عنوان مضمون

Visual Programming-I

توسط : صفری

خزان1397

Visual Programming-I

- **Course Code:** 78024
- **Lecturer:** Baba Ali Safari
- **Course Credit:** 3
- **Class Hours:** 48
- **Theory:** 2
- **Practice:** 1

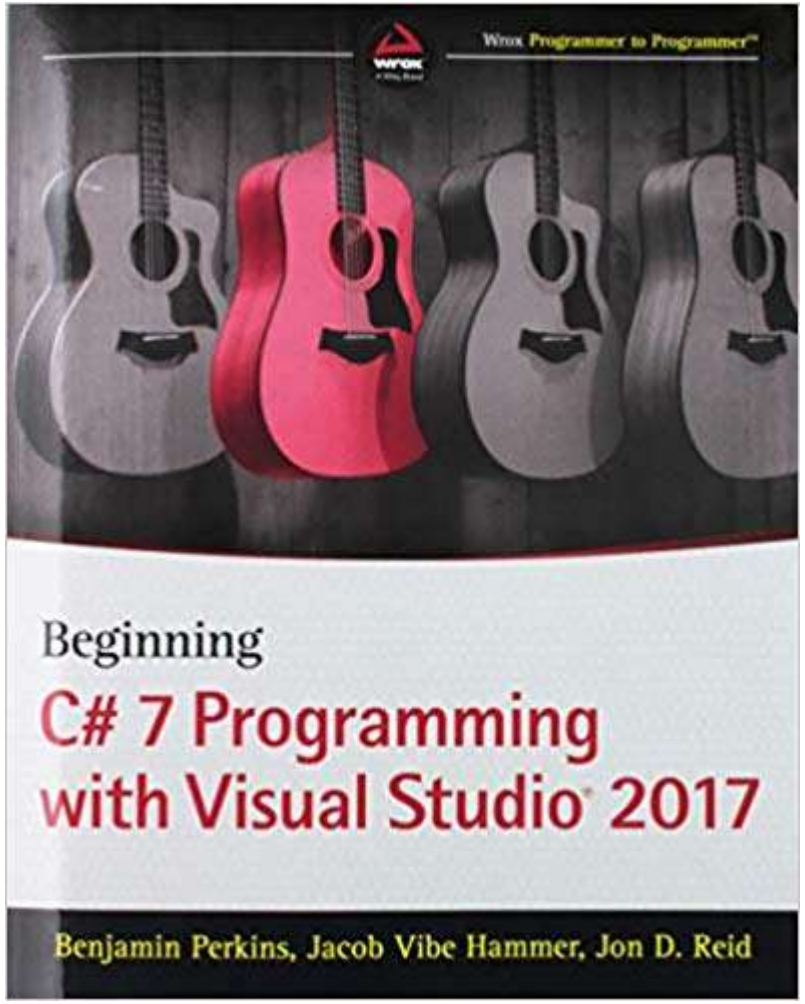
Textbooks:

BEGINNING C# Programming with Visual Studio 2017

Benjamin Perkins

Jacob Vibe Hammer

Jon D. Reid



Topic to be Covered:

1. Introducing C# program and writing C# program using console and Visual studio.net (Week 1)
2. C# data type and type conversion (Week 2)
3. C# variable, constant and literals (Week 3)
4. C# operators (Week 4)
5. C# decision making (Week 5)
6. C# methods and Arrays (Week 6)
7. C# Strings and regular expressions (Week 7)
8. Mid Term (Week 8)
9. C# Object oriented features (Week 9)
10. C# Exception handling (Week 10)
11. C# File I/O (Week 11)
12. Windows form Application (Week 12)
13. Class library (Week 13)
14. collections (Week 14)
15. generics(Week 15)
16. Overview (Week 16)

Grading Policy:

- Presentation and Quizzes: 20%
- Midterm Exam: 20%
- Project: 20%
- Final Exam: 40%

Introducing C# program and writing C# program using console and Visual studio.net

WHAT IS?

