

Lab 01

Building Applications Using C#

Variables and Data Types

Mục tiêu

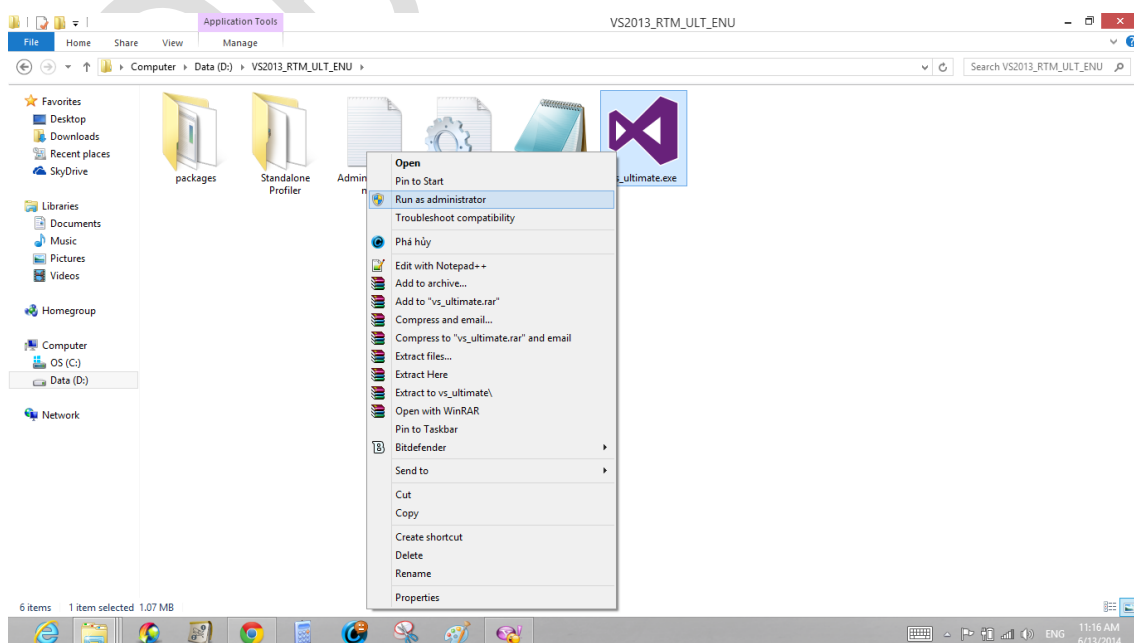
- Cài đặt Visual Studio 2013
- Tạo project, biên dịch, chạy và xem kết quả
- Khai báo và sử dụng biến
- Nhập xuất trong ứng dụng Console
- Chuyển đổi và định dạng dữ liệu

Phần I Bài tập step by step

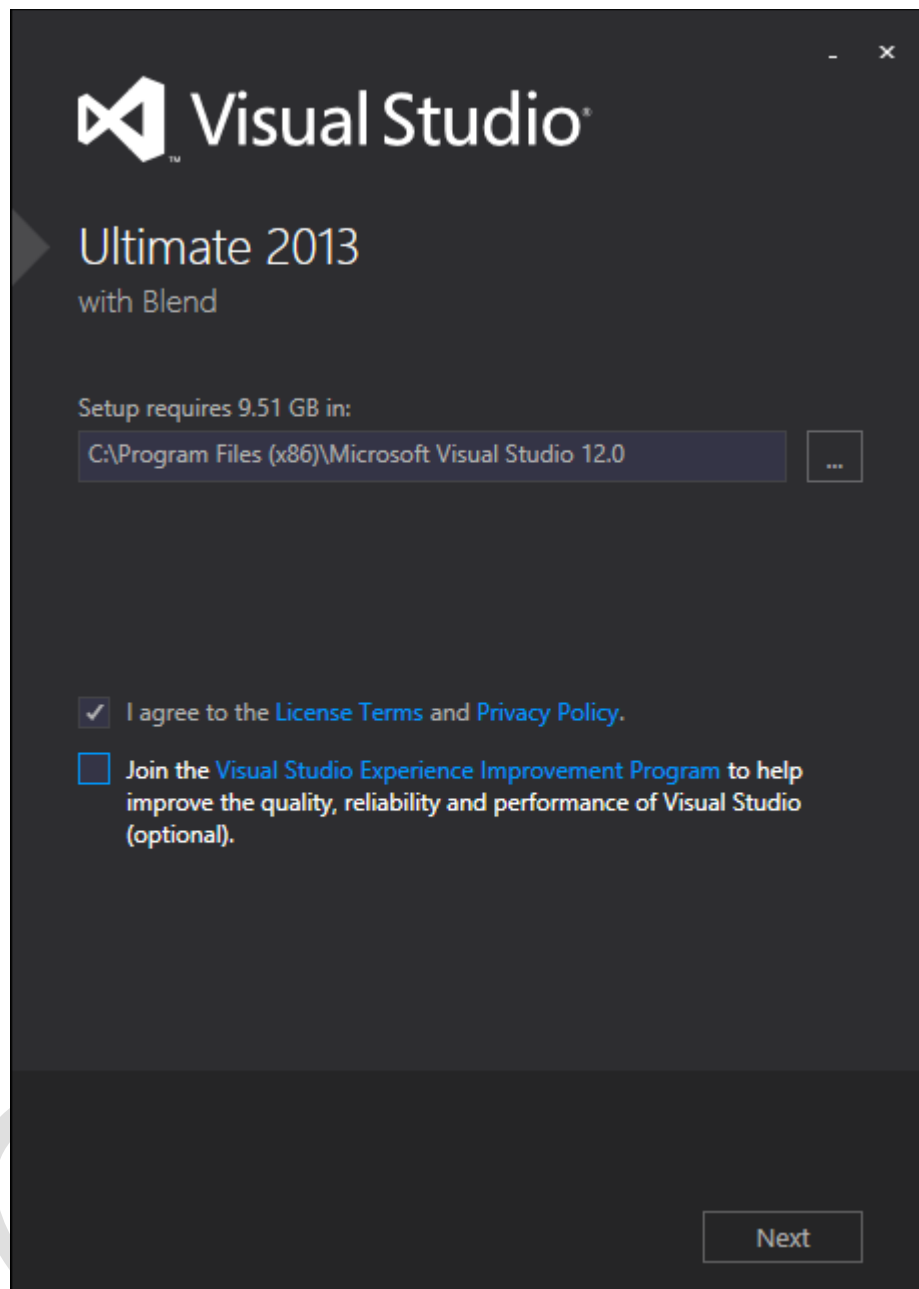
Bài 1.1

Hướng dẫn cài đặt Visual Studio 2013 (Bộ cài copy của thầy hoặc phòng kỹ thuật)

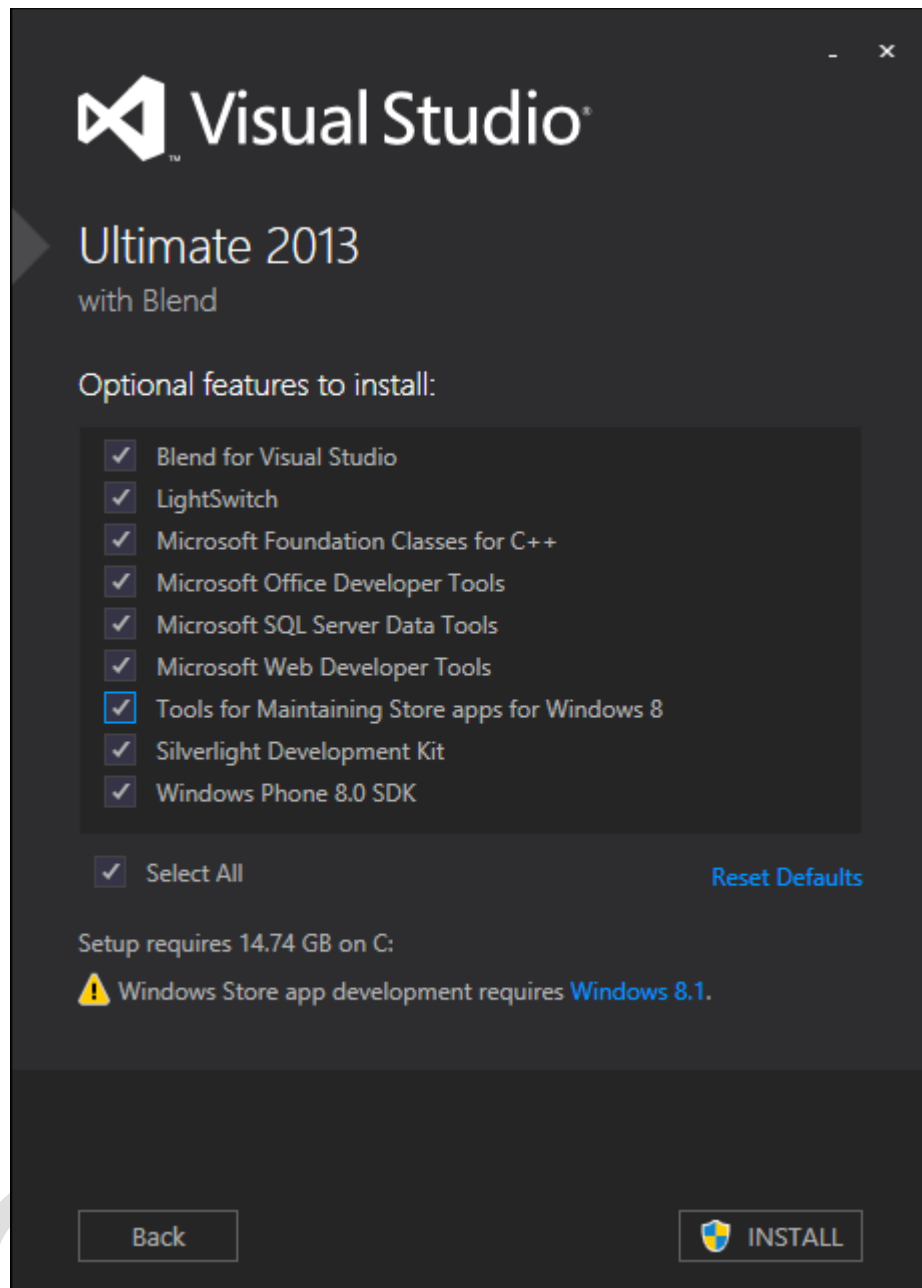
Bước 1: Giải nén tệp .ISO sau đó mở thư mục chạy tệp vs_ultimate.exe với “Run as administrator”



