HNDIT1012 Visual Application Programming

Week 4

C# Variables

Variables are containers for storing data values.

Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

Syntax

```
type variableName = value;
```

Where type is a C# type (such as int or string), and variableName is the name of the variable (such as x or name). The equal sign is used to assign values to the variable.

Eg:-

```
int myNum = 5;
double myDoubleNum = 5.99D;
char myLetter = 'D';
bool myBool = true;
string myText = "Hello";
```

```
int x = 5, y = 6, z = 50;
```

Declare Many Variables

To declare more than one variable of the **same type**, use a comma-separated list:

```
int x = 5, y = 6, z = 50;
```

C# Identifiers

All C# variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

It is recommended to use descriptive names in order to create understandable and maintainable code:

The general rules for naming variables

- Names can contain letters, digits and the underscore character (_)
- Names must begin with a letter
- Names should start with a lowercase letter and it cannot contain whitespace
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like C# keywords, such as int or double) cannot be used as names

```
string text = "This is a \"string\" in C#."; string str = "xyzdef\\rabc"; string path = "\\\\mypc\\ shared\\p
```

escaping character

C# includes escaping character \ (backslash) before these special characters to include in a string

Use backslash \ before double quotes and some special characters such as \,\n,\r,\t, etc. to include it in a string.

Eg:

string text = "This is a \"string\" in C#.";

string str = "xyzdef\\rabc";

string path = "\\\\mypc\\ shared\\project";

Escape Sequence	Represents
\a	Bell (alert)
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\'	Single quotation mark
\"	Double quotation mark
\\	Backslash
\?	Literal question mark
\ *ooo*	ASCII character in octal notation
\x *hh*	ASCII character in hexadecimal notation
\x *hhhh*	Unicode character in hexadecimal notation if this escape sequence is used in a wide-character constant or a Unicode string literal. For example, WCHAR f = L'\x4e00' or WCHAR b[] = L"The Chinese character for one is \x4e00".

Verbatim string

Verbatim string in C# allows a special characters and line brakes. Verbatim string can be created by prefixing @ symbol before double quotes.

```
string str = @"xyzdef\rabc";
string path = @"\\mypc\shared\project";
string email = @"test@test.com";
```

Control Structures in C#

The if Statement

Use the if statement to specify a block of C# code to be executed if a condition is True.

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
```

Eg:

```
int x = 20;
int y = 18;
if (x > y)
{
  Console.WriteLine("x is greater than y");
}
```

If --- else --

Use the else statement to specify a block of code to be executed if the condition is False.

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
else
{
  // block of code to be executed if the condition is False
}
```

```
int time = 20;
if (time < 18)
    {
      Console.WriteLine("Good day.");
    }
else
    {
      Console.WriteLine("Good evening.");
    }

// Outputs "Good evening."
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is False.

Syntax

```
if (condition1)
{
  // block of code to be executed if condition1 is True
}
else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is True
}
else
{
  // block of code to be executed if the condition1 is false and condition2 is False
}
```

```
int time = 22;
if (time < 10)
    {
     Console.WriteLine("Good morning.");
    }
else if (time < 20)
        {
         Console.WriteLine("Good day.");
        }
    else
        {
         Console.WriteLine("Good evening.");
        }
// Outputs "Good evening."
```

Short Hand If...Else (Ternary Operator)

There is also a short-hand if else, which is known as the ternary operator because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements:

Syntax

variable = (condition) ? expressionTrue : expressionFalse;

Eg:

```
int time = 20;
string result = (time < 18) ? "Good day." : "Good evening.";
Console.WriteLine(result);
```

Thank You