- **WHAT IS THE .NET FRAMEWORK?**
- **WHAT IS C#?**
- **WHAT IS Visual Studio?**

WHAT IS THE .NET FRAMEWORK?

- The .NET Framework is a revolutionary platform created by Microsoft for developing applications.
- the .NET Framework enables the creation of :
  - desktop applications
  - Windows Store applications,
  - web applications,
  - web services, and pretty much anything else you can think of.
- Although the Microsoft release of the .NET Framework runs on the Windows and Windows Phone operating systems, it is possible to find alternative versions that will work on other systems.

  One example of this is Mono, an open-source version of the .NET Framework (including a C# compiler) that runs on several operating systems, including various flavors of Linux and Mac OS.

  There are also variants of Mono that run on iPhone (MonoTouch) and Android
- The .NET Framework has been designed so that it can be used from any language, including C# , as well as C++, Visual Basic, JScript, and even older languages such as COBOL.

What’s in the .NET Framework?

- The .NET Framework consists primarily of a gigantic library of code that you use from your client languages (such as C#) using object-oriented programming (OOP) techniques.
- This library is categorized into different modules — you use portions of it depending on the results you want to achieve.
  - For example, one module contains the building blocks for Windows applications, another for network programming, and another for web development.
  - Some modules are divided into more specific submodules, such as a module for building web services within the module for web development.

Writing Applications Using the .NET Framework

- Writing an application using the .NET Framework means writing code (using any of the languages that support the Framework) using the .NET code library.

- In this book you use **Visual Studio(VS)** for your development. VS is a powerful**, integrated development environment(IDE)** that supports C# (C++, Visual Basic, and some others).

WHAT IS C#?

- C#:
  - is one of the languages you can use to create applications that will run in the .NET .
  - It is an evolution of the C and C++ languages and has been created by Microsoft specifically to work with the .NET platform.
- C# is a powerful language, and there is little you might want to do in C++ that you can't do in C#.
  - such as **directly accessing** and **manipulating system memory**, can be carried out only by using code marked as ***unsafe***.
  - This is a consequence of C# being a ***typesafe*** language (unlike C++).
  - . . .

Applications You Can Write with C#

- **Desktop applications** — Applications, such as Microsoft Office, that have a familiar Windows look and feel about them. This is made simple by using the Windows Presentation Foundation (WPF) module of the .NET Framework, which is a library of *controls* (such as buttons, toolbars, menus, and so on) that you can use to build a Windows user interface (UI).
- **Windows Store applications** — Windows 8 has introduced a new type of application, known as a Windows Store application. This type of application is designed primarily for touch devices, and it is usually run full-screen, with a minimum of clutter, and an emphasis on simplicity. You can create these applications in several ways, including using WPF.
- **Web applications** — Web pages such as those that might be viewed through any web browser. The .NET Framework includes a powerful system for generating web content dynamically, enabling personalization, security, and much more. This system is called ASP.NET (Active Server Pages .NET), and you can use C# to create ASP.NET applications using Web Forms. You can also write applications that run inside the browser with Silverlight.
- **WCF services** — A way to create versatile distributed applications. Using WCF you can exchange virtually any data over local networks or the Internet, using the same simple syntax regardless of the language used to create a service or the system on which it resides.

Solutions

- A solution, in VS terms, is more than just an application.