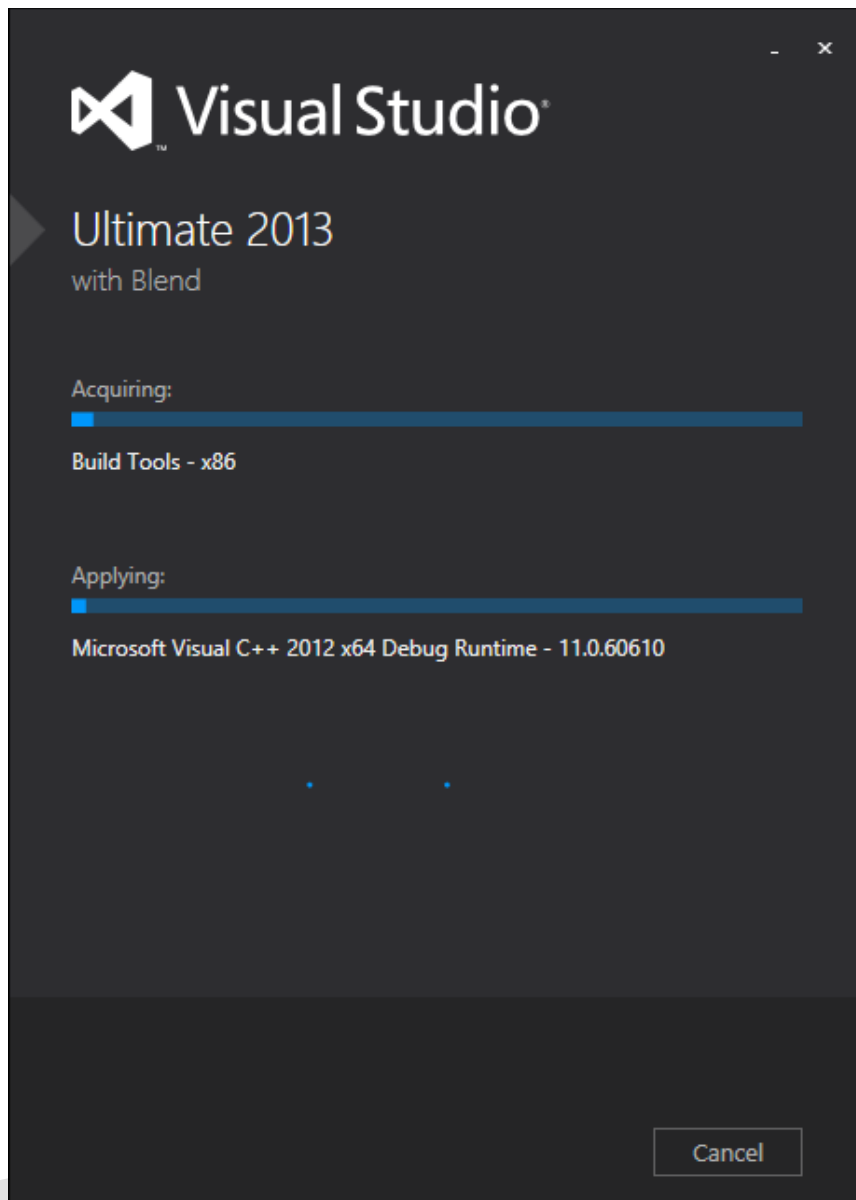
Bước 2: Chọn “I gree...” -> Next



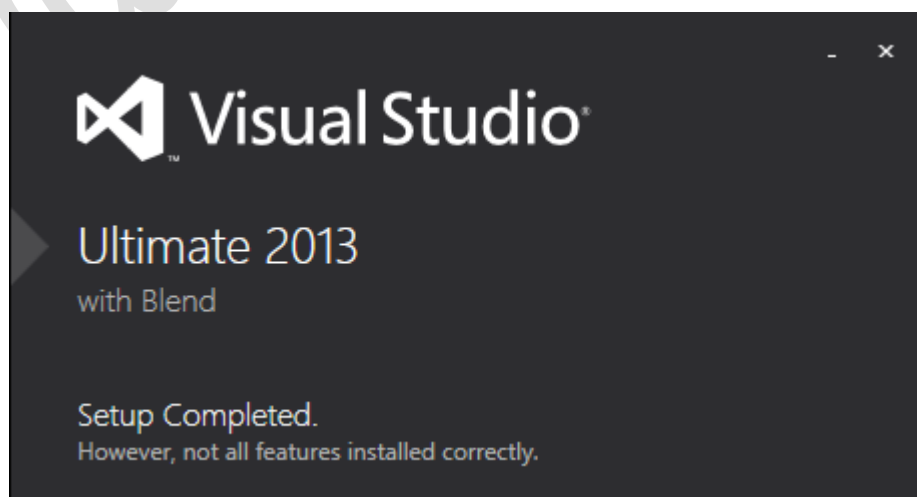
Bước 3: Chọn All để phục vụ cho lập trình Windows Apps Store và Windows Phone 8 ở SEM III nhé -> kích “Install”.



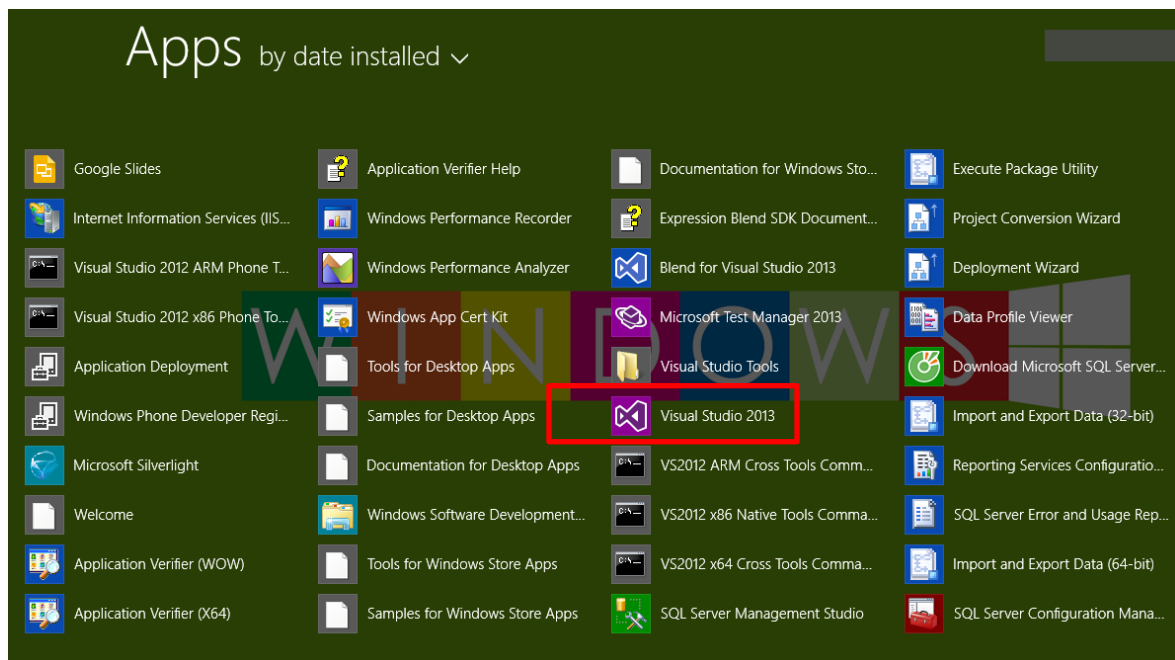
Quá trình cài đặt diễn ra...



