telerik

Primitive Data Types and Variables

Creating and Running Your First C# Program

Svetlin Nakov

Telerik Corporation

www.telerik.com

telerik

Table of Contents

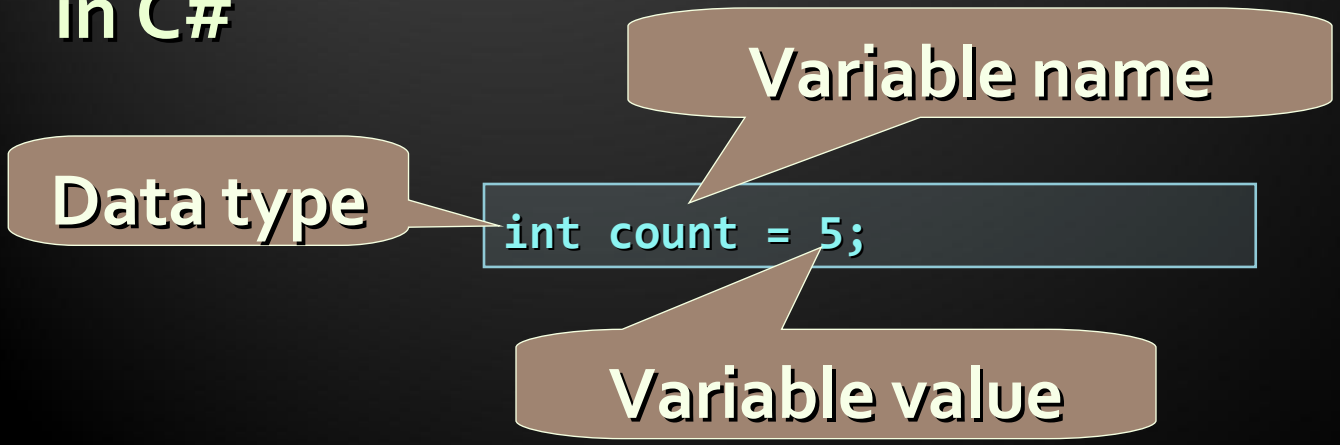
telerik

Primitive Data Types

telerik

How Computing Works?

- Computers are machines that process data
  - Data is stored in the computer memory in variables
  - Variables have name, data type and value
- Example of variable definition and assignment in C#



telerik

What Is a Data Type?

- A data type:
  - Is a domain of values of similar characteristics
  - Defines the type of information stored in the computer memory (in a variable)
- Examples:
  - Positive integers: 1, 2, 3, …
  - Alphabetical characters: a, b, c, …
  - Days of week: Monday, Tuesday, …

telerik

Data Type Characteristics

- A data type has:
  - Name (C# keyword or .NET type)
  - Size (how much memory is used)
  - Default value
- Example:
  - Integer numbers in C#
  - Name: `int`
  - Size: 32 bits (4 bytes)
  - Default value: 0

telerik

Integer Types

telerik

What are Integer Types?

- Integer types:
  - Represent whole numbers
  - May be signed or unsigned
  - Have range of values, depending on the size of memory used
- The default value of integer types is:
  - `0` – for integer types, except
  - `0L` – for the long type

telerik

Integer Types

- Integer types are:
  - `sbyte` (-128 to 127): signed 8-bit
  - `byte` (0 to 255): unsigned 8-bit
  - `short` (-32,768 to 32,767): signed 16-bit
  - `ushort` (0 to 65,535): unsigned 16-bit
  - `int` (-2,147,483,648 to 2,147,483,647): signed 32-bit
  - `uint` (0 to 4,294,967,295): unsigned 32-bit

telerik

Integer Types (2)

- More integer types:
  - `long` (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807): signed 64-bit
  - `ulong` (0 to 18,446,744,073,709,551,615): unsigned 64-bit

telerik

Measuring Time – Example

- Depending on the unit of measure we may use different data types:

```
byte centuries = 20;    // Usually a small number
ushort years = 2000;
uint days = 730480;
ulong hours = 17531520; // May be a very big number
Console.WriteLine("{0} centuries is {1} years, or
{2} days, or {3} hours.", centuries, years, days,
hours);
```

telerik

Integer Types

Live Demo

telerik

Floating-Point and Decimal Floating-Point Types

telerik

What are Floating-Point Types?

