Software Development Techniques

Lecture 1

What This Course Is:

- This course is designed to teach students the beginnings of programming
- This course will use Microsoft .NET and C# as the tools of programming
- Since you will be using other languages throughout the program, we will pay more attention at the common programming elements and less at windows details

What This Course Is Not:

- This course will not teach you to be the greatest C# programmer ever
- We will not be going far enough in depth of C# and .NET so you can claim to be an experienced C# programmer
- .NET and C# is a huge programming area and even in the 75 hour course we will not be able to cover everything.

How to Start?

- If you never programmed before, please pay attention to everything said in this class
- Repeat all exercises and try to be present at every lecture and lab
- I recommend to look through "Murach's C# 2015" book
- It is also recommended to use Lynda.com tutorials, for which you have free access

Tools We Will Use

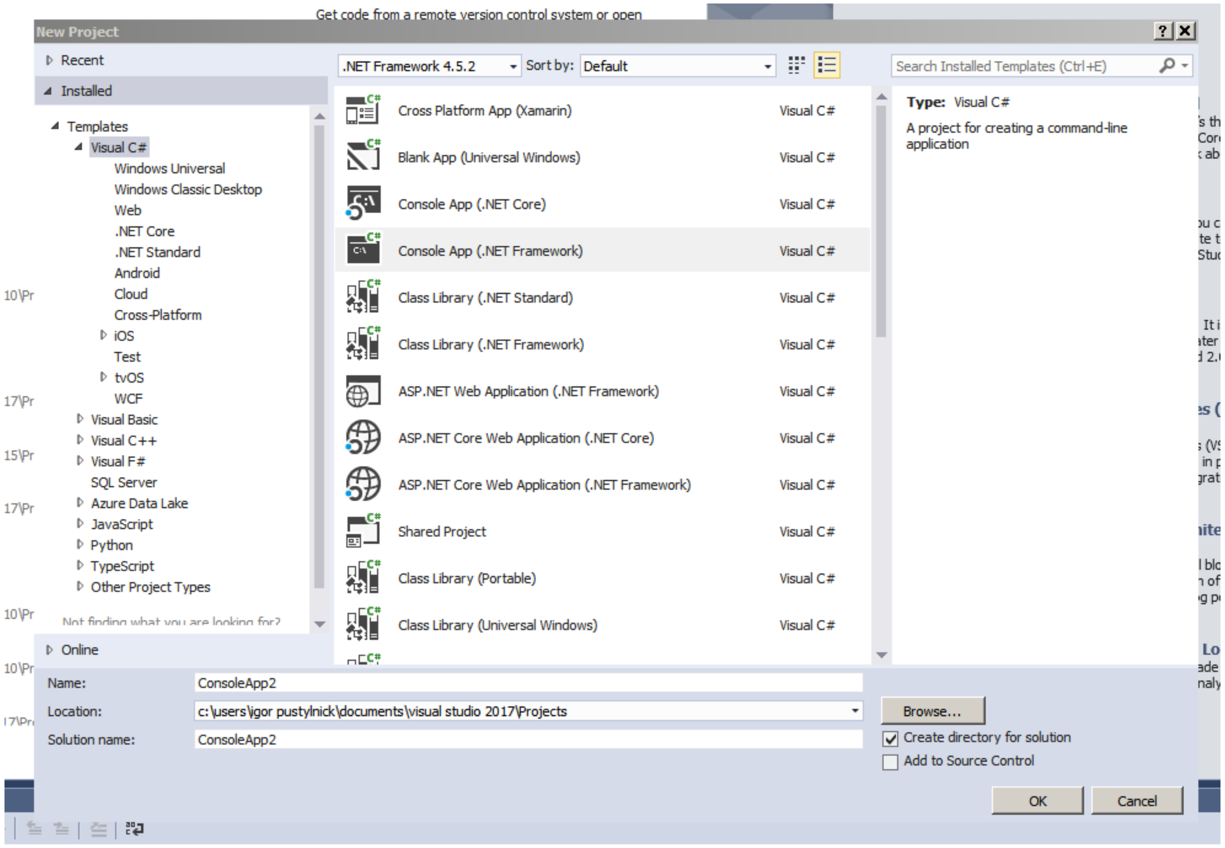
- We will definitely use Visual Studio 2017. If you have Visual Studio 2015, you can use it as an alternative
- We will use C#, that comes with Visual Studio.
- We will also use Microsoft SQL Server 2016
- For your assignments you may need Word 2016 (to put them together)
- All tools are available for download in Microsoft Imagine web site