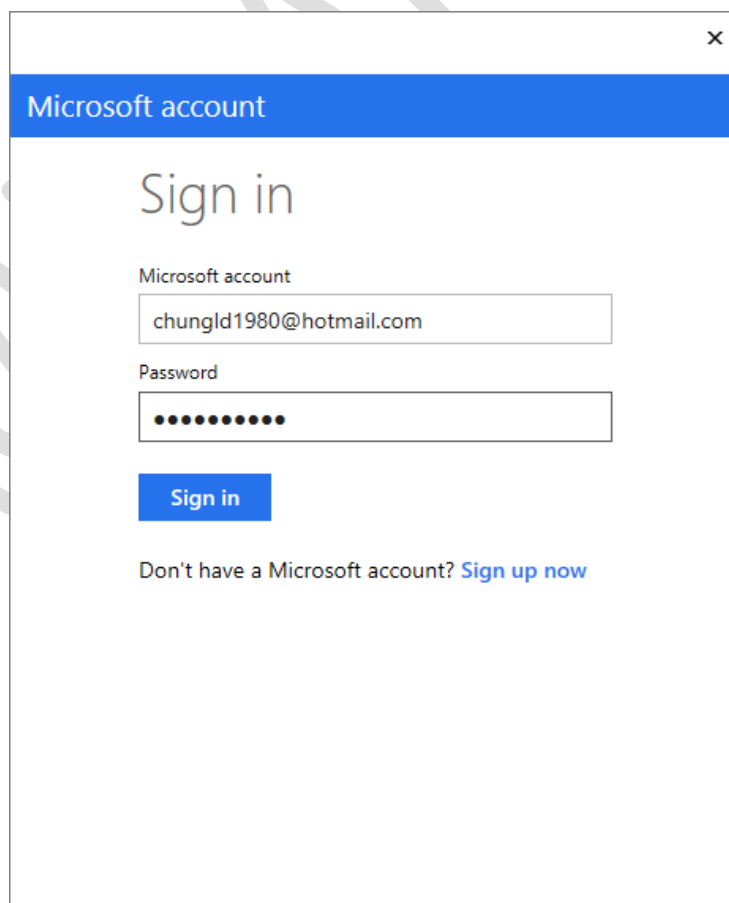
Thông báo khi hoàn thành



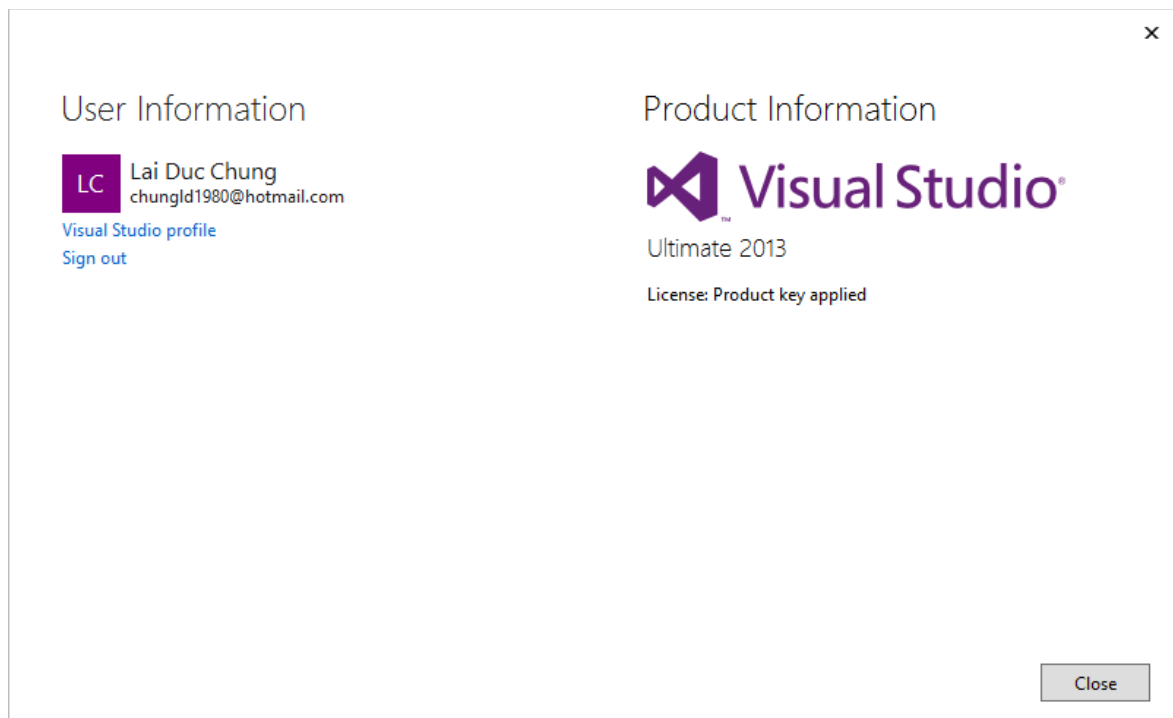
Trong danh sách app của windows 8 xuất hiện



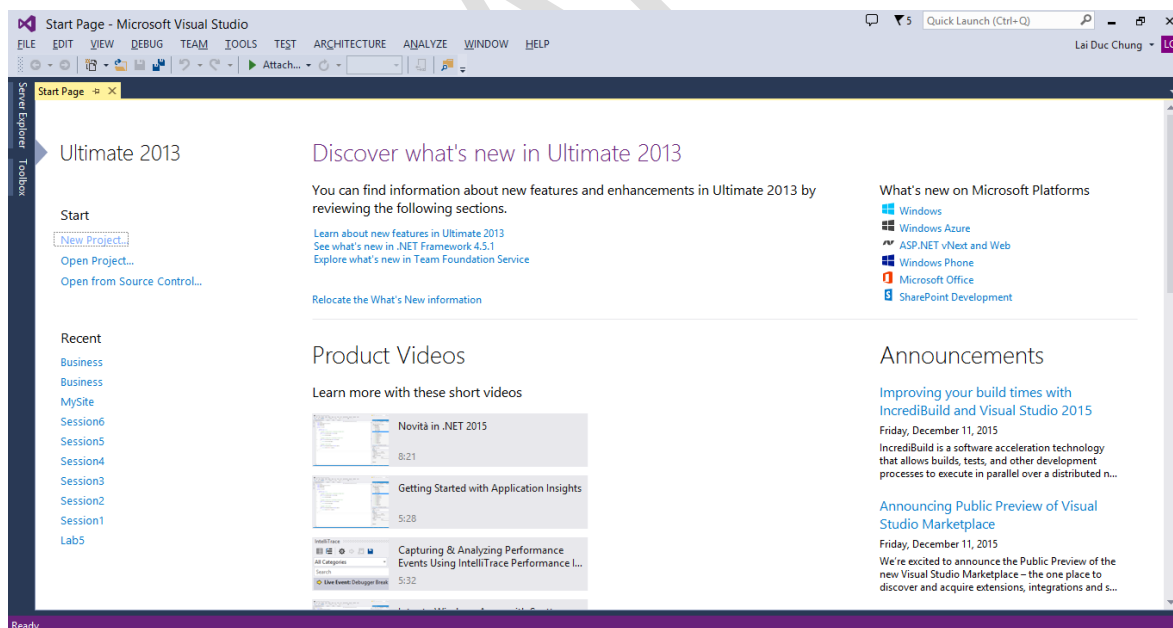
Bước 4: Khởi động Visual Studio và Sign in để sử dụng (tài khoản của google, hoặc microsoft đều được nhé)



Sign in thành công (đây là bản visual đã có key sẵn nhé)



