

LESSON 01

I. NGÔN NGỮ C#

❖ Sơ lược về ngôn ngữ C#

- C# (đọc là “C thăng” hay “C sharp” (“xi-sáp”)) là một ngôn ngữ lập trình **thuần hướng đối tượng** được phát triển bởi Microsoft.
- C# ra đời năm 2000, được thiết kế chủ yếu bởi **Anders Hejlsberg** – kiến trúc sư phần mềm nổi tiếng với các sản phẩm Turbo Pascal, Delphi, . . .
- Được xây dựng dựa trên nền tảng của 2 ngôn ngữ lập trình mạnh nhất đó là C++ và Java. Do đó C# được miêu tả là ngôn ngữ có sự cân bằng giữa C++, Visual Basic, Delphi và Java.
- C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng

❖ Ngôn ngữ lập trình C# có những đặc trưng cơ bản sau:

- Là một ngôn ngữ thuần hướng đối tượng (trình bày sau)
- Là ngôn ngữ khá đơn giản, chỉ có khoảng 80 từ khóa và hơn mười mấy kiểu dữ liệu được dựng sẵn.
- Cung cấp những đặc tính hướng thành phần (component-oriented) như là Property, Event (trình bày sau)
- C# có bộ **Garbage Collector** sẽ tự động thu gom vùng nhớ khi không còn sử dụng nữa
- C# đã loại bỏ đa kế thừa trong C++ mà thay vào đó C# sẽ hỗ trợ thực thi giao diện interface (trình bày sau).

❖ Một số ưu điểm nổi bật của C#:

- Gần gũi với các ngôn ngữ lập trình thông dụng (C++, Java, Pascal).
- Xây dựng dựa trên nền tảng của các ngôn ngữ lập trình mạnh nên thừa hưởng những ưu điểm của những ngôn ngữ đó.
- Cải tiến các khuyết điểm của C/C++ như con trỏ, các hiệu ứng phụ, . . .
- Dễ tiếp cận, dễ phát triển.
- Được sự chống lưng của .NET Framework.

❖ Một số nhược điểm của C#:

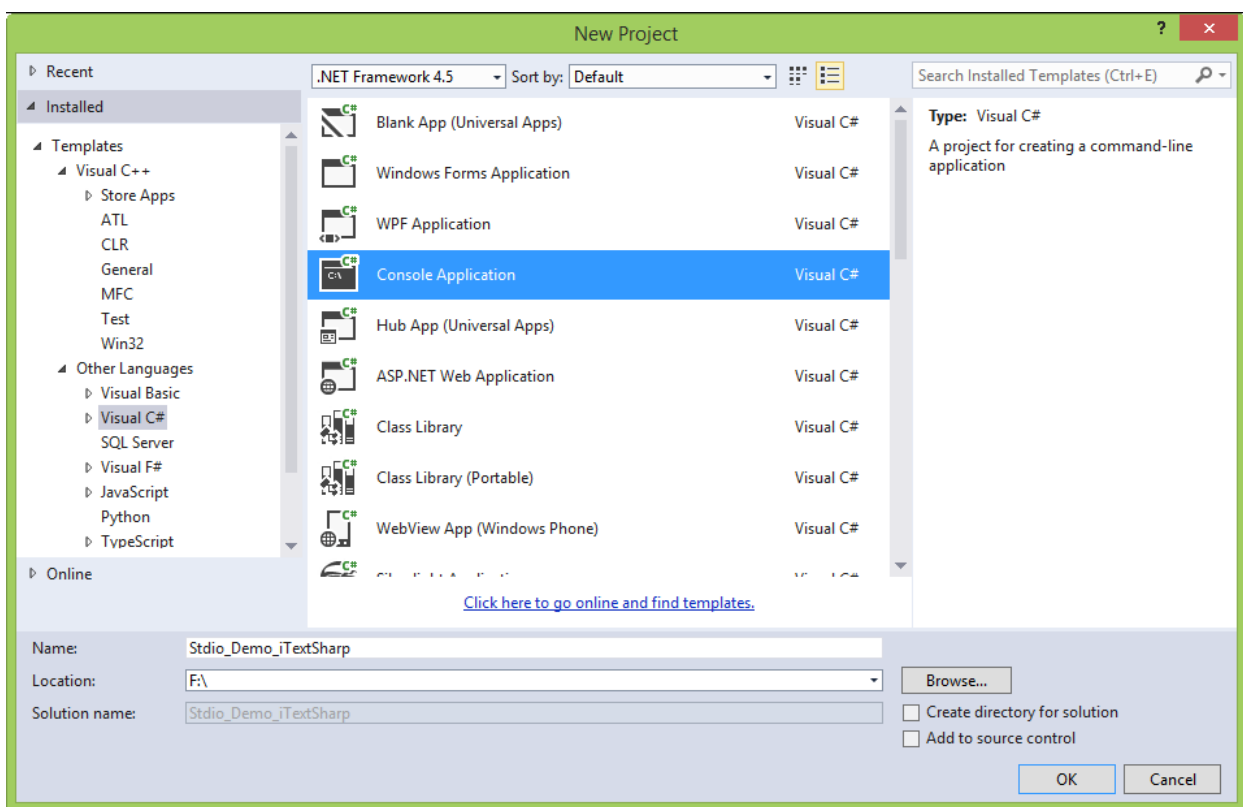
- Nhược điểm lớn nhất của C# là chỉ chạy trên nền Windows và có cài .NET Framework
- Thao tác đối với phần cứng yếu hơn so với ngôn ngữ khác. Hầu hết phải dựa vào windows.