Programming: How do I Start

- First of all, every program needs an idea.
- After we got the idea, we need a proper starting point
- We need to look at our starting point (where we have nothing) and make a plan on how we get to the finish point (ready program)
- The rest is easy: we just write a program

Visual Studio Primer(1)

- Open Visual Studio and start a new project



New Project

- Recent
- Installed
  - Templates
    - Visual C#
      - Windows Universal
      - Windows Classic Desktop
      - Web
      - .NET Core
      - .NET Standard
      - Android
      - Cloud
      - Cross-Platform
      - iOS
      - Test
      - tvOS
      - WCF
    - Visual Basic
    - Visual C++
    - Visual F#
    - SQL Server
    - Azure Data Lake
    - JavaScript
    - Python
    - TypeScript
    - Other Project Types
- Online

.NET Framework 4.5.2 | Sort by: Default | Search Installed Templates (Ctrl+E)

Cross Platform App (Xamarin)	Visual C#
Blank App (Universal Windows)	Visual C#
Console App (.NET Core)	Visual C#
Console App (.NET Framework)	Visual C#
Class Library (.NET Standard)	Visual C#
Class Library (.NET Framework)	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#
ASP.NET Core Web Application (.NET Core)	Visual C#
ASP.NET Core Web Application (.NET Framework)	Visual C#
Shared Project	Visual C#
Class Library (Portable)	Visual C#
Class Library (Universal Windows)	Visual C#

Type: Visual C#

A project for creating a command-line application

Name: ConsoleApp2

Location: c:\users\igor pustylnick\documents\visual studio 2017\Projects

Solution name: ConsoleApp2

Browse...

[✓] Create directory for solution

[ ] Add to Source Control

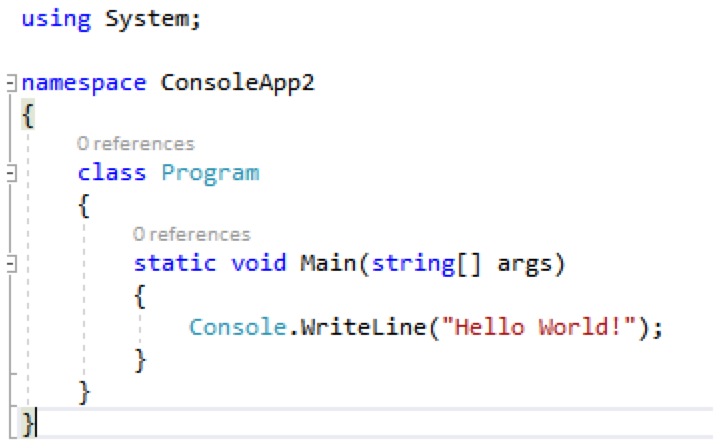
OK | Cancel

Types of Projects

- We will be interested in two types of projects:
  - Console application, which runs in windows console
  - Windows Forms application, which allows to create and run applications consisting of multiple windows forms

Console Application

- Console application always has at least one file Program.cs
- The template provides one class "Program" and one function "Main"
- The function "Main" is the entry point to the console application
- If there is no Main the program will not run