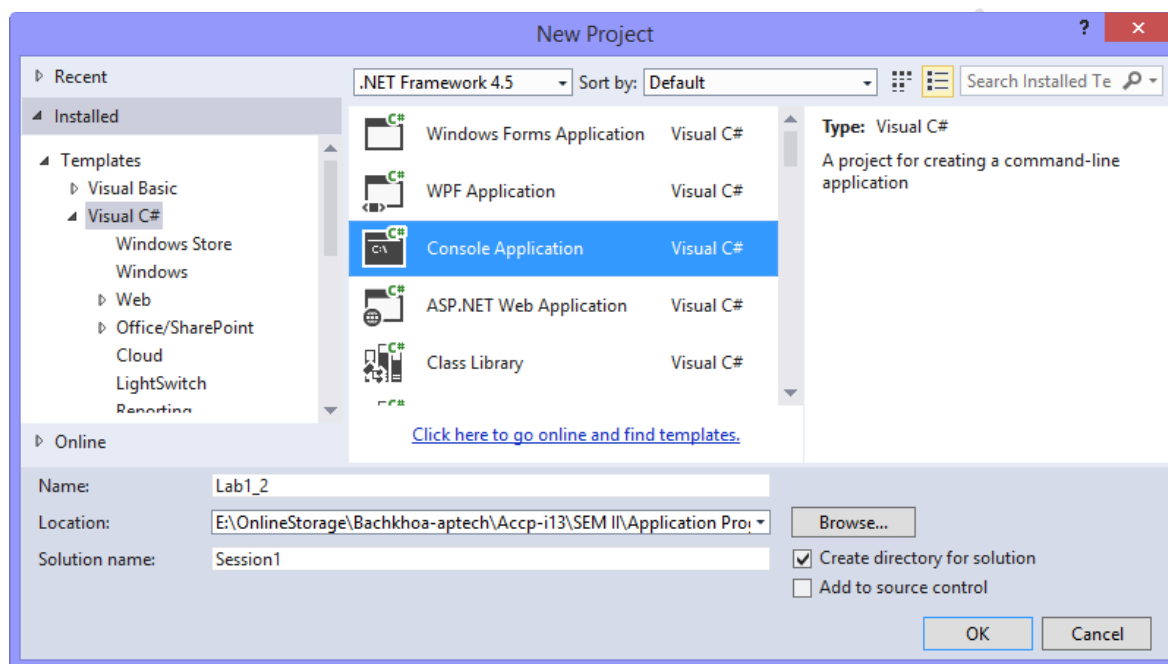
Màn hình visual Studio 2013 khởi động lên



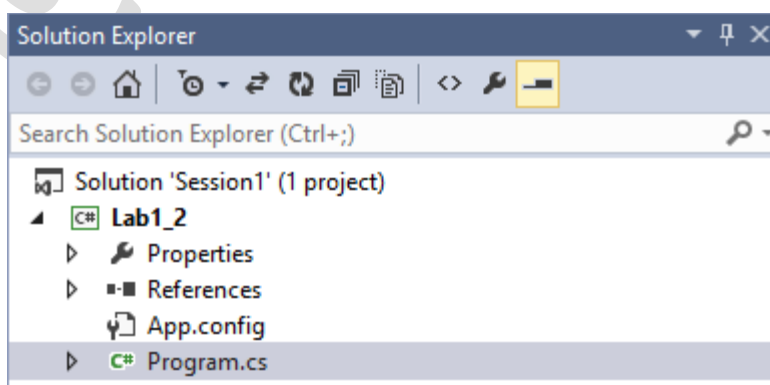
Bài 1.2

Tạo project C# thực hiện việc in ra màn hình thông tin giới thiệu về bản thân, biên dịch, chạy và kiểm tra kết quả

Bước 1: Mở Visual Studio 2013, vào menu File -> New -> Project -> chọn loại project “Console Application”, nhập tên project, tên solution -> OK.



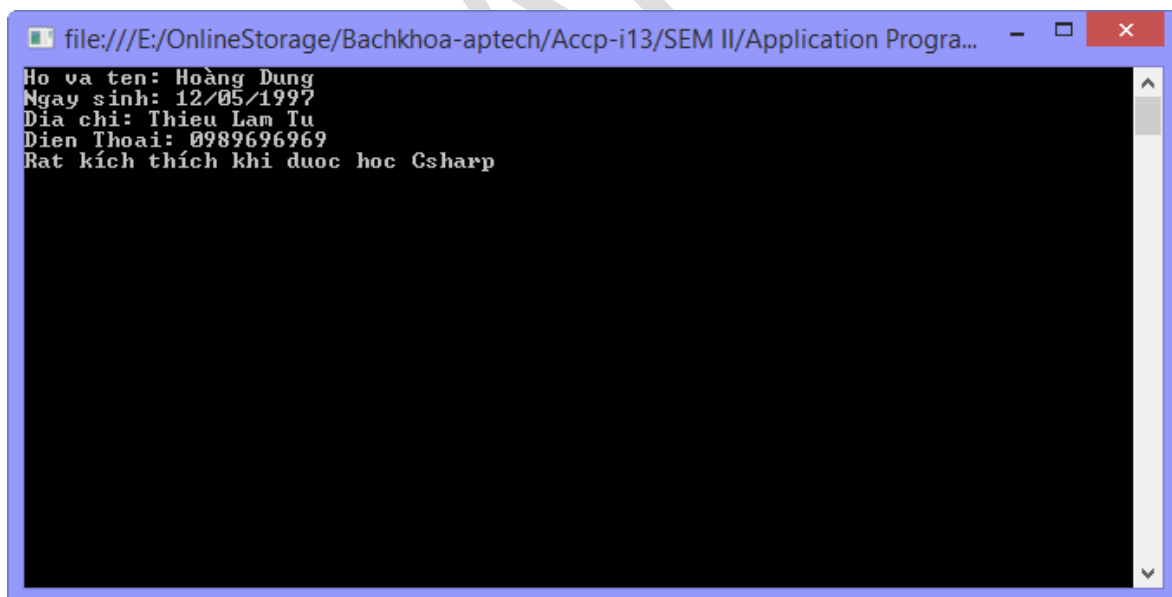
Visual sẽ tạo ra một Solution có tên “Session1”, trong đó có một project “Lab1_2”. Mỗi project Console Application luôn mặc định tạo ra một lớp Program với phương thức Main (đây là phương thức mặc định sẽ chạy đầu tiên trong ứng dụng) đặt trong namespace là tên solution(namespace chúng ta sẽ học sau).



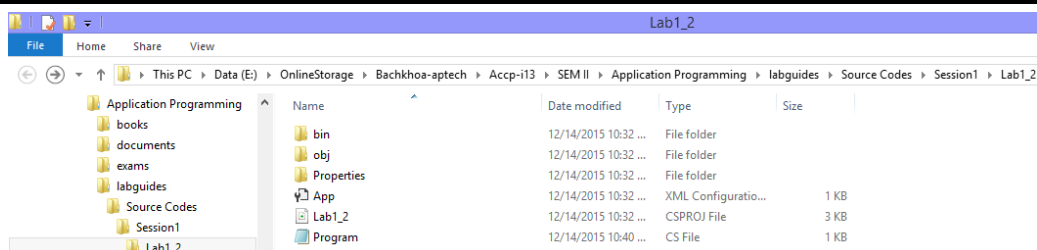
Bước 3: Chú thích vào viết code vào trong Main theo gợi ý sau:

```
/// <summary>
/// Chương trình c# đầu tiên
/// In ra thông tin giới thiệu về bản thân
/// </summary>
/// <param name="args"></param>
static void Main(string[] args)
{
    //Lệnh in ra màn hình 1 dòng văn bản, sau đó xuống dòng
    Console.WriteLine("Ho va ten: Hoàng Dung");
    Console.WriteLine("Ngày sinh: 12/05/1997");
    Console.WriteLine("Địa chỉ: Thieu Lam Tu");
    Console.WriteLine("Điện Thoại: 0989696969");
    Console.WriteLine("Rất thích khi được học Csharp");
    //lệnh dừng màn hình và chờ nhận 1 phím
    Console.Read();
}
```

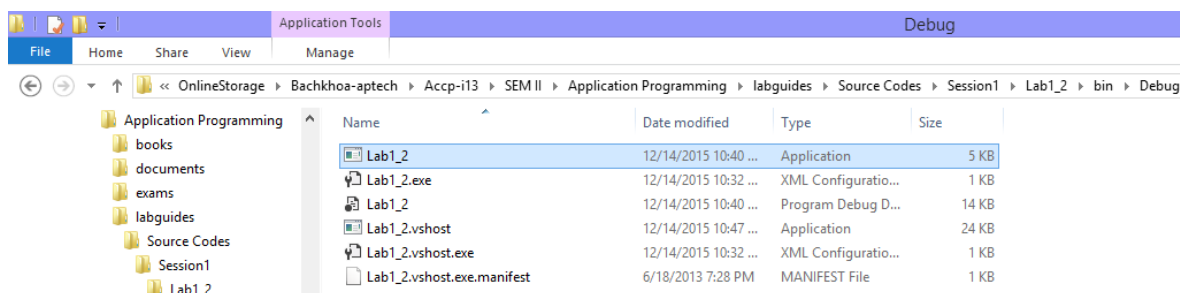
Bước 4: Nhấn F5 để build và chạy chương trình



Mặc định Visual thiết lập chạy ở mode “Debug” bạn cũng thể chọn chế độ khác, sau khi chạy xong Visual sẽ build thành tệp ứng dụng tại thư mục Debug. Để kiểm tra kết quả build các bạn kích chuột phải vào Project chọn “Open folder in File Explorer” -> mở thư mục bin.