- Floating-point types:
  - Represent real numbers
  - May be signed or unsigned
  - Have range of values and different precision depending on the used memory
  - Can behave abnormally in the calculations

telerik

Floating-Point Types

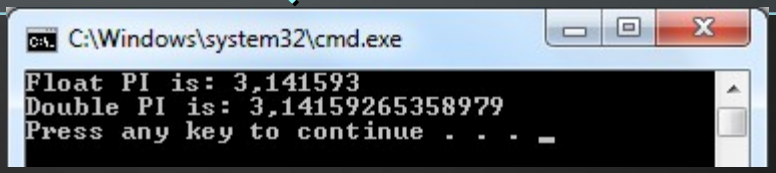
- Floating-point types are:
  - `float` ($\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$): 32-bits, precision of 7 digits
  - `double` ($\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$): 64-bits, precision of 15-16 digits
- The default value of floating-point types:
  - Is `0.0F` for the `float` type
  - Is `0.0D` for the `double` type

telerik

PI Precision – Example

- See below the difference in precision when using `float` and `double`:

```
float floatPI = 3.14159265358793238f;
double doublePI = 3.14159265358793238;
Console.WriteLine("Float PI is: {0}", floatPI);
Console.WriteLine("Double PI is: {0}", doublePI);
```



- NOTE: The "f" suffix in the first statement!
  - Real numbers are by default interpreted as `double`!
  - One should explicitly convert them to `float`

telerik

Abnormalities in the Floating-Point Calculations

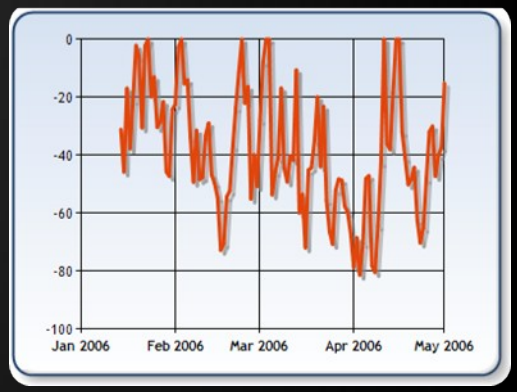
- Sometimes abnormalities can be observed when using floating-point numbers
  - Comparing floating-point numbers can not be performed directly with the == operator
- Example:

```
double a = 1.0f;
double b = 0.33f;
double sum = 1.33f;
bool equal = (a+b == sum); // False!!!
Console.WriteLine("a+b={0}  sum={1}  equal={2}",
    a+b, sum, equal);
```

telerik

Decimal Floating-Point Types

- There is a special decimal floating-point real number type in C#:
  - `decimal` ($\pm 1{,}0 \times 10^{-28}$ to $\pm 7{,}9 \times 10^{28}$): 128-bits, precision of 28-29 digits
  - Used for financial calculations
  - No round-off errors
  - Almost no loss of precision



- The default value of `decimal` type is:
  - `0.0M` (`M` is the suffix for decimal numbers)

telerik

Floating-Point and Decimal Floating-Point Types

Live Demo

telerik

Boolean Type

telerik

The Boolean Data Type

- The Boolean data type:
  - Is declared by the `bool` keyword
  - Has two possible values: `true` and `false`
  - Is useful in logical expressions
- The default value is `false`

telerik

Boolean Values – Example

- Example of boolean variables taking values of `true` or `false`:

```
int a = 1;
int b = 2;

bool greaterAB = (a > b);

Console.WriteLine(greaterAB);  // False

bool equalA1 = (a == 1);

Console.WriteLine(equalA1);    // True
```