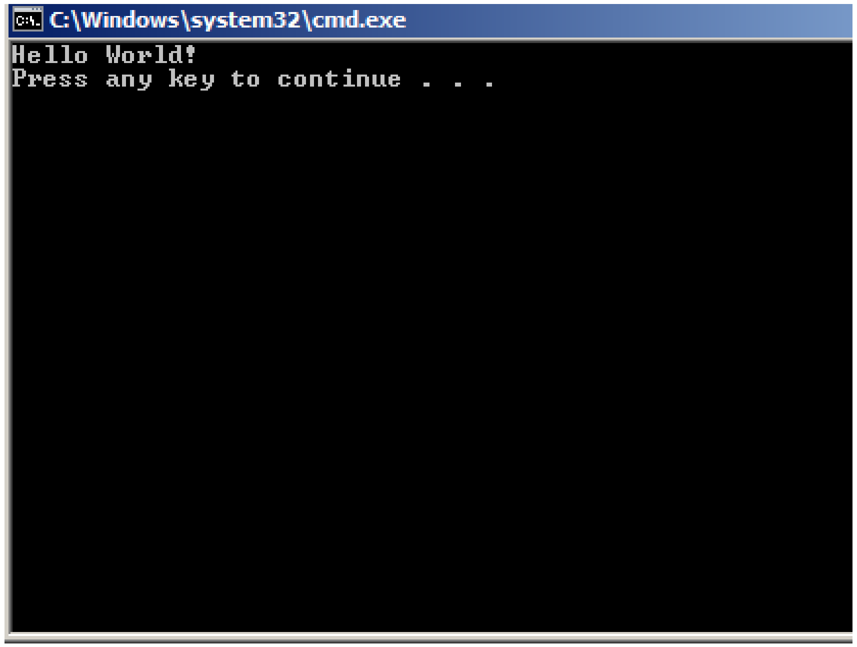
```
using System;

namespace ConsoleApp2
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Using keyword

- We cannot write everything by ourselves
- “using” keyword tells us which packages/libraries are included in our application
- For instance, in order to use Console, we need to have “using System”

Running The App

- In order to run the application, we need to press Debug/Run
- The application will be run in the console window
- The application will show everything, that is inside "Console.WriteLine"
- The WriteLine statement outputs the text



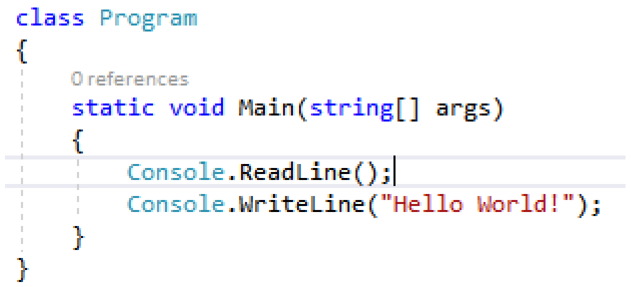
```
Hello World!
Press any key to continue . . .
```

Functions of Program

- We saw one of the important functions of the program: Output
- Any program has three very important functions:
  - Input – when the program gets information from outside world
  - Processing – when the program decides what to do with the information and does it
  - Output – when the program shows to the user what it has done with the information

Performing Input

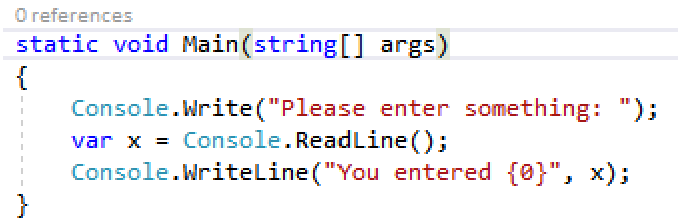
- There are console functions for reading data
- If we run this code, the program will be waiting for input
- There are two problems:
  - We do not know what we input ☹
  - We do not know how to get what we input and use it ☹



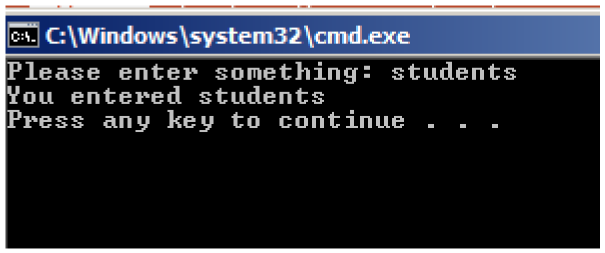
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.ReadLine();
        Console.WriteLine("Hello World!");
    }
}
```