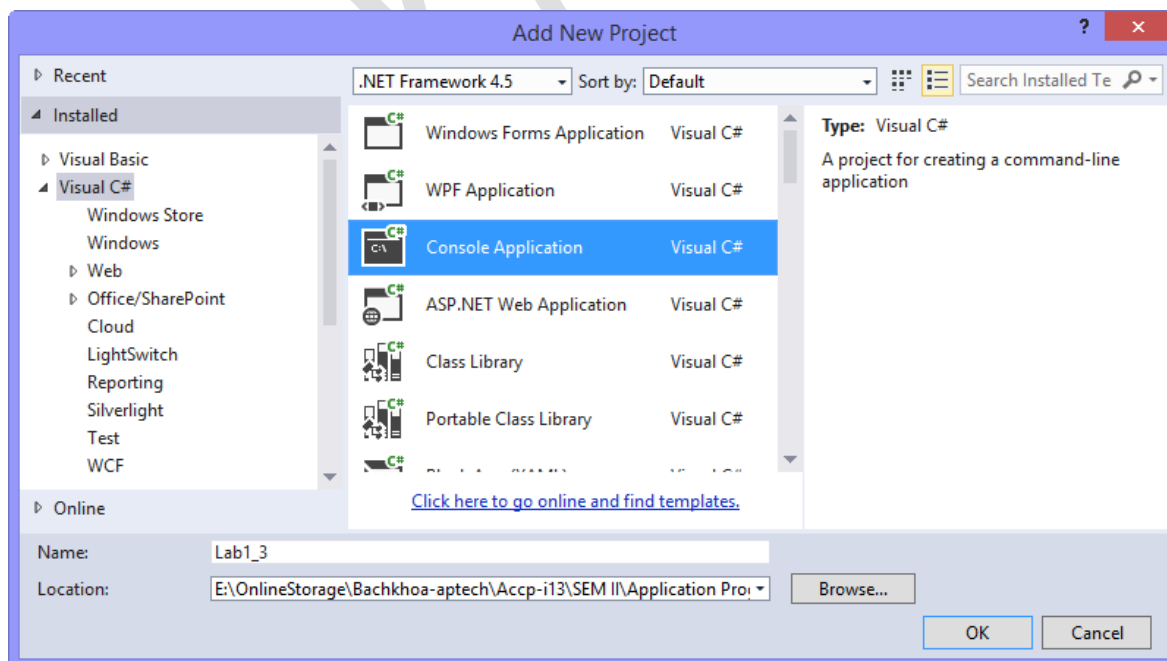
Mở thư mục bin/debug



Bài 1.3

Tạo ứng dụng # Console Application minh họa việc khai báo biến, hằng số và sử dụng chúng.

Bước 1: Kích chuột phải vào Solution “Session1” chọn Add -> New Project -> nhập tên.

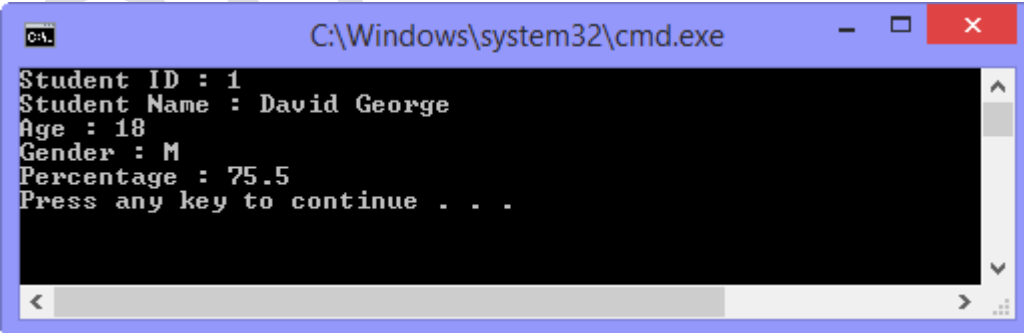


Bước 2: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
/// <summary>
/// Chương trình minh họa việc sử dụng biến, hằng số và kiểu dữ liệu
/// </summary>
/// <param name="args"></param>
static void Main(string[] args)
{
    //Khai báo các biến
    int id = 1;
    string name = "David George";
    byte age = 18;
    char gender = 'M';
    //Khai báo hằng số
    const float percent = 75.50F;

    //Hiển thị giá trị các biến và hằng số
    Console.WriteLine("Student ID : {0}", id);
    Console.WriteLine("Student Name : {0}", name);
    Console.WriteLine("Age : " + age);
    Console.WriteLine("Gender : " + gender);
    Console.WriteLine("Percentage : {0}", percent);
}
```

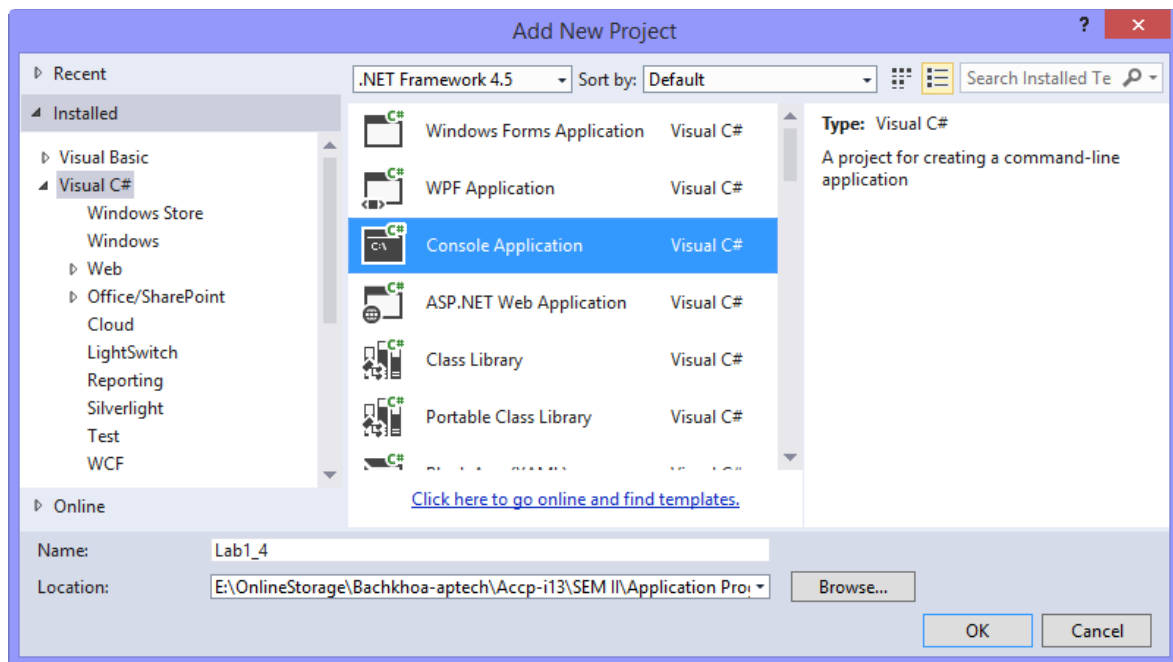
Bước 3: Nhấn Ctrl+F5 để chạy và xem kết quả



Bài 1.4

Nhập và thông tin sinh viên bao gồm (Id, Name, Birthday, Mark1, Mark2, Mark3), in ra thông tin sinh viên bao gồm cả điểm trung bình của 3 môn.

Bước 1: Kích chuột phải vào Solution "Session1" chọn Add -> New Project -> nhập tên.



Bước 2: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
/// <summary>
/// Chương trình minh họa việc nhập xuất trong ứng dụng console
/// Nhập vào mã sinh viên, tên sinh viên, ngày sinh và điểm 3 môn
/// In ra thông tin sinh viên và điểm trung bình
/// </summary>
/// <param name="args"></param>
static void Main(string[] args)
{
    //khai báo các biến
    string id, name;
    double mark1, mark2, mark3, average;
    DateTime birthday;
    //nhập dữ liệu từ bàn phím
    Console.Write("Nhập mã so:");
    id = Console.ReadLine();
    Console.Write("Nhập ten:");
    name = Console.ReadLine();
    Console.Write("Nhập ngày sinh:");
    birthday = Convert.ToDateTime(Console.ReadLine());
}
```

```
Console.Write("Nhap diem mon 1:");
mark1 = Convert.ToDouble(Console.ReadLine());
Console.Write("Nhap diem mon 2:");
mark2 = Convert.ToDouble(Console.ReadLine());
Console.Write("Nhap diem mon 3:");
mark3 = Convert.ToDouble(Console.ReadLine());
//tính điểm trung bình
average = (mark1 + mark2 + mark3) / 3;
//In thông tin ra màn hình
Console.WriteLine("\n Thông tin sinh viên");
Console.WriteLine("Ma so:{0}", id);
Console.WriteLine("Ho va ten:{0}", name);
Console.WriteLine("Ngày sinh:{0:dd/MM/yyyy}", birthday);
Console.WriteLine("Diem 1:{0:N}, Diem 2:{1:N}, Diem 2:{2:N}", mark1,
mark2, mark3);
Console.WriteLine("Diem trung binh:{0:N}", average);
}
```

