Getting Started – Bắt Đầu:

1. Phần Mở Rộng File C#?

* “.cs”

1. C# Có Phân Biệt Chữ Hoa Thường Không?

* Có

1. Cách Biên Dịch?

* Bạn đéo cần cài thêm trình biên dịch nào vì Windows đã tích hợp sẵn
* Trong thư mục “C:\Windows\Microsoft.NET\Framework\v4.0.30319\” sẽ chứa File “csc.exe” là trình biên dịch
* Cho thư mục này vào biến hệ thống Path
* Chạy lệnh sau để biên dịch File C# ra File EXE

csc /out:<Đường Dẫn Đến File EXE Có Phần Mở Rộng> <Đường Dẫn Đến Các File C# Có Phần Mở Rộng>

* Ví dụ

csc /out:gg\foo.exe bar\bob\bab.cs thanglon\thangngu.cs

* Khi này File “bab.cs” và “thangngu.cs” sẽ được gộp chung lại thành 1 File C# duy nhất, gọi là A, sau đó A sẽ được biên dịch thành “foo.exe”
* Nếu không chỉ định <Đường Dẫn Đến File EXE Có Phần Mở Rộng> thì mặc định File EXE sẽ tạo ra trong thư mục làm việc hiện tại, cùng tên với File C# là ngõ vào chương trình
* Ví dụ

csc bob.cs bab.cs gg\haha.cs

* Giả sử “bab.cs” chứa ngõ vào, thì File tạo ra là “bab.exe”
* Ngõ vào chương trình sẽ là hàm tên “Main”, kiểu static
* Hàm này luôn được chạy đầu tiên dù cho nó đặt bất cứ chỗ nào
* Trong A chỉ được phép có nhiều nhất 1 hàm dạng này

1. DLL (Dynamic Link Library)?

* Là 1 File chứa các Class, hàm, thuộc tính, …
* Khi 1 Process tải 1 DLL vào bộ nhớ, thực chất nó đang tải vào vùng địa chỉ ảo, vùng địa chỉ này thông qua Paged Memory ánh xạ tới 1 vùng nhớ tại RAM, vùng này mới là vùng lưu DLL, đồng thời khi nhiều Process tải DLL này, thì trong RAM luôn chỉ tồn tại 1 vùng nhớ lưu DLL, không bị Copy cho từng Process, chỉ địa chỉ ảo của DLL trong các Process là khác nhau, các địa chỉ ảo này thông qua Paged Memory ánh xạ vào cùng 1 vùng nhớ trong RAM

1. Biên Dịch Ra File DLL?

csc /target:library /out:<Đường Dẫn Đến File DLL Có Phần Mở Rộng> <Đường Dẫn Đến Các File C# Có Phần Mở Rộng>

* Ví dụ

csc /target:library /out:connguoi.dll concac.cs conbuom.cs

* Các File C# sẽ được gộp chung lại như khi biên dịch ra EXE, tuy nhiên không cần sự tồn tại của ngõ vào chương trình
* Nếu <Đường Dẫn Đến File DLL Có Phần Mở Rộng> không được chỉ định, thì nó sẽ có tên = tên của File C# đầu tiên được liệt kê

1. Tải File DLL Vào Chương Trình?

* Bước 1, tải DLL vào bộ nhớ ảo, đồng thời tạo 1 Handle đến nó

System.Reflection.Assembly <Handle> = System.Reflection.Assembly.LoadFile(<Đường Dẫn Tuyệt Đối Đến File DLL Có Phần Mở Rộng>);

* Ví dụ

System.Reflection.Assembly concu = System.Reflection.Assembly.LoadFile("C:\\Users\\pc\\Desktop\\C# Test\\concac.dll");

* Bước 2, trích xuất 1 Class trong DLL

System.Type <Type> = <Handle>.GetType(<Namespace Và Tên Class>);

* Nếu Class không đặt trong Namespace thì không cần ghi Namespace
* Ví dụ
* Trong File C# ứng với “concac.dll”, ta có tạo 1 Class bar trong Namespace alice, ta sẽ trích xuất Class này như sau

System.Type haha = concu.GetType("alice.bar");

* Bước 3, tạo 1 Object của Class vừa trích xuất

System.Object <Object> = System.Activator.CreateInstance(<Type>);

* Ví dụ

System.Object thangcu = System.Activator.CreateInstance(haha);

* Bước 4, trích xuất 1 hàm của Class vừa trích xuất

System.Reflection.MethodInfo <Hàm> = <Type>.GetMethod(<Tên Hàm>);

* Ví dụ trong Class bar ta có tạo hàm john, để trích xuất hàm này

System.Reflection.MethodInfo wick = haha.GetMethod("john");

* Để gọi hàm trên Object vừa tạo cùng với các tham số cụ thể

<Hàm>.Invoke(<Object>, <Mảng>);

* <Mảng> là 1 mảng chứa các đối số, cho là null nếu hàm không có đối số
* Ví dụ

wick.Invoke(thangcu, null);

Namespace:

1. Cách Hoạt Động?

* Đã có sẵn một vài Namespace mặc định như System, Math, …
* Mục đích là để gom 1 đống Class hoặc hàm, thuộc tính thành 1 khối riêng
* Lệnh using trong C# = using namespace trong C++, chỉ áp dụng cho Namespace lồng nhau, không áp dụng cho Class lồng nhau
* Ví dụ thay vì viết

foo.bar.bob.alice();

* Thì có thể viết như sau, giả sử foo, bar là Namespace, bob là Class, alice là hàm

using foo.bar;

bob.alice();

1. Tạo 1 Namespace?

namespace <Tên Namespace> {

<Các Class, Hàm, Thuộc Tính>

}

* Dùng dấu chấm để truy cập vào bên trong

Variable – Biến:

1. Kiểu Dữ Liệu?

* Giống Java, chỉ có “String” đổi thành “string”

1. Trả Về Kiểu Dữ Liệu Của 1 Object Dưới Dạng String Tại Runtime?

<Object>.GetType()

1. Khai Báo Biến Mà Kiểu Dữ Liệu Mình Không Biết?

System.Object <Biến> = <Giá Trị Khởi Tạo>;

* <Giá Trị Khởi Tạo> là giá trị nào đó mình đéo biết kiểu dữ liệu
* System.Object là kiểu dữ liệu chung nhất, nó có thể chứa bất kì dữ liệu nào, khi Runtime, <Biến> tự động có được kiểu dữ liệu cụ thể

String – Chuỗi:

1. Trả Về Kiểu Viết Thường Của 1 String?

<String>.ToLower()

1. Check Xem 1 String Có Chứa 1 String Con Không?

<String>.Contains(<String Con>)

* Trả về True nếu chứa, False nếu không chứa

1. Trả Về Index Của String Con Trong 1 String?

<String>.IndexOf(<String Con>)

* Nếu không tìm thấy, trả về –1
* Nếu tìm thấy nhiều, trả về cái đầu tiên
* Index là Index của kí tự đầu tiên trong <String Con>

1. Trả Về String Con Của 1 String?

<String>.Substring(<Start Index>, <Số Kí Tự>)

* Nếu không chỉ định <Số Kí Tự> thì sẽ lấy hết từ <Start Index> đến cuối

Console:

1. In String Rồi Xuống Dòng?

System.Console.WriteLine(<String>);

* Giống print trong Python