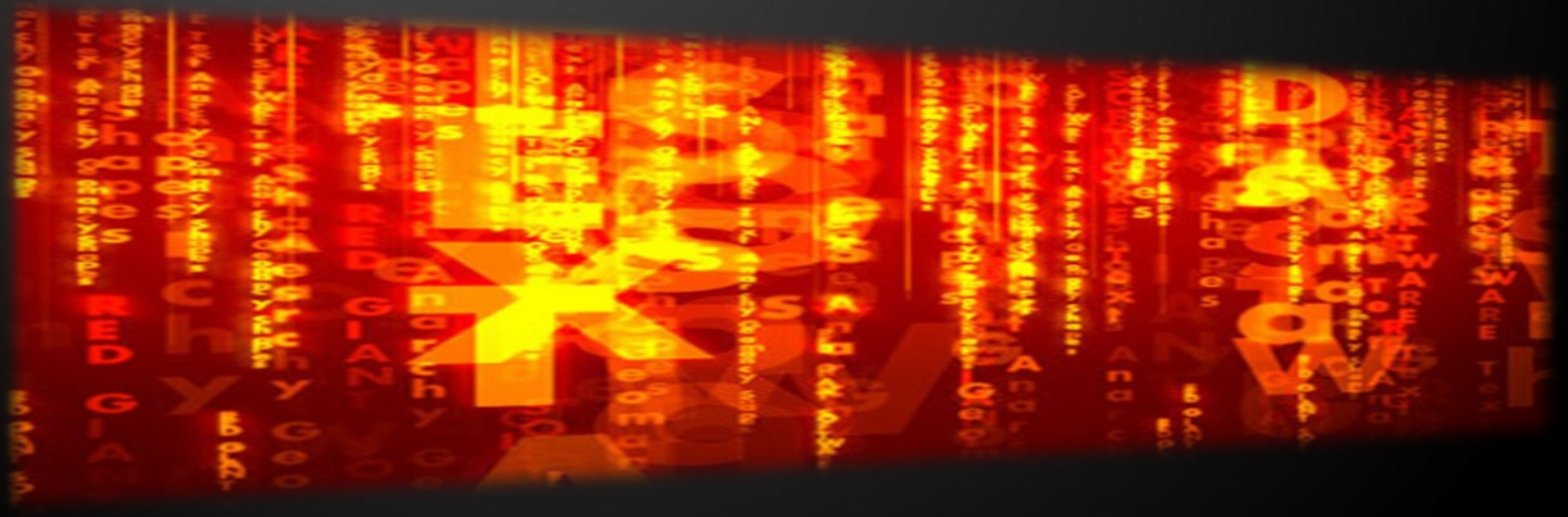
telerik

Boolean Type

Live Demo

telerik

Character Type



telerik

The Character Data Type

- 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`)

telerik

Characters and Codes

- The example below shows that every symbol has an its unique Unicode code:

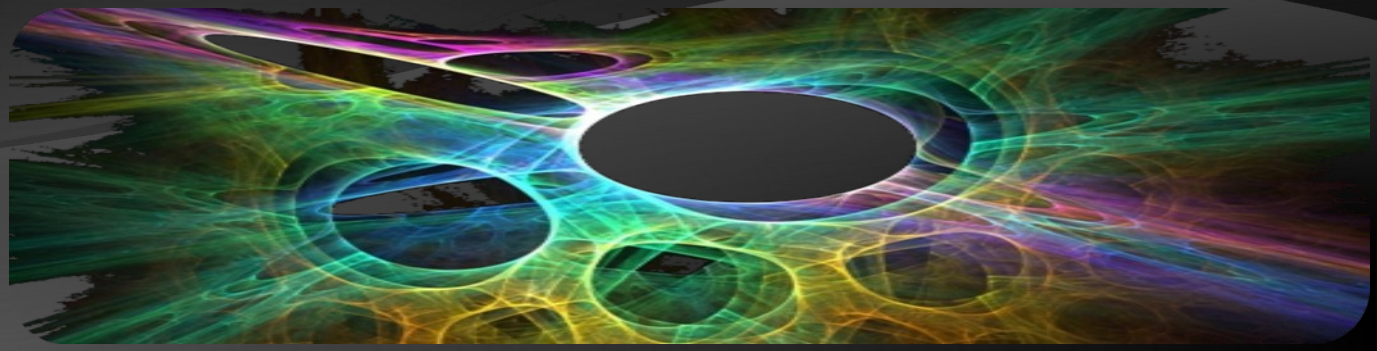
```
char symbol = 'a';
Console.WriteLine("The code of '{0}' is: {1}",
    symbol, (int) symbol);

symbol = 'b';
Console.WriteLine("The code of '{0}' is: {1}",
    symbol, (int) symbol);

symbol = 'A';
Console.WriteLine("The code of '{0}' is: {1}",
    symbol, (int) symbol);
```