<HƯỚNG DẪN CÀI ĐẶT PHẦN MỀM VISUAL STUDIO COMMUNITY>

II. CẤU TRÚC LỆNH

Đầu tiên, để viết chương trình C# trên nền Console Application ta cần tạo một project Console Application như sau:

- File > New.. > Project..
- Tìm đến project của C# và chọn Console Application.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Cau_Truc_Lenh_Co_Ban
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

❖ Using

➤ Cú pháp

using <tên thư viện>

- Dùng để chỉ cho trình biên dịch biết rằng những thư viện (thư viện là một tập các phương thức, kiểu dữ liệu nào đó được tạo ra nhằm hỗ trợ cho việc lập trình nhanh chóng hiệu quả hơn. Chúng ta sẽ tìm hiểu kỹ hơn ở những bài sau) được sử dụng trong chương trình. Các bạn hoàn toàn có thể không sử dụng thư viện nào trong chương trình của mình
- Ví dụ

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

❖ Namespace

➤ Cú pháp

```
namespace <tên namespace>
{
    // Các thành phần bên trong namespace bao gồm các lớp,
    // enum, delegate hoặc các
    // namespace con
}
```

- báo cho trình biên dịch biết rằng các thành phần bên trong khối { } ngay bên dưới tên namespace thuộc vào chính namespace đó. Chi tiết sẽ được trình bày rõ hơn trong các bài sau

➤ Ví dụ:

```
namespace Cau_Truc_Lenh_Co_Ban
{
    public class Action { }
    public delegate void Art();
    namespace Sub_Namespace { }
}
```

❖ Class

➤ Cú pháp

```
class <Tên lớp> { }
```

- báo cho trình biên dịch biết rằng những thành phần trong khối { } ngay sau tên lớp thuộc vào chính lớp đó. Chi tiết về lớp sẽ được trình bày trong bài CLASS TRONG C#

➤ Ví dụ:

```
class Program
{
    static void Main(string[] args)
    {
    }
}
```

Dễ thấy phương thức Main này khối { } của lớp **Program** nên phương thức này thuộc lớp **Program**.

❖ Method (Phương thức) Main

```
static void Main(string[] args) { }
```

➤ Hàm chính của toàn chương trình. Mỗi khi trình biên dịch dịch chương trình ra sẽ đi vào hàm Main đầu tiên để bắt đầu vòng đời của chương trình. Từ thời điểm này chúng ta sẽ viết code (mã chương trình) bên trong khối { } của hàm Main.

❖ Comment (Ghi chú)

- Đôi khi bạn không nhớ đoạn code mình viết ra dùng để làm gì. Thì chú thích lại ý nghĩa của nó cũng rất cần thiết.
- Hay bạn có thể đóng đoạn code không dùng tới mà không cần xóa nó đi. Khi nào cần sử dụng thì lại mở nó ra sửa lại
- Comment không được biên dịch khi dịch chương trình

Có 3 cách để comment code trong Visual Studio:

🔧 Sử dụng ký tự // - Cmt 1 dòng

Bất kỳ đoạn code hay chữ nào phía sau ký tự **//** cũng sẽ không được biên dịch

```
// Comment cho 1 dòng
// Console.Write("TT Design");
// Hoàn toàn có thể comment như thế này.
```

🚩 Sử dụng ký tự `/**/` - Cmt Nhiều dòng

Bất kỳ đoạn code hay chữ nào nằm trong khối `/**/` đều tính là comment. Mỗi khi xuống dòng thì vẫn là comment

```
Console.ReadKey(/*haha đoạn comment này không được biên dịch*/);  
/*Comment  
Hay như thế này*/
```