Bước 3: Nhấn Ctrl+F5 để chạy và xem kết quả



```
C:\Windows\system32\cmd.exe
Nhap ma so:SU2204
Nhap ten: Hoa Bat Tu
Nhap ngay sinh:02/24/1997
Nhap diem mon 1:8
Nhap diem mon 2:9
Nhap diem mon 3:9

Thông tin sinh viên
Ma so:SU2204
Ho va ten: Hoa Bat Tu
Ngày sinh:24/02/1997
Diem 1:8.00, Diem 2:9.00, Diem 2:9.00
Diem trung binh:8.67
Press any key to continue . . .
```

Phần II Bài tập tự làm

Bài 1.1: Viết chương trình C# in ra chữ sau:

```
+++++++      +++++      +++++
+++++++      +++++
++++      +++++      +++++
++++      +++++      +++++
++++      +++++      +++++
++++      +++++      +++++
++++      +++++      +++++
++++      +++++      +++++
++++      +++++      +++++
```

Biên dịch, chạy và kiểm tra kết quả tệp biên dịch trong thư mục bin/debug của ứng dụng.

Bài 1.2: Viết chương trình C# nhập vào các thông tin nhân viên (Id, Name, Address, Birthday, Salary, Bonus). In ra màn hình các thông tin nhân viên có định dạng và thêm trường TotalSalary=Salary+Bonus).

Bài 1.3: Đọc các nội dung tham khảo sau để nâng cấp level

Giới thiệu Microsoft .NET Framework

Ngày nay .NET Framework đã trở thành một trong công nghệ được sử dụng rộng rãi trên toàn thế giới. Theo số liệu thống kê không chính thức thì số lượng lập trình viên sử dụng ngôn ngữ C# của .NET cũng rơi vào khoảng 6 triệu. Đây là một sự tăng trưởng cực kỳ nhanh nếu như chúng ta biết rằng .NET mới chỉ ra đời được hơn 10 năm. Điều đấy cũng cho thấy mức độ hấp dẫn của .NET đối với lập trình viên lớn như thế nào.

.NET framework cùng với hệ sinh thái Microsoft đã phát triển rất nhanh với tất cả các công nghệ có thể giúp lập trình viên xây dựng nên các ứng dụng mất ít thời gian nhất, nó có thể giúp xây dựng từ ứng dụng chạy trên desktop cho đến ứng dụng web và gần đây là cả ứng dụng mobile/tablet. Cách xây dựng ứng dụng bằng .NET tương đối dễ dàng và tiện lợi so với các công nghệ khác vì Microsoft đã làm rất

tốt phần việc của mình khi giúp cho lập trình viên không còn phải can thiệp và hiểu quá sâu về hệ thống và cách thức hoạt động của .NET.

Tuy nhiên, để có thể nắm vững và sử dụng hiệu quả .NET, tránh những lỗi khó xử lý thì chúng ta cần phải hiểu được kiến trúc của .NET cũng như cách thức hoạt động của nó. Vì vậy trong thời gian tới chúng ta sẽ cùng thảo luận xoay quanh chủ đề này.

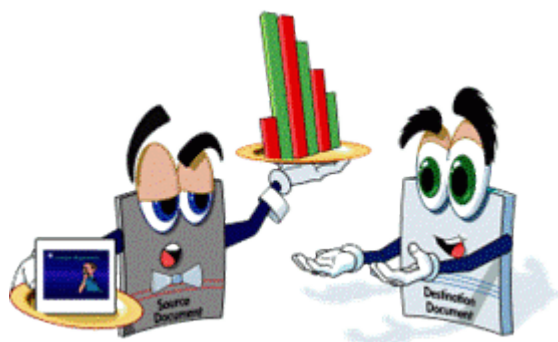
Bài viết đầu tiên này là giới thiệu cơ bản về .NET và các công nghệ tiền thân của nó để giúp chúng ta có một cái nhìn tổng quan hơn về các công nghệ Microsoft.

1. Công nghệ OLE

OLE (Object Linking and Embedding) là công nghệ ra đời đầu tiên của Microsoft trong những năm 90 để đơn giản hóa việc giao tiếp giữa các ứng dụng. Chúng hỗ trợ:

- ✓ Nhúng tài liệu từ một ứng dụng sang ứng dụng khác
- ✓ Cho phép một ứng dụng chỉnh sửa các đối tượng trong ứng dụng khác

OLE cho phép người dùng phát triển ứng dụng yêu cầu liên kết giữa các sản phẩm khác nhau VD: Word hay Excel.

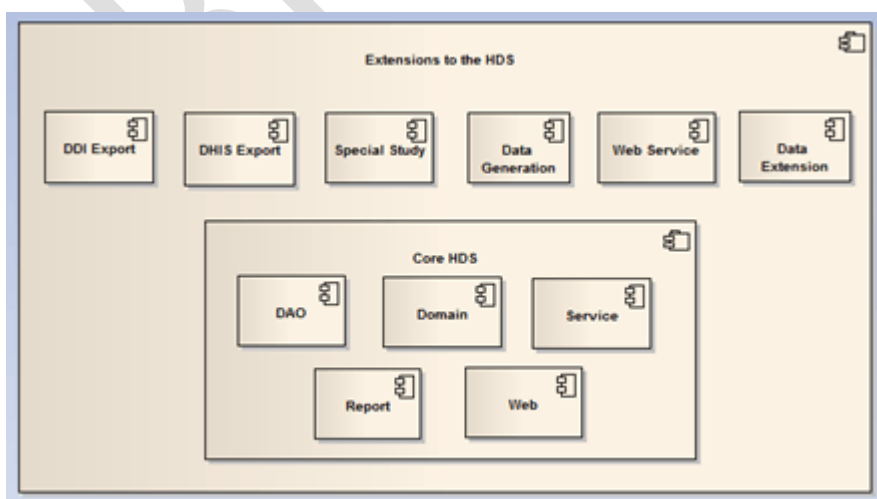


2. Công nghệ COM

Trước khi COM xuất hiện cách phát triển phần mềm thông thường là xây dựng thành một khối duy nhất. Thế nhưng khi chương trình lớn và phức tạp dần lên thì cách làm này dẫn tới một số vấn đề về bảo trì và kiểm thử phần mềm. Để giải quyết vấn đề này Microsoft đã tiến tới mô hình dựa trên các thành phần để phát triển phần mềm. Cách này đơn giản là chia phần mềm thành các module (các thành phần độc lập) mà mỗi module sẽ cung cấp một dịch vụ cụ thể. Mỗi module này có thể kiểm thử và phát triển độc lập sau đó tích hợp vào phần mềm chính. Kỹ thuật này được gọi là mô hình đối tượng dựa trên thành phần **COM (Component Object Model)**.

Mô hình này giúp cho việc phát triển phần mềm linh hoạt hơn:

- ✓ Giảm độ phức tạp của toàn bộ phần mềm
- ✓ Cho phép phát triển các module phân tán giữa nhiều nhóm, phòng ban...
- ✓ Tăng khả năng bảo trì phần mềm



3. Công nghệ .NET

Công nghệ .NET là mô hình dựa trên thành phần thể hệ thứ 3. Nó nâng cấp cách thức liên kết giữa các thành phần trong hệ thống so với công nghệ COM. Trong khi COM cung cấp một cơ chế nhị phân chuẩn để giao tiếp giữa các module thì .NET thay thế cơ chế này bởi một ngôn ngữ trung gian gọi là **Microsoft Intermediate Language (MSIL) hay IL**. Các trình biên dịch .NET khác nhau sẽ dịch code của các module thành mã IL nên sẽ tự động tương thích với các IL của module khác.