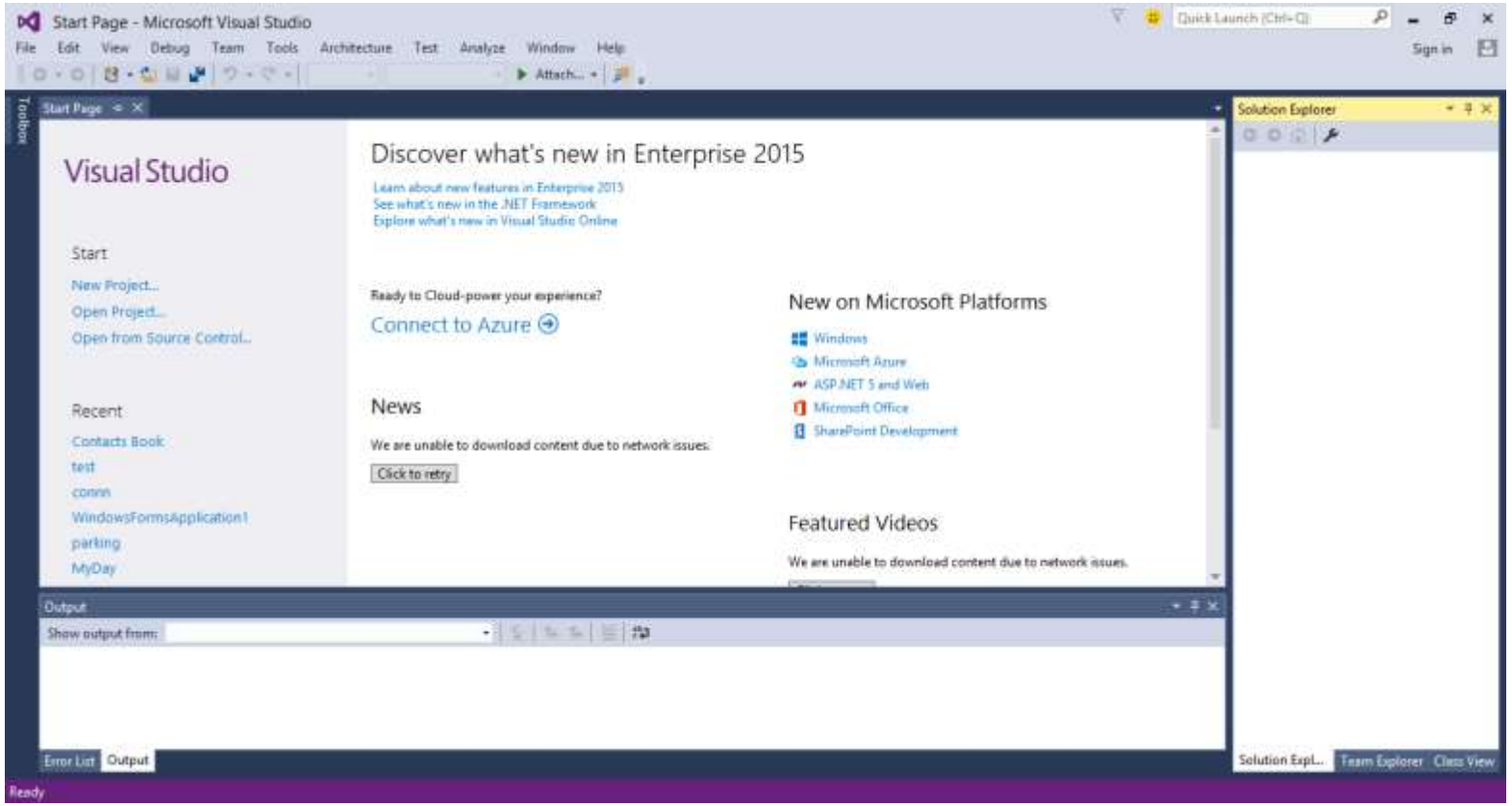
  Solutions contain *projects,* which might be WPF projects, Web Application projects, and so on.

- Because solutions can contain multiple projects, you can group together related code in one place, even if it will eventually compile to multiple assemblies in various places on your hard disk.
- This is very useful because :
  - it enables you to work on shared code at the same time as applications that use this code.
  - Debugging code is a lot easier when only one development environment is used, because you can step through instructions in multiple code modules.

