**What is C#**

* A programming language
* A syntax that allow to give instructions to the computer

C# features:

* Cutting edge language
* Extremely powerful
* Easy to learn
* Easy to read and understand
* Object-oriented

**What is .NET Frmework**

* Environment for execution of .NET programs
* Powerful library of classes
* Programming model
* Common execution engine for many programming languages

**What is CLR – The Heart of .NET Framework**

Common Language Runtime (CLR)

* Managed execution environment
  + Executes .NET applications
  + Controls the execution process
* Automatic memory management (garbage collection)
* Programming languages integration
* Multiple versions support for assemblies
* Integrated type safety and security

**What is Framework Class Library (FCL)**

Provides basic functionality to developers:

* Console applications
* WPF and Silverlight rich-media applications
* Windows Forms GUI applications
* Web applications (dynamic Web sites)
* Web services, communication and workflow
* Server & desktop applications
* Applications for mobile devices

**What is Visual Studio?**

* Compiling, Running and Debugging C# Programs
* Visual Studio is an Integrated Development Environment (IDE)
* Development tool that helps us to:
* Write code
* Design user interface
* Compile code
* Execute / test / debug applications
* Browse the help
* Manage project's files

**What is Compiling Source Code**

The process of compiling includes:

* Syntactic checks
* Type safety checks
* Translation of the source code to lower level language (MSIL)
* Creating of executable files (assemblies)

You can start compilation by

* Using Build->Build Solution/Project
* Pressing [F6] or [Shift+Ctrl+B]

**What is MSDN Library?**

* Complete documentation of all classes and their functionality
  + With descriptions of all methods, properties, events, etc.
  + With code examples
* Related articles
* Library of samples

**Variables** have name, data type and value

**What Is a Data Type?**

* A domain of values of similar characteristics
* Defines the type of information stored in the computer memory (in a variable)

**The default value of integer types is:**

* 0 – for integer types
* except 0L – for the long type

sbyte (signed 8-bit)

* Min: -128
* Max: 127

byte (unsigned 8-bit)

* Min: 0
* Max: 255

short (signed 16-bit)

* Min: -32,768
* Max: 32,767

ushort (unsigned 16-bit)

* Min: 0
* Max: 65,535

int (signed 32-bit)

* Min: -2,147,483,648
* Max: 2,147,483,647

uint (unsigned 32-bit)

* Min:0
* Max: 4,294,967,295

long (signed 64-bit)

* Min: -9,223,372,036,854,775,808
* Max: 9,223,372,036,854,775,807

ulong (unsigned 64-bit)

* Min: 0
* Max: 18,446,744,073,709,551,615

**Floating-point types are:**

float (32-bits)

* Min: ±1.5 × 10-45
* Max: ±3.4 × 1038
* Precision: 7 digits

double (64-bits)

* Min: ±5.0 × 10-324
* Max: ±1.7 × 10308
* Precision: 15-16 digits

The default value of floating-point types:

* Is 0.0F for the float type
* Is 0.0D for the double type

Real numbers are by default interpreted as double!

decimal (128-bits)

* Min: ±1,0 × 10-28
* Max: ±7,9 × 1028 -Precision: 28-29 digits

**The character data type:**

* Represents symbolic information
* Is declared by the char keyword
* Gives each symbol a corresponding integer code
* Has a '\0' default value
* Takes 16 bits of memory (from U+0000 to U+FFFF)

**The object type:**

* Is declared by the object keyword
* Is the base type of all other types
* Can hold values of any type

**What Is a Variable**?

* Placeholder of information that can usually be changed at run-time
* Variables allow you to:
  + Store information
  + Retrieve the stored information
  + Manipulate the stored information

**Literals are:**

* Representations of values in the source code
* There are six types of literals
  + Boolean
  + Integer
  + Real
  + Character
  + String
  + The null literal

The **'0x' and '0X'** prefixes mean a hexadecimal value 0xA8F1

* int numberInHex = -0x10;
* int numberInDec = -16;

**The 'u' and 'U'** suffixes mean a ulong or uint type 12345678U

**The 'l' and 'L'** suffixes mean a long or ulong type 9876543L

**Escaping sequences are:**

Means of presenting a symbol that is usually interpreted otherwise (like ')

Means of presenting system symbols (like the new line symbol)

**Common escaping sequences are:**

* \' for single quote \" for double quote
* \\ for backslash \n for new line
* \uXXXX for denoting any other Unicode symbol
* Benefits of quoted strings (the @ prefix):In quoted strings \" is used instead of ""!