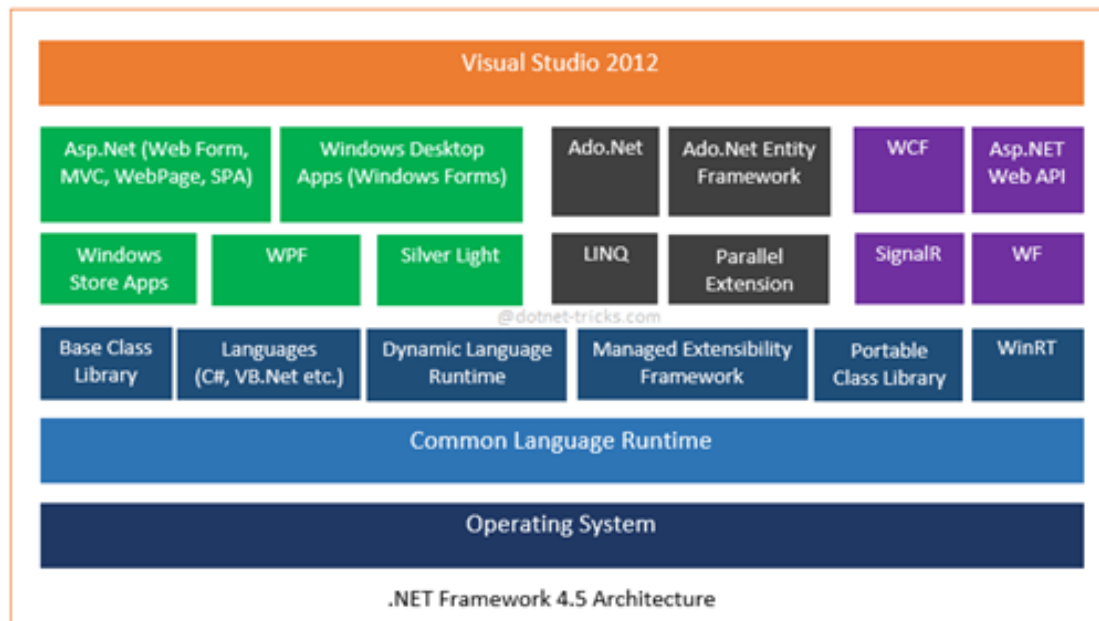
IL code có một thuộc tính là metadata: data về các dữ liệu và mô tả thuộc tính của nó: kiểu và vị trí. IL giúp cho việc tích hợp giữa các ngôn ngữ khác nhau trở nên dễ dàng. Ngoài IL, .NET còn bao gồm một loạt các công nghệ và công cụ khác giúp cho chúng ta phát triển và xây dựng các ứng dụng dễ dàng hơn. Chúng ta sẽ cùng xem qua bên dưới đây.

4. Cơ bản về .NET Framework

Đây là môi trường giúp cho việc xây dựng, phát triển và chạy các ứng dụng trên nền .NET, bao gồm tất cả các loại ứng dụng: Windows Form, Web, Silverlight, WPF, Windows Phone...

.NET Framework bao gồm 3 công nghệ khác nhau:

- ✓ **Common Language Runtime(CLR)**
- ✓ Các lớp cơ sở của framework (**Framework Based Classes – FBC**)
- ✓ Các giao diện chương trình (Web, Winform, Windows Phone ...)



a. Common Language Runtime (CLR)

CLR được coi như trái tim và linh hồn của .NET Framework. CLR như tên ám chỉ cung cấp một môi trường thực thi nơi mà các ứng dụng viết bằng .NET (C#, VB.NET, C++...) có thể chạy được.

CLR cung cấp một số dịch vụ như sau:

- ✓ Nạp và thực thi chương trình
- ✓ Phân chia vùng nhớ của ứng dụng
- ✓ Xác minh tính an toàn của kiểu dữ liệu
- ✓ Dịch mã IL thành mã máy thực thi được
- ✓ Cung cấp metadata
- ✓ Quản lý bộ nhớ tự động (automatic garbage collection)
- ✓ Thực thi bảo mật
- ✓ Quản lý lỗi và ngoại lệ
- ✓ Hỗ trợ các công việc như debug hoặc profile ứng dụng
- ✓ Liên kết với các hệ thống khác

b. Common Type System (CTS)

.NET framework hỗ trợ nhiều ngôn ngữ và đều dùng một thành phần gọi là hệ thống kiểu chung CTS trong CLR. CTS hỗ trợ một loạt kiểu và toán tử có thể thấy trong hầu hết các ngôn ngữ lập trình nên gọi một ngôn ngữ từ một ngôn ngữ khác sẽ không yêu cầu chuyển kiểu. Dẫn đến chúng ta có thể xây dựng các ứng dụng .NET sử dụng cả ngôn ngữ VB.NET lẫn C#, C++...

c. Common Language Specification (CLS)

Đặc tả ngôn ngữ chung CLS là một tập con của CTS, nó định nghĩa một tập các quy tắc cho phép liên kết hoạt động trên nền tảng .NET. Các quy tắc này sẽ trợ giúp và chỉ dẫn cho các nhà thiết kế compiler của hãng thứ 3 hoặc những người muốn xây dựng thư viện dùng chung.

d. Microsoft Intermediate Language (MSIL)

MSIL hay còn gọi IL là một tập lệnh mà tất cả các chương trình .NET được biên dịch thành. Nó trông hơi giống ngôn ngữ assembly và nó chứa các lệnh để nạp, lưu trữ, khởi tạo, gọi các phương thức trong chương trình. Khi chúng ta biên dịch một chương trình C# hoặc bất kỳ chương trình nào được viết bởi ngôn ngữ tuân theo CLS thì mã của nó sẽ là IL.

e. Managed Code

CLR chịu trách nhiệm quản lý việc thực thi mã được biên dịch trên nền tảng .NET. Mã chạy được trên môi trường thực thi CLR được gọi là managed code. Trình biên dịch tương thích với nền tảng .NET sẽ tạo ra managed code. Managed code được tạo bởi C# chính là IL code.

Trong bài viết này chúng ta đã cùng tìm hiểu về các công nghệ trước khi .NET xuất hiện cũng như tìm hiểu tổng quan về .NET framework, mà trong đó thành phần quan trọng nhất được xem như trái tim và linh hồn của .NET chính là CLR.

Tính năng mới của từng phiên bản .NET Framework

.NET Framework là một trong những framework được sử dụng rất rộng rãi nay để xây dựng các ứng dụng trong hệ sinh thái của Microsoft. Cùng với thời



hiện
dụng
gian

.NET framework ngày càng lớn lên và bổ sung rất nhiều tính năng vào mỗi phiên bản. Điều này đôi khi làm khó lập trình viên vì không cập nhật kịp dẫn đến là đi viết lại những thứ đã tồn tại trong framework (đây cũng là một trong những lý do mà rất nhiều lập trình viên Java ko thích .NET).

Vì vậy trong bài viết này chúng ta sẽ cùng điểm qua những tính năng mới qua từng phiên bản .NET framework để có một cái nhìn tổng quát về framework này.

1. .NET 1.0

➤ Common Language Runtime

Tính năng quan trọng nhất trong .NET framework đó chính là khả năng tạo ứng dụng sử dụng nhiều ngôn ngữ khác nhau. Tăng khả năng sử dụng lại (reuse) code. VD: Trong cùng một ứng dụng có thể module này được viết bằng C# nhưng module kia lại viết bằng VB.NET, C++...

Để làm được điều này thì tất cả ngôn ngữ mà .NET hỗ trợ (ước chừng gần 50 ngôn ngữ) đều được biên dịch thành ngôn ngữ trung gian (IL) và sau đó IL mới được dịch thành

ngôn ngữ máy. Tốc độ thực thi ứng dụng .NET khá nhanh, chỉ chậm hơn khoảng 5% so với ngôn ngữ máy và như ta đã thấy ở trên lợi ích mà .NET mang lại rất lớn.

➤ Garbage Collector

Trình dọn rác (GC) đảm bảo việc quản lý bộ nhớ trong ứng dụng .NET một cách tự động, nó giúp cho lập trình viên không còn phải quan tâm đến quản lý bộ nhớ thủ công như khi viết các ứng dụng C++ trước đây nữa (Tất nhiên điều này không phải tuyệt đối đúng – chúng ta sẽ bàn sâu hơn về vấn đề này ở các bài viết sau nói về quản lý bộ nhớ trong .NET). Ngoài ra kể từ .NET 4 GC được xử lý ở các thread riêng biệt vì vậy hiệu năng của ứng dụng .NET được tăng cường đáng kể.

➤ ADO.NET

ADO (ActiveX Data Objects) là một phần của COM (Component Object Model) dùng để truy cập các nguồn dữ liệu (Xem thêm về COM trong bài viết [Cơ bản về .NET](#)). ADO.NET là một bước tiến lớn so với ADO nhưng nó không có gì chung với ADO ngoại trừ cái tên 😊. Nói một cách ngắn gọn ADO.NET là một tập các API nằm trong namespace *System.Data* dùng để truy cập cơ sở dữ liệu. Các hệ quản trị cơ sở dữ liệu khác nhau như MS SQL, Oracle, DB2... sẽ cung cấp các provider ADO.NET khác nhau để làm việc với cơ sở dữ liệu tương ứng.