telerik

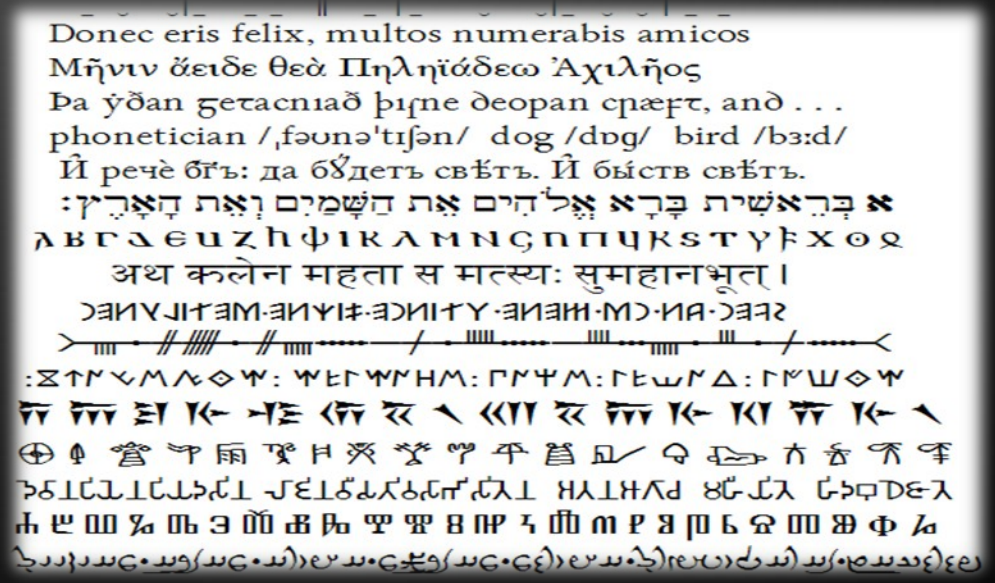
Character Type

Live Demo

telerik



String Type



telerik

The String Data Type

- The string data type:
  - Represents a sequence of characters
  - Is declared by the `string` keyword
  - Has a default value `null` (no value)
- Strings are enclosed in quotes:

```
string s = "Microsoft .NET Framework";
```

- Strings can be concatenated
  - Using the + operator

telerik

Saying Hello – Example

- Concatenating the two names of a person to obtain his full name:

```
string firstName = "Ivan";
string lastName = "Ivanov";
Console.WriteLine("Hello, {0}!\n", firstName);

string fullName = firstName + " " + lastName;
Console.WriteLine("Your full name is {0}.",
  fullName);
```

- NOTE: a space is missing between the two names! We have to add it manually

telerik

String Type

Live Demo

telerik

Object Type

telerik

The Object Type

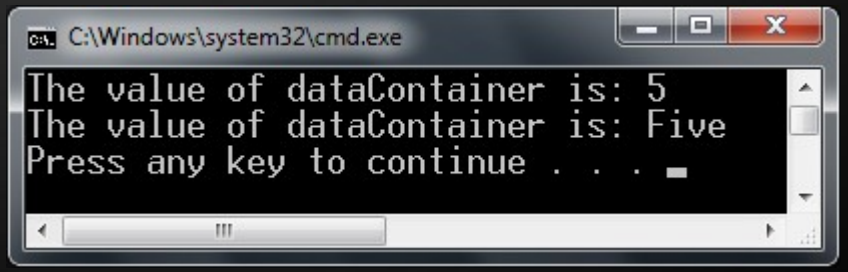
- The object type:
  - Is declared by the `object` keyword
  - Is the base type of all other types
  - Can hold values of any type

telerik

Using Objects

- Example of an object variable taking different types of data:

```
object dataContainer = 5;
Console.Write("The value of dataContainer is: ");
Console.WriteLine(dataContainer);

dataContainer = "Five";
Console.Write("The value of dataContainer is: ");
Console.WriteLine(dataContainer);
```



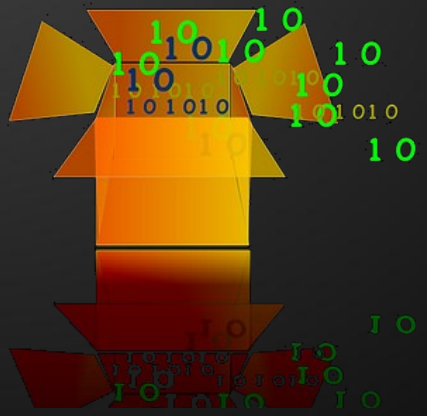
telerik

Objects

Live Demo

telerik

Introducing Variables



telerik

What Is a Variable?

