быть использован следующий фрагмент программы:

int b, c; // объявление переменных

// текст, запрашивающий ввод значений

Console.Write("b = ");

// ввод, преобразование и присваивание

b = Convert.ToInt32(Console.ReadLine());

Console.Write("c = ");

c = Convert.ToInt32(Console.ReadLine());

1. Позаботиться об описании результата вычислений (в нашем случае - переменная a)и его выводе на экран. Для вывода на экран числовых и символьных значений используется стандартный метод ***WriteLine()*** класса ***Console***:

Console.WriteLine("a = " + a);

Этот метод выводит на консоль текстовую строку, получаемую сцеплением строковой константы "a = " и числового значения переменной a, которое перед выводом автоматически преобразуется в текстовую строку. Сцепление текстовых строк производится с помощью операции «+».

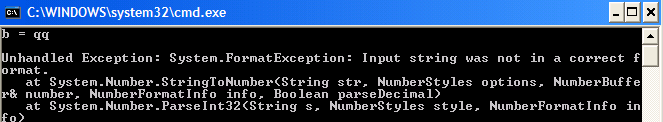
## Проверка значений введенных данных

Так как ввод исходной информации в программу выполняется пользователем, нет никакой гарантии, что введенные данные будут корректными и обеспечат правильный результат вычислений.

Поэтому необходимо для устойчивой и безошибочной работы программы предусматривать проверку данных, которая может включать:

* Контроль корректности ввода числовой информации с клавиатуры (с помощью методов класса ***Convert***);
* Контроль принадлежности введенных данных области допустимых значений (с помощью условного оператора if()).

Методы класса ***Convert*** выполняют преобразование введенной пользователем символьной строки в необходимый для работы программы тип данных. Если выполнить нужное преобразование невозможно (обычно из-за ошибки пользователя при вводе), возникает исключительная ситуация и программа аварийно заканчивается:

******

Обработка такого рода ошибок требует знания технологии обработки исключительных ситуаций, рассмотрение которой выходит за рамки данной работы и поэтому относится к заданиям повышенного уровня сложности.

Как правило, простые программы при неверном вводе не пытаются «исправить ситуацию», запросив повторный ввод или каким-либо иным способом. Для ошибок, связанных с попаданием введенных значений в область определения стандартных функций достаточно выдать пользователю сообщение об ошибке и завершить программу. Это можно сделать с помощью условного оператора if() (если) следующим образом:

if(<*Проверяемое условие*>)

{

Console.WriteLine(“Ошибка при вводе значения А!”);

}

else

{

// Остальные операторы программы

}

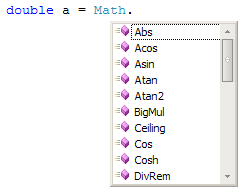
Таким образом выполняется контроль вводимых числовых значений на принадлежность области определения стандартных функций, проверки возможности возникновения ситуации деления на ноль и т.п.

## Получение контекстной помощи

Справочная система С# содержит (на английском или русском языке в зависимости от используемой версии) достаточно большое количество справочной информации:

* О разработке различных видов программ с помощью С#,
* О стандартных методах, имеющихся в классах С#.

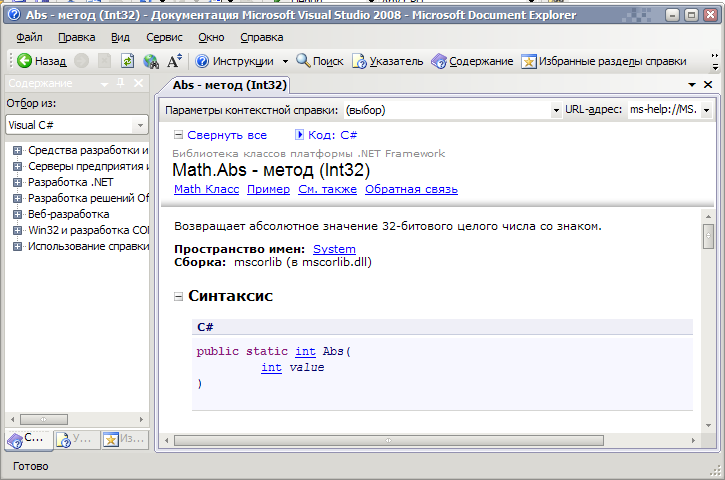
Простейшим способом получения помощи является использование интеллектуальных возможностей редактора кода Visual Studio. При использовании любого стандартного класса (Convert, Console, Math) после ввода имени класса и символа «точка» открывается список, где можно выбирать из доступных данных и методов этого класса:



После выбора из списка имени метода и открытия круглой скобки выводится краткая информация о параметрах метода и возвращаемом им значении:



Подробную информацию о назначении каждого метода можно получить с помощью клавиши F1. Если перед этим выделить в тексте программы имя интересующего Вас метода, то будет выдано его описание в виде:



# Задание на выполнение работы

Написать программу, которая подсчитывает результат выражения (выражение студент формирует самостоятельно). Выписать и проанализировать возможные ошибки, возникающие при работе программы. Добавить в программу проверку и предупреждение ошибок.

# Выполнение работы

1. Сформировать выражение, значение которого планируется вычислить.
2. Записать выражение, заданное в полученном варианте в виде, пригодном для программирования:

* добавить скобки, если нужно;
* уточнить имена стандартных методов, для чего использовать контекстную помощь.

1. Написать программу, выполняющую

* ввод исходных данных;
* расчёт по заданной формуле;
* вывод результатов расчёта;

1. Запустить Visual С#, создать новый проект консольного приложения, набрать программу и сохранить проект на жёсткий диск.
2. Скомпилировать (ЭТО МЕНЮ СБОРКА-ПОСТРОИТЬ) программу F6, не запуская её на выполнение.
3. Программу выполнить не менее трёх раз, для различных исходных данных, дающих:

* верный результат без ошибок,
* арифметическую ошибку (например, деление на ноль),
* ошибку области определения функции.

Количество запусков программы с различными исходными данными определяется студентом самостоятельно таким образом, чтобы вызвать ***все*** обрабатываемые программой ошибки.

1. Изменить программу, добавив в неё проверку исходных данных и выполнить её ещё раз для данных, обработка которых приводила к ошибке.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1zadanie2

{

class Program

{

static void Main(string[] args)

{

int x, y;

double a; // объявление переменных

// текст, запрашивающий ввод значений

Console.Write("x = ");

// ввод, преобразование и присваивание

x = Convert.ToInt32(Console.ReadLine());

Console.Write("y = ");

y = Convert.ToInt32(Console.ReadLine());

if ((5 \* x \* x \* x - 6 \* y) != 0)

{ a = (15 \* Math.Pow(x, 2) - 7 / 12 \* y) / (5 \* x \* x \* x - 6 \* y);

Console.WriteLine("a = " + a);

}

else

{

Console.Write("Деление на ноль, введите другие числа");

}

Console.ReadLine();

}

}

}

