Lab#6

Introduction to Visual Studio .NET

Overview

In this lab, you will learn to use Visual Studio .NET 2005. As a starting point, you will learn how to operate and program in the Visual Studio .NET environment using Visual C# (pronounced Visual C Sharp).
Visual Studio .NET 2005 (VS .NET), code-named “Whidbey,” is an integrated development environment created and released by Microsoft in October 2005. It is a programming platform used for developing a variety of applications (e.g. Web Services, Web Applications, Windows Applications, and Mobile Applications). The environment itself is comprised of a number of languages that share a common set of classes and a development environment. The platform consists of Visual Basic .NET, Visual C#, Visual FoxPro, Visual J#, and Visual C++.

To better understand the .NET initiative and to help you see its components, we need to know what the .NET framework is and its goal.

Microsoft .NET Framework is integral Windows component for building and running all kinds of software, including Web-based applications, smart client applications, and XML Web services—components that facilitate integration by sharing data and functionality over a network through standard, platform-independent protocols such as XML (Extensible Markup Language), SOAP, and HTTP. The .NET Framework is composed of the common language runtime (CLR) and a set of common class libraries which enable developers to easily build and deploy applications.
Common Language Runtime (CLR): The common language runtime (CLR) is responsible for runtime services such as language integration, security enforcement, memory process, and thread management. In addition, the CLR reduces the amount of code that a developer must write to turn business logic into a reusable component.

Class Libraries: The class libraries are a set of unified classes that provides a common, consistent development interface across all languages supported by the .NET Framework. The class libraries consist of the base classes which provide standard functionality such as input/output, string manipulation, security management, network communications, thread management, text management, and user interface design features. It also includes the ADO.NET classes which enable developers to interact with databases. Furthermore, the class libraries also contain the ASP.NET classes and the Windows Forms classes which are used to support the development of Web-based applications and desktop-based applications.

Below in Figure 1 is an overview architecture of Visual Studio .NET.

![Overview Architecture of Visual Studio .NET](image)

**Figure 1: Overview Architecture of Visual Studio .NET [1]**

The programming language that we will be using for this lab is C# (pronounced C Sharp). Visual C# is an object-oriented programming language developed by Microsoft as part of the .NET initiative. Visual C# combines the best features of C++, Java, and Visual Basic. In

---

3
this section we will introduce you to the features of Visual C#, starting with the basic terms followed by a brief review on variables and data types.

I. BASIC TERMS

• Variable: is a memory location that is use to store data. A variable is referred to by name which points to a memory address that contains the data.

• Data Type: refers to kinds of data such as numbers or text. Common data types include integer, string, and double.

• Constant: is a variable that stores a value that always remains the same. (e.g. PI = 3.14159)

• Keyword: is a term with special meaning for the system. They are often denoted by a blue font color. (e.g. int, string, and etc.)

• Operator: is a symbol used to perform arithmetic or relational operations on constants and variables. (e.g. +, -, <=, and >=)

• Statement: is a combination of variables, constants, operators, expressions and keywords.

II. VARIABLES AND DATA TYPE

In this section you will learn about the different data types supported by C#. In addition, you will learn the fundamentals of creating variables.

Variables are useful when you need to temporarily store data in your application. For example, your application requires your user to input 10 numbers and perform addition to them. Obviously, you would need 10 variables to store the 10 numbers from the users and 1 additional variable to store the result after adding the 10 numbers.

Again, a variable is a memory location that stores data temporarily. Before using a variable, you must declare the variable first. An example includes:

    int integer_var1 = 0;
    int integer_var2;
There are many different types of data that can be stored into a variable. However, only certain operations can be performed on the data of a particular type. Visual C# provides you with many different data types to handle numeric, text, byte, Boolean, and date/time data. In addition, C# also provides an Object data type, which can store any type of data. Below we’ll explain the different data types that will handle numeric, text, and Boolean values.

**Numeric data types:** C# provides Integer, Short, and Long data types to store numerals (e.g., 1, -1, and 50). To store numbers with floating points you can use Double or Decimal.

**Text data types:** C# provides Char and String data types to store text data. Char data types will only store characters, whereas String will store the entire line of text.

### Structures concepts: While Loop, For Loop, Case

Recall in LabVIEW, there are 3 types of structures: While Loop, For Loop, and Case. These structures were used to conditionally repeat or execute a set of instructions. Similarly, these structures can also be found in C#.

### WHILE LOOP

```csharp
while (condition) {
    Statement(s);
}
```

**Figure 2:** The while loop structure represented by LabVIEW (left) and C# (right).

In LabVIEW, the while loop executes a sub-diagram until a condition is met. Same applies to C#. We use the while loop statement to execute a set of statements until a given condition becomes false. At that point, we exit the block and continue with our program.
Below is example where we use the while loop:

```csharp
static void Main(string[] args)
{
    int i = 0;

    while (i < 10)  // The while loop will continue to
    // execute these statements until
    // i is greater or equal to 10
    {
        // Output the value of i onto the screen
        System.Console.WriteLine("The value of i is " + i);
        i++;        // Increment i by 1
    }
}
```

Figure 3: Sample code describing the use of the while loop.

**FOR LOOP**

```csharp
for (initialization_expr; test_expr; change_expr)
{
    Statement(s);
}
```

Figure 4: The for loop structure represented by LabVIEW (left) and C# (right).

In C#, the For Loop executes a set of statements until a given condition becomes false. For example, we could use the For statement to repeat a set of statements. The syntax of the For statement is given in Figure 4 (Right).

- `initialization_expr` refers to the statements that initialize the counters used in a loop. This statement is executed only once.

- `test_expr` is the expression that is evaluated for repeating the enclosed statements.
change_expr is used to increment or decrement the value of the counters used in the loop.