- A variable is a:
  - 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

telerik

Variable Characteristics

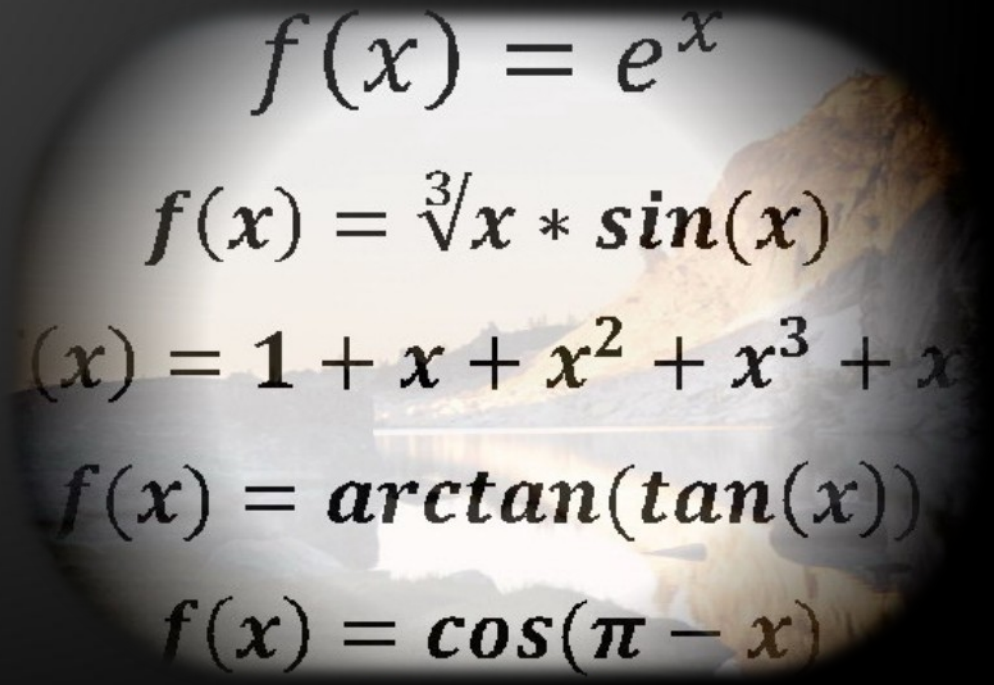
- A variable has:
  - Name
  - Type (of stored data)
  - Value
- Example:

```
int counter = 5;
```

  - Name: `counter`
  - Type: `int`
  - Value: `5`

telerik

Declaring And Using Variables



telerik

Declaring Variables

- When declaring a variable we:
  - Specify its type
  - Specify its name (called identifier)
  - May give it an initial value
- The syntax is the following:

```
<data_type> <identifier> [= <initialization>];
```

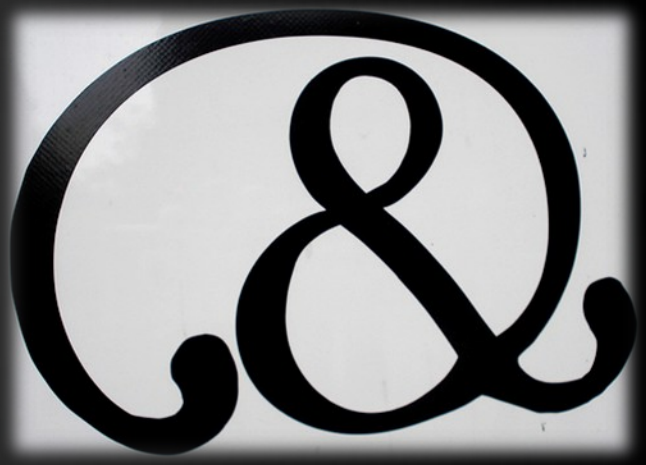
- Example:

```
int height = 200;
```

telerik

Identifiers

- Identifiers may consist of:
  - Letters (Unicode)
  - Digits [0-9]
  - Underscore "_"
- Identifiers
  - Can begin only with a letter or an underscore
  - Cannot be a C# keyword



telerik

Identifiers (2)

- Identifiers
  - Should have a descriptive name
  - It is recommended to use only Latin letters
  - Should be neither too long nor too short
- Note:
  - In C# small letters are considered different than the capital letters (case sensitivity)

telerik

Identifiers – Examples

- Examples of correct identifiers:

```
int New = 2; // Here N is capital
int _2Pac; // This identifiers begins with _

string поздрав = "Hello"; // Unicode symbols used
// The following is more appropriate:
string greeting = "Hello";

int n = 100; // Undescriptive
int numberOfClients = 100; // Descriptive

// Overdescriptive identifier:
int numberOfPrivateClientOfTheFirm = 100;
```

- Examples of incorrect identifiers:

```
int new;        // new is a keyword
int 2Pac;       // Cannot begin with a digit
```

telerik

Assigning Values To Variables

telerik

Assigning Values

- Assigning of values to variables
  - Is achieved by the = operator
- The = operator has
  - Variable identifier on the left
  - Value of the corresponding data type on the right
  - Could be used in a cascade calling, where assigning is done from right to left

telerik

Assigning Values – Examples

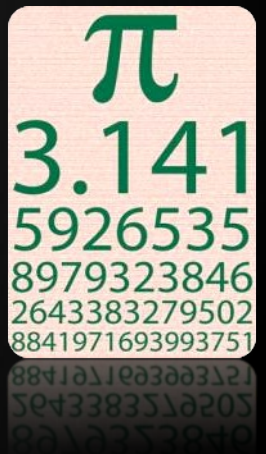
- Assigning values example:

```
int firstValue = 5;
int secondValue;
int thirdValue;

// Using an already declared variable:
secondValue = firstValue;

// The following cascade calling assigns
// 3 to firstValue and then firstValue
// to thirdValue, so both variables have
// the value 3 as a result:

thirdValue = firstValue = 3; // Avoid this!
```