Enhancements to a Program

- We wrote something before reading data. This something is called *prompt*
- Write does not change line so we can output after prompt
- We wrote the result of what we input into something called x
- We output to the console our text and 'x' together



```
0 references
static void Main(string[] args)
{
    Console.Write("Please enter something: ");
    var x = Console.ReadLine();
    Console.WriteLine("You entered {0}", x);
}
```



C:\Windows\system32\cmd.exe

```
Please enter something: students
You entered students
Press any key to continue . . .
```

Variables

- In the real life we classify information into text and numbers. That is how we write.
- Text ca be written in various languages and consists of letters
- Numbers can be whole, fractures of whole or combination of the two
- Data in the program is stored in bits and bytes, where bit can be one or zero and byte contains 8 bits. *<u>All computer storage is divided into bytes</u>*

Two Words on Binary System

- Binary system writes every number as ones and zeroes. Every position represents the power of 2 multiplied on 1 or 0 stored in the position
- Number, written as 10101001 can be translated as

$$2^7 * 1 + 2^6 * 0 + 2^5 * 1 + 2^4 * 0 + 2^3 * 1 + 2^2 * 0 + 2^1 * 0 + 2^0 * 1$$

- Calculating this expression we will get:

128+0+32+0+8+0+0+1=169

- One byte can store any number from 0 to 255

Variables (Numbers)

- Whole numbers occupy various number of bytes
  - Byte – 1
  - Short – 2
  - Integer – 4
  - Long Integer – 8
- Decimal numbers divide their storage between whole and fractions
- Fractures store only mantissa multiplied powers of 10 (ex: 10.123 will be stored as 0.10123*10^2)

Variables (Strings)

- Computer does not know what the strings are
- In the computer, strings are stored as bytes
- At the beginning it was enough to map every byte into a character and each character was a number from 0 to 255 (ASCII codes)
- Now, when we write in the large number of languages every character is represented by 2 bytes (Unicode)

Variables in C#

Numbers

- Whole numbers are *byte, short, int, long*
- Decimal numbers are *decimal, money (8 bytes)*
- Fraction numbers are *float(4 bytes), double (8 bytes)*

Strings

- We have variables of type *string*

Back to Programming

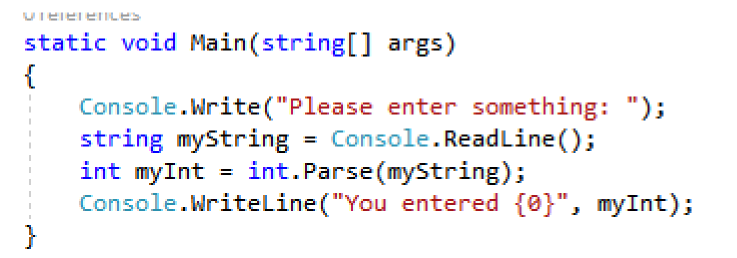
- Our program has one variable ‘x’
- We used implicit type ‘var’ for this variable because we did not know what type it must represent
- ‘var’ takes the first type it is assigned. It cannot mutate. Please, do not use it unless it is required!!!!!!
- In our program x can only be of type ‘string’ because ReadLine can only read text

Type Conversion

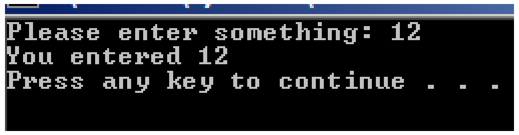
- We do not always need strings so we need to convert variable of type string into types, that we require
- C# has conversion functions, that we need:
- Every datatype has ability to parse string into its own type of data
- All datatypes can be converted into string