Introducing C# program and writing C# program using console and Visual studio.net

THE VISUAL STUDIO DEVELOPMENT ENVIRONMENT

- The VS environment layout is completely customizable, but the default is fine here.



main features that you will use the most:

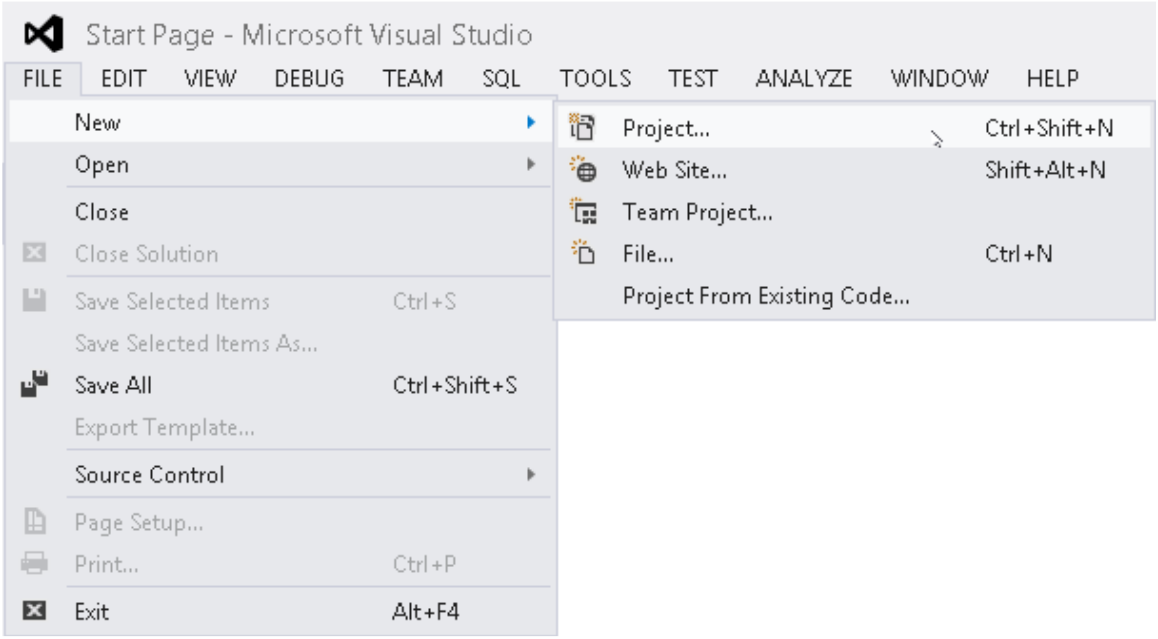
- **The Toolbox window** pops up when you click its tab. It provides access to, among other things, the user interface building blocks for desktop applications
- **The Solution Explorer** window displays information about the currently loaded *solution.*

  The Solution Explorer window displays various views of the projects in a solution, such as what files they contain and what is contained in those files.
- **The Team Explorer** window displays information about the current Team Foundation Server or Team Foundation Service connection. This allows you access to source control, bug tracking, build automation, and other functionality. However, this is an advanced subject and is not covered in this book.
- **The Properties window :**Just below the Solution Explorer window you can display a Properties window, not shown in Figure because it appears only when you are working on a project .For example, you can use this window to change the appearance of a button in a desktop application.
- **The Error List window:** which you can display using View ⇨ Error List. It shows errors, warnings, and other project-related information. The window updates continuously, although some information appears only when a project is compiled.