telerik

Initializing Variables

- Initializing
  - Is assigning of initial value
  - Must be done before the variable is used!
- Several ways of initializing:
  - By using the new keyword
  - By using a literal expression
  - By referring to an already initialized variable



telerik

Initialization – Examples

- Example of some initializations:

```
// The following would assign the default
// value of the int type to num:
int num = new int(); // num = 0

// This is how we use a literal expression:
float heightInMeters = 1.74f;

// Here we use an already initialized variable:
string greeting = "Hello World!";
string message = greeting;
```

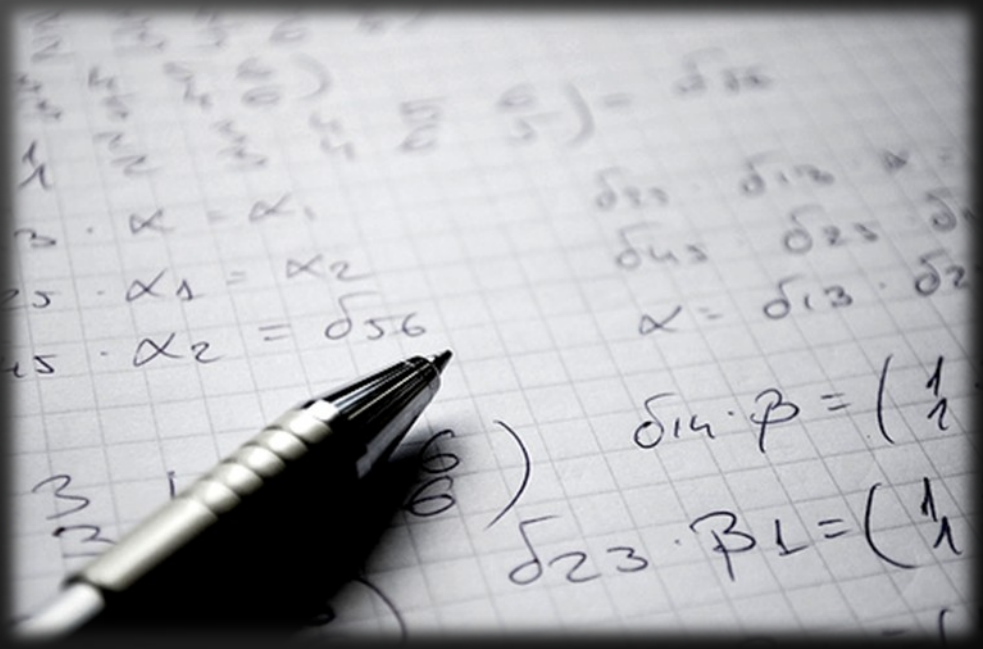
telerik

Assigning and Initializing Variables

Live Demo

telerik

Literals



telerik

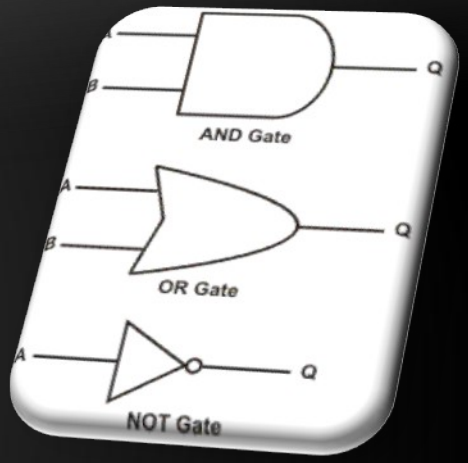
What are Literals?

- Literals are:
  - Representations of values in the source code
- There are six types of literals
  - Boolean
  - Integer
  - Real
  - Character
  - String
  - The `null` literal

telerik

Boolean and Integer Literals

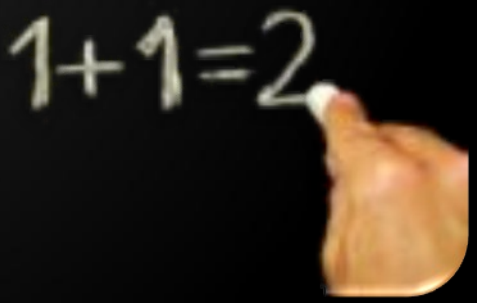
- The boolean literals are:
  - `true`
  - `false`
- The integer literals:
  - Are used for variables of type `int`, `uint`, `long`, and `ulong`
  - Consist of digits
  - May have a sign (+, -)
  - May be in a hexadecimal format