Type Conversion

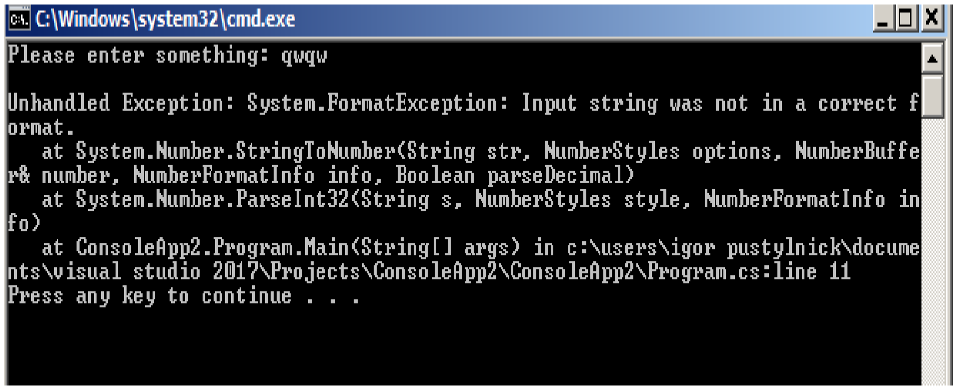
- We can use int.Parse to parse our integer
- If we try to enter something non-numeric, the error will occur and the program will die
- We will talk about exceptions (errors) later in this course



```
static void Main(string[] args)
{
    Console.Write("Please enter something: ");
    string myString = Console.ReadLine();
    int myInt = int.Parse(myString);
    Console.WriteLine("You entered {0}", myInt);
}
```



```
Please enter something: 12
You entered 12
Press any key to continue . . .
```



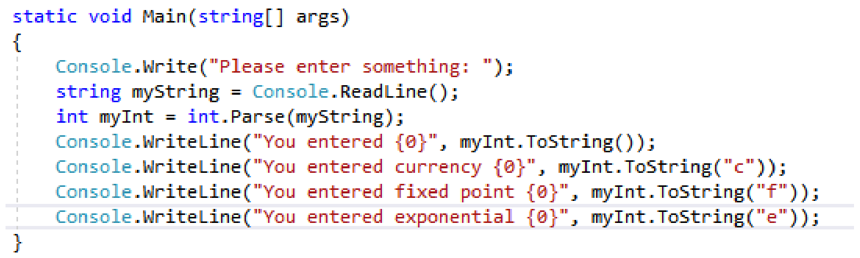
```
C:\Windows\system32\cmd.exe
Please enter something: qwqw

Unhandled Exception: System.FormatException: Input string was not in a correct f
ormat.
   at System.Number.StringToNumber(String str, NumberStyles options, NumberBuffe
r& number, NumberFormatInfo info, Boolean parseDecimal)
   at System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo in
fo)
   at ConsoleApp2.Program.Main(String[] args) in c:\users\igor pustylnick\docume
nts\visual studio 2017\Projects\ConsoleApp2\ConsoleApp2\Program.cs:line 11
Press any key to continue . . .
```

Converting to String

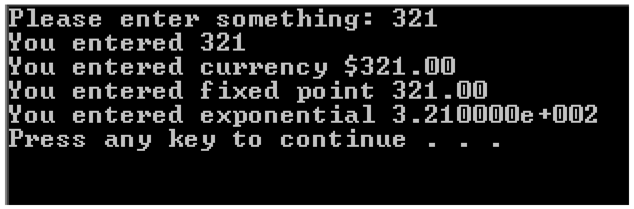
- Every variable can be converted back to string by typing x.ToString()
- You can make it a bit more interesting by adding the format specifier.
- You can read about formats in https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings

Using ToString

- We entered integer and by default it remained integer
- If we add ToString, we can get different notations



```
static void Main(string[] args)
{
    Console.Write("Please enter something: ");
    string myString = Console.ReadLine();
    int myInt = int.Parse(myString);
    Console.WriteLine("You entered {0}", myInt.ToString());
    Console.WriteLine("You entered currency {0}", myInt.ToString("c"));
    Console.WriteLine("You entered fixed point {0}", myInt.ToString("f"));
    Console.WriteLine("You entered exponential {0}", myInt.ToString("e"));
}
```