Below is an example of how we use the For loop:

```csharp
static void Main(string[] args)
{
    int counter; // Initialize variable

    for (counter = 0; counter < 10; counter = counter + 1)
    {
        System.Console.WriteLine("The value of counter is " +
        counter); // Output the value of counter to the screen.
    }
}
```

Figure 5: Sample code describing the use of the for loop.

(SWITCH) CASE STRUCTURE

```
switch (expression)
{
    case ConstantExpression1
        Statement(s);
        break;
    default:
        Statement(s);
        break;
}
```

Figure 6: The (switch) case structure represented by LabVIEW (left) and C# (right).

In C# we refer to the case structure as the "Switch" statement. The Switch statement executes a set of statements depending on the result of the expression. The result of the expression is tested across a number of constants provided using the case statement.

The syntax for the switch statement is shown on figure 6 (right), the expression is evaluated and the result is checked across ConstantExpression. If the two values match, the statements between the keyword "case" and "break" will be executed. However, if none
of the ConstantExpressions match, the statements following the keyword “default” are executed.

For additional information and help on developing applications using Visual C#, please visit www.msdn.com.

EXPERIMENT

Now that you have an idea about Visual Studio .NET and Visual C#, you are ready to begin the experiment.

Creating a New Project

- Log on to the computer
- Launch Visual Studio .NET 2005 as shown in figure 7.
The next step is to create a new project. Go to File → New Project (Ctrl+Shift+N) as shown in figure 8.
In a New Project window, select “Visual C#” in the Project types and under Templates, highlight “Console Application.” Then press OK to continue.
As a result, a new window will appear which is the basic programming environment for Visual Studio .NET.

![Figure 10: Visual Studio Programming Environment](image)

- **Solution Explorer**: offers you an organized view of your projects and their files as well as ready access to the commands that pertain to them.
- **Code Editor**: allows you to input your code into the program. All Visual C# programs have the .cs extension.
Create a new Visual C# Project named "L6_Sample1_TeamName". Enter the code found in figure 10 into the code editor and when you are done, please run the program ("Ctrl + F5").

```csharp
static void Main(string[] args)
{
    int counter;
    string strName;

    System.Console.WriteLine("Hello, what is your name? ");
    strName = System.Console.ReadLine();

    for (counter = 0; counter < strName.Length; counter = counter + 1)
    {
        System.Console.WriteLine("Your name is " + strName.Trim() + "! ");
    }

    System.Console.WriteLine("Hi " + strName.Trim() + ", it is pleasure to meet you!");
}
```

**Figure 10: Sample Code #1**

What is the program trying to do? For each statement in the program, insert a comment stating what this line does.

**Example:**

```csharp
int counter; \ \ Initialize an integer variable call counter.
```

When you are done with this part, please save this project and notify the lab instructor for check-off.
Create a new Visual C# Project named "L6_Sample2_TeamName". Enter the code found in figure 11 into the code editor and when you are done, please run the program ("Ctrl + F5").

```csharp
static void Main(string[] args)
{
    int value;
    string strInformation;

    while (true)
    {
        System.Console.WriteLine("Please enter a number between 1 to 10: ");
        strInformation = System.Console.ReadLine();
        value = int.Parse(strInformation); // Convert a type string to an integer.

        if (value > 5)
        {
            System.Console.WriteLine("The number you entered is too high.");
        }
        else if (value < 5)
        {
            System.Console.WriteLine("The number you entered is too low.");
        }
        else
        {
            System.Console.WriteLine("The number you entered is just right!! Goodbye.");
            break; // Exit out of a loop, commonly use for infinite loop.
        }
    }
}
```

Figure 11: Sample Code #2

What is the program trying to do? For each statement in the program, insert a comment stating what this line does.

Example:

```csharp
int counter; \ \ Initialize an integer variable call counter.
```

When you are done with this part, please save this project and notify the lab instructor for check-off.
Application #1: Number Guessing Game

For this section, you are asked to write a game. The objective of this game is to guess the correct number which the program has randomly generated. The game must take in a number between 1 and 100. If the user enters a number that is too low or too high, the system will inform the user of the error and ask the user to re-enter a number. If the number that the user has entered is higher than the correct number, the system must notify the user, the number is too high. Similarly, if the number is lower then the correct number, the system must notify the user, the number is too low.

The system will keep track of the number of tries and display the result when the user has found the number. Label this project "L6_App1_TeamName."
The code below is used to generate a random number, please incorporated into your game.

```csharp
Random randomizer = new Random();
int correctNumber = randomizer.Next(100);
```

Figure 12: Number generator.

When you are done, please notify your lab instructor for sign-off.

What seems to be the hardest part of this lab for you? Explain.

References:

[1] ".NET Framework"  

Lab 6: Introduction to Visual Studio .NET

<table>
<thead>
<tr>
<th>Group Number &amp; Name:</th>
<th>Station ID:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Session Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Students’ Names and Check-in/Check-out Time:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Name</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>1.</td>
</tr>
<tr>
<td>2.</td>
</tr>
<tr>
<td>3.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Equipment Verification:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK IN: Speedy-33 Webcam Headphones</td>
</tr>
<tr>
<td>CHECK OUT: Speedy-33 Webcam Headphones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Items to be checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Sample Code #1 TA Signature:</td>
</tr>
<tr>
<td>Run Sample Code #2 TA Signature:</td>
</tr>
<tr>
<td>Create Number Guessing Game TA Signature:</td>
</tr>
<tr>
<td>All above items and lab report submitted to Digital Drop Box as a single archived file TA Signature:</td>
</tr>
</tbody>
</table>

Comments (TA use)