CONSOLE APPLICATIONS

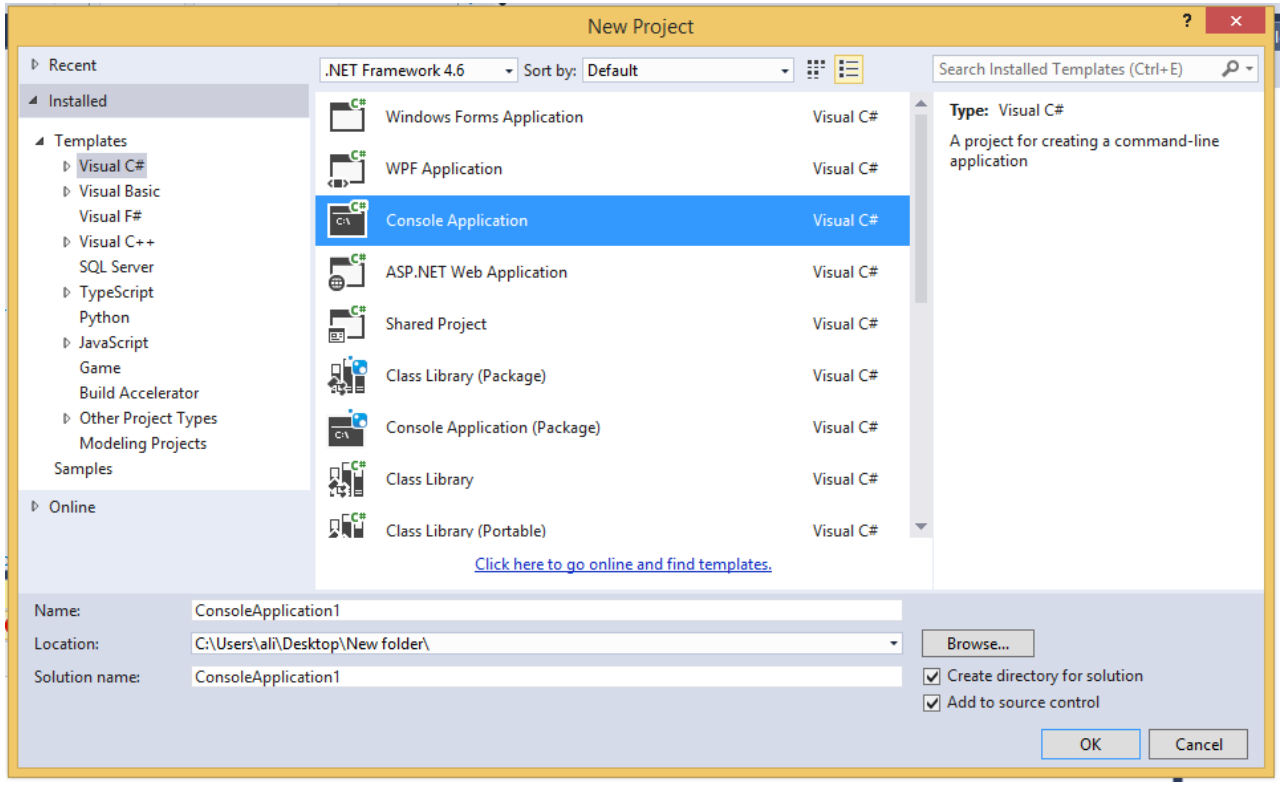
- Creating a Simple Console Application: ConsoleApplication1\Program.cs

1- Create a new console application project by selecting File ➪ New ➪ Project



CONSOLE APPLICATIONS

2. Ensure that the Visual C# node is selected in the left pane of the window that appears
3. choose the Console Application project type in the middle pane
4. Change the Location text box to C:\BegVCSharp\Chapter02
5. Leave the default text in the Name text box (ConsoleApplication1)
6. the other settings as they are