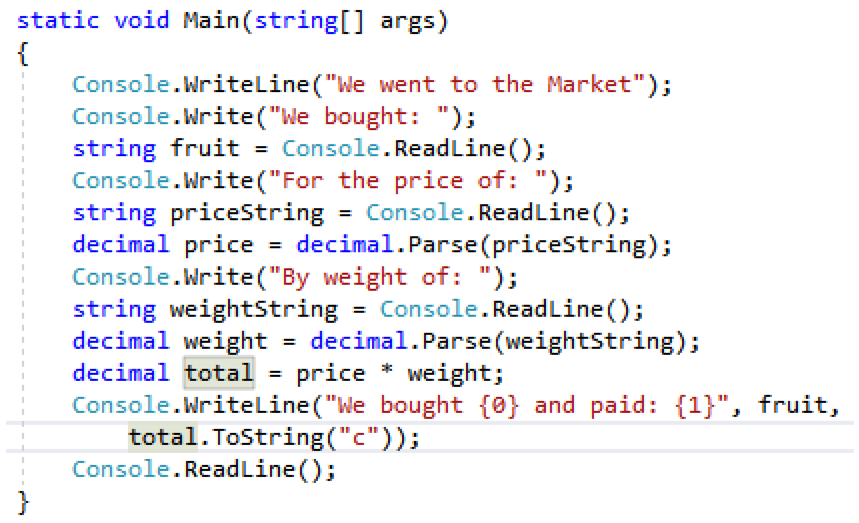


```
Please enter something: 321
You entered 321
You entered currency $321.00
You entered fixed point 321.00
You entered exponential 3.210000e+002
Press any key to continue . . .
```

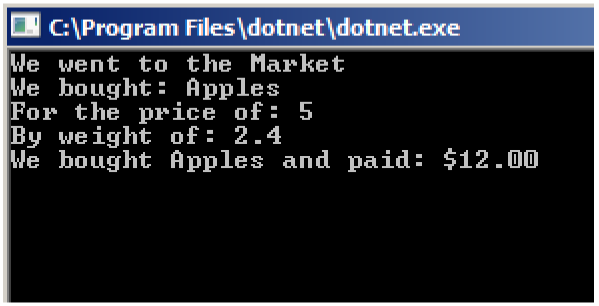
Math Operators

- The majority of the programs need to calculate something. For this, we have regular operators (+, -, *, /).
- % operator returns a remainder of the division, i.e. 5%2=1
- If you use '+' with strings, the right operand will be joined to the end of the left operand

Putting All Together



```
static void Main(string[] args)
{
    Console.WriteLine("We went to the Market");
    Console.Write("We bought: ");
    string fruit = Console.ReadLine();
    Console.Write("For the price of: ");
    string priceString = Console.ReadLine();
    decimal price = decimal.Parse(priceString);
    Console.Write("By weight of: ");
    string weightString = Console.ReadLine();
    decimal weight = decimal.Parse(weightString);
    decimal total = price * weight;
    Console.WriteLine("We bought {0} and paid: {1}", fruit,
        total.ToString("c"));
    Console.ReadLine();
}
```



C:\Program Files\dotnet\dotnet.exe

```
We went to the Market
We bought: Apples
For the price of: 5
By weight of: 2.4
We bought Apples and paid: $12.00
```

Important Notes:

- Every statement ends with ‘;’
- The statement starts at previous ‘;’ and runs to its own ending ‘;’ through many lines
- The last ReadLine prevents the program to finish.
- WriteLine can have more than one placeholder, all enumerated starting from zero. The first variable relates to {0}, second to {1}
- Numeric variables are converted to string automatically in WriteLine

Last Tips

You may have noticed:

- All variables must have informative names starting with lowercase letter (fruit, weight, etc.)
- If your variable uses more than one word, use camel case notation (weightString)
- Variables x, y, z are explicitly forbidden, apart from the cases that I will specify!!!!!!!

Lab Task

You have to write a similar program where you will tell me:

- Which country you flied from (input)
- How many bags you had with you (Input)
- What was the total weight of your luggage (Input)
- The final output must look like:

"I flew from 'xxx' having 'yyy' bags with average weight of 'zzz'"