➤ String

Trong .NET thì chỉ có duy nhất một kiểu *string* được dùng để mô tả giá trị chuỗi. Chuỗi được biểu diễn dưới định dạng UTF-8 và trong bộ nhớ chỉ tồn tại một thể hiện (instance). Mỗi khi chúng ta trích xuất một chuỗi con từ chuỗi ban đầu thì sẽ xuất hiện một instance mới chứ không phải thay đổi chuỗi ban đầu (đặc tính của kiểu immutable type)

2..NET 1.1

➤ Regular Expressions

Biểu thức chính quy cung cấp một kỹ thuật mạnh mẽ, linh hoạt và rất hiệu quả cho việc xử lý chuỗi. Bất cứ khi nào bạn cần lấy một chuỗi con từ một chuỗi cho trước, hãy nghĩ ngay đến việc sử dụng biểu thức chính quy. Lý thuyết về biểu thức chính quy trông có vẻ phức tạp lúc mới tiếp cận nhưng nó thực sự rất đáng giá khi nắm được. Chúng ta sẽ nghiên cứu về biểu thức chính quy trong các bài viết sau.

➤ XML Processing

Lớp *XmlDocument* trong namespace *System.Xml* cung cấp các API để phân tích và đọc các tài liệu XML. Nó cũng cung cấp ngôn ngữ XPath dùng để trích xuất dữ liệu từ tài liệu XML.

➤ HttpUtility

HttpUtility là một lớp rất quan trọng dùng để mã hóa và giải mã tham số URL khi xử lý các web request. Đặc biệt nó có một phương thức vô cùng hữu dụng – *HtmlDecode* dùng để thay các thẻ HTML bằng các kí tự Unicode tương ứng (phòng tránh các lỗi bảo mật như XSS, SQL Injection...)

➤ String Builder

Một lớp hữu dụng khi xử lý việc kết hợp nhiều chuỗi (thường khi kết hợp lớn hơn 3 chuỗi), đặc biệt là khi kết hợp chuỗi trong vòng lặp lớn bởi vì kiến trúc bên trong String Builder dùng mảng kí tự và đã được Microsoft tối ưu hóa về hiệu năng.

3..NET 2.0

➤ Generics

Các thuật toán Generic được trừu tượng hóa và yêu cầu thành phần kiểu. Thành phần này có thể được mô tả bởi interface hoặc lớp trừu tượng (abstract class). VD:

Generic *IEnumerable<T>* là một tập các object kiểu T. Generic rất hữu ích khi chúng ta

cần viết những lớp chung như lớp base hoặc các phương thức dùng chung an toàn về kiểu.

➤ Event Log

EventLog là hệ thống ghi nhật ký thống nhất trên Windows. Thay vì quản lý việc tạo ra và xóa các file log hoặc lọc và phân loại dữ liệu log thì chúng ta có thể sử dụng lớp *EventLog* với các API được cấp sẵn để thay thế.

➤ Configuration Manager

Lớp *ConfigurationManager* chứa chuỗi kết nối và các hằng số bạn muốn thay đổi lúc chạy ứng dụng mà không phải biên dịch lại ứng dụng. VD: Trong ứng dụng Winform thì là file App.config còn ứng dụng ASP.NET là web.config.

➤ Nullable Types

Trong .NET có 2 kiểu dữ liệu là kiểu giá trị (Value Type) và kiểu tham chiếu (Reference Type). Kiểu giá trị là các kiểu cơ sở của .NET: int, long, byte, bool, short, char, structure... Kiểu tham chiếu là các đối tượng. (Chúng ta sẽ tìm hiểu kỹ hơn về 2 kiểu dữ liệu này ở các bài viết sau)

Kiểu tham chiếu với giá trị null mang ý nghĩa là không chỉ đến đối tượng nào. Với kiểu giá trị có trường hợp khi lấy dữ liệu từ SQL Server một trường BIT có thể mang giá trị là true, false hoặc undefined vì vậy cần kiểu nullable để tương thích. VD khai báo kiểu giá trị nullable: `int? x = null;`

4..NET 3.0

➤ XAML (Silverlight/WPF/Windows Phone/Windows RT)

XAML cho phép chúng ta vẽ các giao diện ứng dụng giống như sử dụng HTML. Cách vẽ giao diện qua việc khai báo thẻ như HTML có khá nhiều ưu điểm mà nổi bật trong số đó chính là binding – một thành phần không thể thiếu của mô hình phát triển ứng dụng

MVVM. Bên cạnh đó XAML được tăng tốc bởi GPU nên hiệu năng của ứng dụng khá tốt. XAML được áp dụng để thiết kế giao diện chính cho hầu hết các loại ứng dụng của Microsoft như: WPF/Silverlight/Windows Phone/Windows RT.

➤ WCF services

Web service giúp cho việc gọi các phương thức từ server trở nên dễ dàng hơn cho client. Trong .NET 3.0 Web services được chuẩn hóa thành WCF services với nhiều tính năng cao cấp hơn so với web service thông thường trong .NET 2.0

➤ Workflow

Workflow là một mô hình các hành động (activity). Các hoạt động này có thể là bất kỳ thứ gì từ tiến trình nghiệp vụ cho đến tiến trình build ứng dụng của Team Foundation Server. Ưu điểm của Workflow là các hành động được định nghĩa bởi XAML nên nó dễ dàng thay đổi bởi file cấu hình mà không cần phải biên dịch lại ứng dụng.

➤ Feed processing

Lớp SyndicationFeed hỗ trợ định dạng syndication RSS 2.0 và Atom 1.0

5..NET 3.5

➤ Language Integrated Query (LINQ)

LINQ là một trong những ngôn ngữ cách mạng cho việc xử lý dữ liệu, nó giống như SQL cho lập trình hướng đối tượng. Tất cả những store procedure được viết trong SQL Server thì giờ chúng ta hoàn toàn có thể viết trên code .NET, hết sức tiện lợi trong việc viết cũng như debug. Tuy nhiên nhược điểm khi sử dụng LINQ là hiệu năng sẽ thấp hơn so với khi sử dụng store procedure.

➤ Entity Framework

Khi sử dụng LINQ thì một thành phần không thể thiếu đó chính là Entity Framework. Một framework giúp chúng ta có thể mapping được ngay các bảng trong SQL server thành các đối tượng .NET để LINQ có thể sử dụng được.

➤ XML processing

Với sự xuất hiện của LINQ và lớp *Xdocument* trong namespace *System.Xml.Linq* việc xử lý XML trở nên dễ dàng, tiện lợi với hiệu năng cao hơn.

➤ Reactive Extensions (Rx)

Rx là một thư viện dùng tạo các chương trình asynchronous dựa trên sự kiện và sử dụng observable collection + LINQ để truy vấn. Rx được tạo ra để làm việc với data stream.

6..NET 4.0

➤ Managed Extensibility Framework (MEF)

MEF cho phép sử dụng các assembly (.dll) như là các module. Khi bạn cần tạo ra các hệ thống module hóa MEF sẽ đảm bảo việc kết nối giữa các module này.

➤ Parallel Extensions (PE)

PE bao gồm Parallel LINQ (PLINQ) và Task Parallel Library (TPL). Khi bạn có nhiều việc cần làm thì có thể chọn hướng tiếp cận dựa trên task. Chia nhỏ công việc thành các việc nhỏ hơn và chạy chúng khi cần. Có thể chạy tuần tự hoặc song song mà không cần quan tâm đến việc quản lý thread. Lớp *Parallel* sẽ đảm nhận việc đếm xem có bao nhiêu bộ vi xử lý để chọn số thread hiệu quả nhất.

➤ Dynamic Language Runtime

Bạn có thể dùng kiểu *dynamic* khi cần tương tác với môi trường thiếu kiểu, ngôn ngữ động... VD: Nó sẽ rất hữu ích khi tương tác với môi trường COM vì nó không cung cấp IntelliSense

➤ Lazy initialization

Khi chương trình chứa đối tượng chiếm nhiều thời gian để khởi tạo mà đối tượng này có thể không được dùng trong suốt chương trình thì chúng ta có thể dùng khởi tạo muộn *Lazy<T>*, đối tượng sẽ chỉ được khởi tạo khi truy cập nó lần đầu tiên

7..NET 4.5

➤ Asynchronous Pattern

Lập trình bất đồng bộ có thể thực hiện từ các phiên bản .NET trước nhưng phải đến .NET 4.5 với sự xuất hiện của *async/await* thì nó mới thực sự trở nên đơn giản với lập trình viên. Với .NET trước đây khi thread đợi server trả lời, nó không làm gì những vẫn tiêu tốn tài nguyên máy. Với *async/await* điều này không còn xảy ra vì nó thực sự không tạo ra thread mới khi thực hiện các lời gọi *async*.

C# Evolution Matrix (By Raymond Tang Http://kosmisch.net)		
	Version/IDE	Key New Features
C# 1.0	VS2002	Managed Code
C# 2.0	VS2005	Generics Anonymous Methods Nullable Types
C# 3.0	VS2008	Lambda Expressions Extension Methods Expression Tree Anonymous Types LINQ Implicit Typing (var)
C# 4.0	VS2010	Late Binding (dynamic) Named Arguments Optional Parameters More COM Support
C# 5.0	VS2011	Async Feature Caller Information

HẾT