Escape Everithing in the string without double quotes "

We use " to escape "

**What is an Operator?**

* Operator is an operation performed over data at runtime
* Takes one or more arguments (operands)
* Produces a new value
* Operators have precedence
* Precedence defines which will be evaluated first
* Expressions are sequences of operators and operands that are evaluated to a single value

**Operators in C# :**

* Unary – take one operand
* Binary – take two operands
* Ternary (?:) – takes three operands
* Except for the assignment operators, all binary operators are left-associative
* The assignment operators and the conditional operator (?:) are right-associative

Precedence Operators

Highest () . []

++ -- (postfix) new typeof

++ -- (prefix) + - (unary) ! ~

\* / %

+ -

<< >>

< > <= >= is as

== !=

Lower &

Higher ^

|

&&

||

?:

Lowest = \*= /= %= += -= <<= >>= &= ^= |=

**Which of the following statements are TRUE about the .NET CLR?**

* **It provides a language-neutral development & execution environment.**
* **2.It ensures that an application would not be able to access memory that it is not authorized to access.**
* **3.It provides services to run "managed" applications.**
* **4.The resources are garbage collected.**

5.It provides services to run "unmanaged" applications.

* **Indicate the incorrect purpose of the Framework Class Library (FCL)?**
  1. **Provides functionality for creating console applications**
  2. **Provides functionality for creating WPF and Silverlight rich-media applications**
  3. **Provides functionality for creating iOS and Android applications**
  4. **Provides functionality for creating Windows Forms GUI applications**

**Provides functionality for creating web applications (dynamic Web sites)**

* **What will Console.ReadLine() return when there aren't any available lines to read?**

NULL

**!!!! False and True are printed with first capital letter on the console**

**int i = 1;**

**int j = 1;**

**switch (i)**

**{**

**case 1:**

**i = 1;**

**break;**

**case j:**

**i = 2;**

**break; The case value must be a constant**

**case i:**

**i = 3;**

**break;**

**default:**

**i = 4;**

**break;**

**}**

**Console.WriteLine(i); => Compilation error**

**--------------------------------------------------------------------------------------------------**

**int number = 5;**

**if (number++ == ++number)**

**{**

**Console.WriteLine(number + 1);**

**}**

**else**

**{**

**Console.WriteLine(number + 2);**

**}**

**cw(number)=> 9;**

**int count = 0;**

**for (int i = 1, j = 2; i < j; i++, j++)**

**{**

**count++;**

**if (i == 3) i++; break;**

**}**

**Console.WriteLine(count); => 1**

**count =1**

**int sum = 0;**

**while (sum < 10)**

**for (int i = 0; i <= 2; i++)**

**sum += i;**

**Console.WriteLine(sum); =>12**

**Operator ^**

Operation **^ ^ ^ ^**

Operand1 **0 0 1 1**

Operand2 **0 1 0 1**

**Result 0 1 1 0**

**Ако нещо е null му задаваме нова стойност за да не ни се прецакат сметките**

**int? x = null;**

**int y = x ?? -1; =>у=-1**

**same as int a = x!=null?x:-1;**

**Implicit type conversion**

Automatic conversion of value of one data type to value of another data type

Allowed when no loss of data is possible

"Larger" types can implicitly take values of smaller "types"

Example:

**int i = 5;**

**long l = i;**

**Explicit type conversion**

Manual conversion of a value of one data type to a value of another data type

Allowed only explicitly by (type) operator

Required when there is a possibility of loss of data or precision

Example:

**long l = 5;**

**int i = (int) l;**

**De Morgan laws**

**!!A equals A**

**!(A || B) equals !A && !B**

**!(A && B) equals !A || !B**

**Factorial**

static void Main()

{

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

// Calculate n! = 1 \* 2 \* ... \* n

int result = 1;

while (true)

{

if (n == 1)

break;

result \*= n;

n--;

}

Console.WriteLine("n! = " + result);

string numberAsString = Console.ReadLine();

int n = Convert.ToInt32(numberAsString);

int factorial = 1;

do

{

factorial \*= n;

n--;

}

while (n > 0);

Console.WriteLine("n! = " + factorial);

}

**Drawing a diagonal line on the console**

static void Main()

{

for (int x = 0, y = 0; x < 10; x++, y++)

{

Console.SetCursorPosition(x, y);

Console.Write('\*');

}

}

**Jump statements are:**

break, continue, goto

How continue works?

In while and do-while loops jumps to the test expression

In for loops jumps to the update expression

To exit an inner loop use break

To exit outer loops use goto with a label

Avoid using goto! (it is considered harmful)