telerik

Integer Literals

- Examples of integer literals
  - The `'0x'` and `'0X'` prefixes mean a hexadecimal value, e.g. `0xA8F1`
  - The `'u'` and `'U'` suffixes mean a `ulong` or `uint` type, e.g. `12345678U`
  - The `'l'` and `'L'` suffixes mean a `long` or `ulong` type, e.g. `9876543L`



telerik

Integer Literals – Example

```
// The following variables are
// initialized with the same value:
int numberInHex = -0x10;
int numberInDec = -16;

// The following causes an error,
because 234u is of type uint
int unsignedInt = 234u;

// The following causes an error,
because 234L is of type long
int longInt = 234L;
```

- Note: the letter 'l' is easily confused with the digit '1' so it's better to use 'L'!!!

telerik

Real Literals

- The real literals:
  - Are used for values of type `float`, `double` and `decimal`
  - May consist of digits, a sign and "`.`"
  - May be in exponential notation: `6.02e+23`
- The "`f`" and "`F`" suffixes mean `float`
- The "`d`" and "`D`" suffixes mean `double`
- The "`m`" and "`M`" suffixes mean `decimal`
- The default interpretation is `double`

telerik

Real Literals – Example

- Example of incorrect `float` literal:

```
// The following causes an error
// because 12.5 is double by default
float realNumber = 12.5;
```

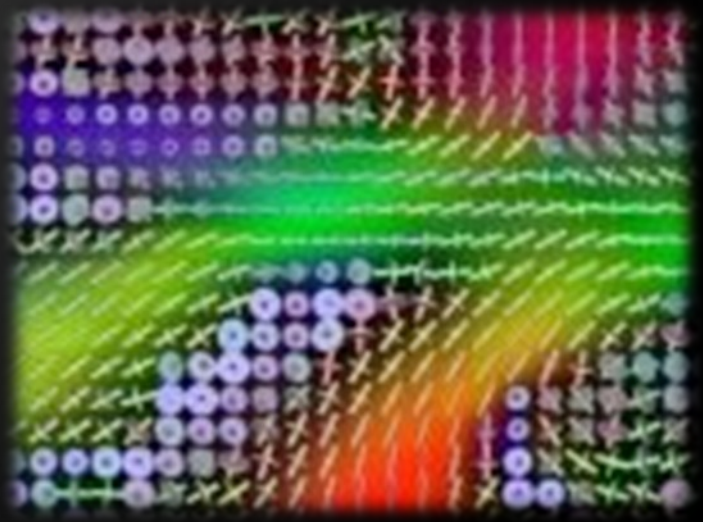
- A correct way to assign floating-point value (using also the exponential format):

```
// The following is the correct
// way of assigning the value:
float realNumber = 12.5f;

// This is the same value in exponential format:
realNumber = 1.25e+7f;
```

telerik

Character Literals

- The character literals:
  - Are used for values of the `char` type
  - Consist of two single quotes surrounding the character value: `'<value>'`
- The value may be:
  - Symbol
  - The code of the symbol
  - Escaping sequence



telerik

Escaping Sequences

- 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

telerik

Character Literals – Example

- Examples of different character literals:

```
char symbol = 'a'; // An ordinary symbol
symbol = '\u006F'; // Unicode symbol code in
                   // a hexadecimal format
symbol = '\u8449'; // 葉 (Leaf in Traditional Chinese)
symbol = '\''; // Assigning the single quote symbol
symbol = '\\'; // Assigning the backslash symbol
symbol = '\n'; // Assigning new line symbol
symbol = '\t'; // Assigning TAB symbol
symbol = "a"; // Incorrect: use single quotes
```

telerik

String Literals

- String literals:
  - Are used for values of the string type
  - Consist of two double quotes surrounding the value: "<value>"
  - May have a @ prefix which ignores the used escaping sequences: @"<value>"
- The value is a sequence of character literals

```
string s = "I am a sting literal";
```

telerik

String Literals – Example

- Benefits of quoted strings (the @ prefix):

```
// Here is a string literal using escape sequences
string quotation = "\"Hello, Jude\", he said.";
string path = "C:\\WINNT\\Darts\\Darts.exe";

// Here is an example of the usage of @
quotation = @"""Hello, Jimmy!"", she answered.";
path = @"C:\WINNT\Darts\Darts.exe";

string str = @"some
             text";
```

- In quoted strings \" is used instead of ""!