CONSOLE APPLICATIONS

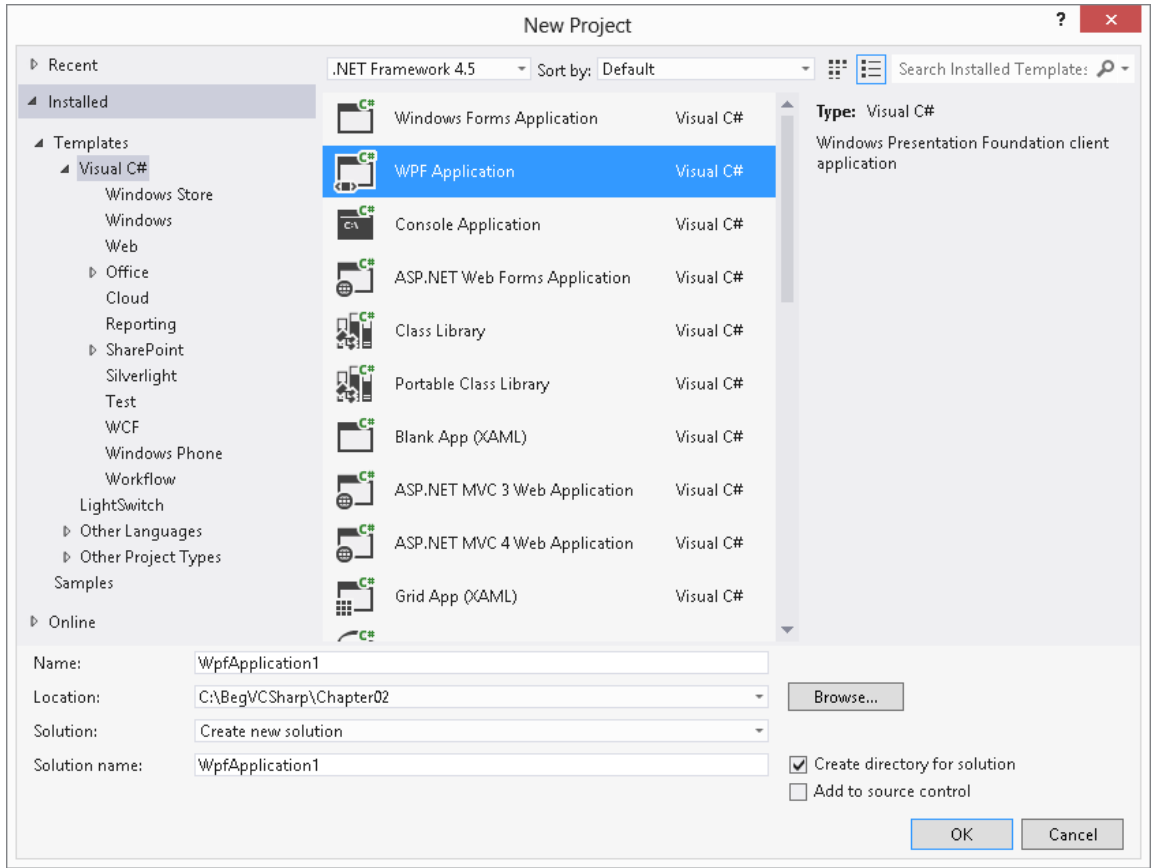
Once the project is initialized, add the following lines of code to the fi le displayed in the main window:

```
namespace ConsoleApplication1
{       class Program
        {       static void Main(string[] args)
                {       // Output text to the screen.
                        Console.WriteLine("The first app in Beginning Visual C# 2012!");
                        Console.ReadKey();
                }
        }
}
```

Select the Debug ➪ Start Debugging menu item. After a few moments you should see the window shown in Figure

DESKTOP APPLICATIONS

- Create a new project of type WPF Application in the same location as before , with the default name WpfApplication1.



DESKTOP APPLICATIONS

- You should see a new tab that's split into two panes.
  1- The top pane shows an empty window called MainWindow
  2- the bottom pane shows some text.
- Click the Toolbox tab on the top left of the screen, then double-click the Button entry in the Common WPF Controls section to add a button to the window.
- Double-click the button that has been added to the window.
- The C# code in MainWindow.xaml.cs should now be displayed. Modify it as follows

```
private void Button_Click_1(object sender, EventArgs e)
{       MessageBox.Show("The first desktop app in the book!");
}
```

- Run the application.