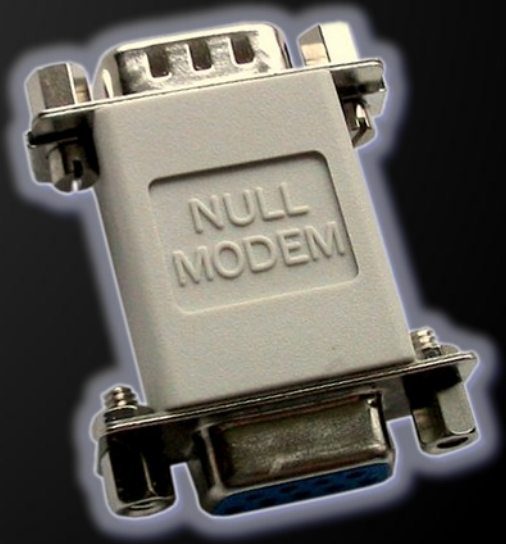
telerik

String Literals

Live Demo

telerik

Nullable Types



telerik

Nullable Types

- `Nullable` types are instances of the `System.Nullable` struct
  - Wrapper over the primitive types
  - E.g. `int?`, `double?`, etc.
- `Nullabe` type can represent the normal range of values for its underlying value type, plus an additional `null` value
- Useful when dealing with `Databases` or other structures that have default value `null`

telerik

Nullable Types – Example

Example with Integer:

```
int? someInteger = null;
Console.WriteLine(
  "This is the integer with Null value -> " + someInteger);
someInteger = 5;
Console.WriteLine(
  "This is the integer with value 5 -> " +  someInteger);
```

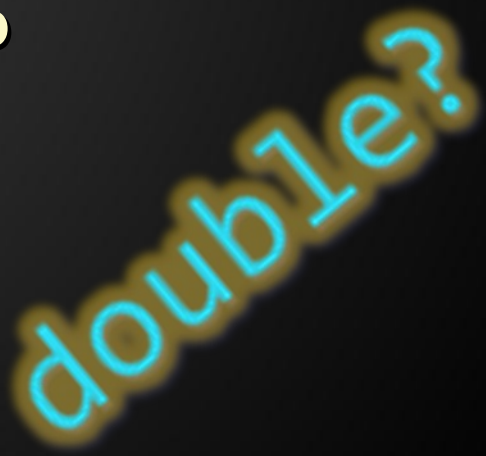
Example with Double:

```
double? someDouble = null;
Console.WriteLine(
  "This is the real number with Null value -> "
  + someDouble);
someDouble = 2.5;
Console.WriteLine(
  "This is the real number with value 5 -> " +
  someDouble);
```

telerik

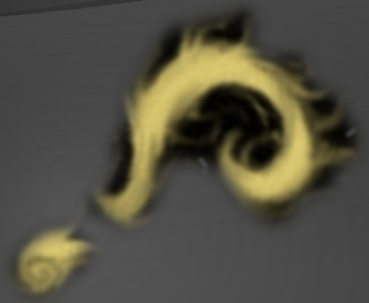
Nullable Types

Live Demo



telerik

Primitive Data Types and Variables



Questions?

telerik

Exercises

1. Declare five variables choosing for each of them the most appropriate of the types `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong` to represent the following values: 52130, -115, 4825932, 97, -10000.
2. Which of the following values can be assigned to a variable of type `float` and which to a variable of type `double`: 34.567839023, 12.345, 8923.1234857, 3456.091?
3. Write a program that safely compares floating-point numbers with precision of `0.000001`.

telerik

Exercises (2)

1. Declare an integer variable and assign it with the value 254 in hexadecimal format. Use Windows Calculator to find its hexadecimal representation.
2. Declare a character variable and assign it with the symbol that has Unicode code 72. Hint: first use the Windows Calculator to find the hexadecimal representation of 72.
3. Declare a boolean variable called `isFemale` and assign an appropriate value corresponding to your gender.

telerik

Exercises (3)

1. Declare two `string` variables and assign them with "Hello" and "World". Declare an `object` variable and assign it with the concatenation of the first two variables (mind adding an interval). Declare a third `string` variable and initialize it with the value of the object variable (you should perform type casting).
2. Declare two `string` variables and assign them with following value:

```
The "use" of quotations causes difficulties.
```

Do the above in two different ways: with and without using quoted strings.

telerik

Exercises (4)

1. Write a program that prints an isosceles triangle of 9 copyright symbols ©. Use Windows Character Map to find the Unicode code of the © symbol.
2. A marketing firm wants to keep record of its employees. Each record would have the following characteristics – first name, family name, age, gender (m or f), ID number, unique employee number (27560000 to 27569999). Declare the variables needed to keep the information for a single employee using appropriate data types and descriptive names.
3. Declare two integer variables and assign them with 5 and 10 and after that exchange their values.