🚩 Sử dụng ký tự `///`

Thêm 1 cách comment code để tiện sử dụng nữa là ký tự `///`. Bạn gõ ký tự này ở phía trên namespace, class, method thì Visual Studio sẽ tự động sinh ra cho bạn 1 đoạn comment như sau:

```
1  using System;  
2  using System.Collections.Generic;  
3  using System.Linq;  
4  using System.Text;  
5  using System.Threading.Tasks;  
6  
7  namespace FirstConsoleApp  
8  {  
9      /// <summary>  
10     /// Class for first program  
11     /// </summary>  
12     0 references  
13     class Program  
14     {  
15         0 references  
16         static void Main(string[] args)  
17         {  
18             Run("phamhoan");  
19             Console.ReadKey();  
20         }  
21         /// <summary>  
22         /// This is first app by pch  
23         /// </summary>  
24         /// <param name="run"> For Running man</param>  
25         1 reference  
26         static void Run(string run)  
27         {  
28             Console.WriteLine("Pham Hoan xda: " + run);  
29         }  
30     }  
31 }
```

```
/// <summary>/// Bạn có thể ghi bất kỳ trong nơi này/// </summary>///  
<param name="args"></param>
```

❖ Dấu chấm phẩy (;)

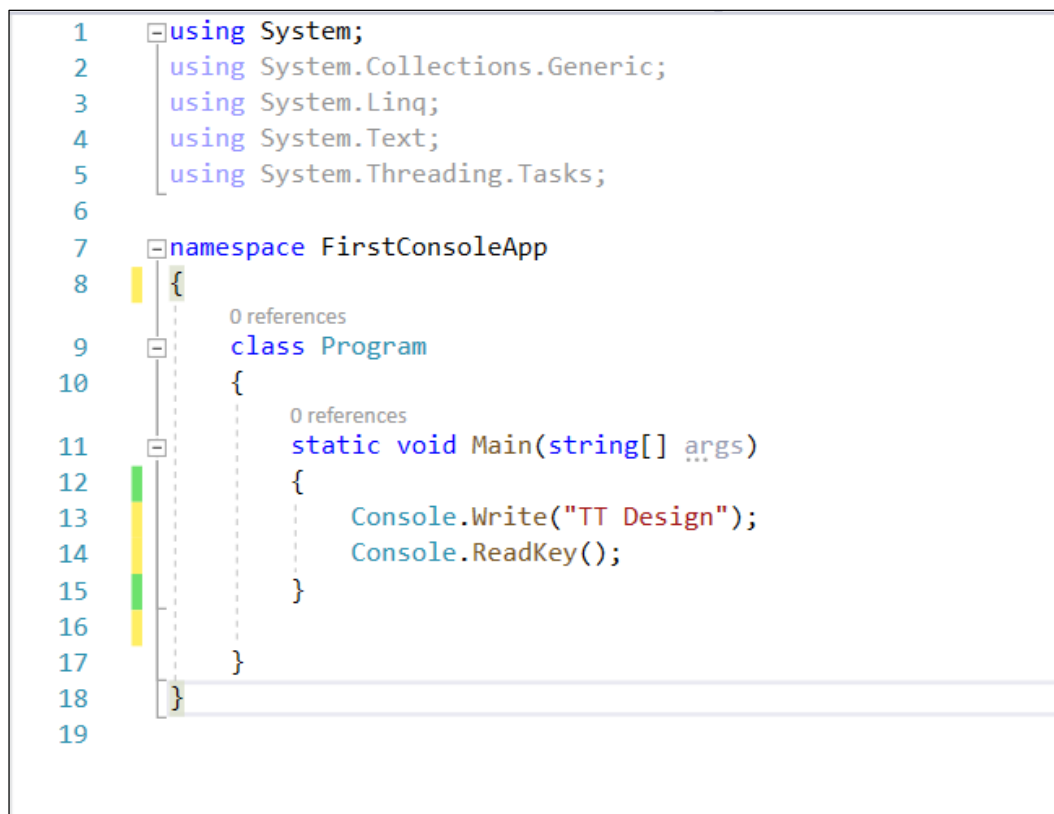
Có một điểm cần lưu ý khi viết code. Mỗi khi kết thúc một dòng lệnh. Chúng ta sẽ viết thêm 1 dấu ; ngay phía sau đoạn code đó để báo hiệu chúng ta đã kết thúc dòng lệnh hiện tại.

Bạn hoàn toàn có thể viết tiếp dòng lệnh tiếp theo ngay trên cùng 1 hàng với dòng lệnh cũ. Nhưng khuyến cáo không nên, để code rõ ràng.

```
Console.Write("TT Design"); // dấu ; ngay cuối dòng lệnh
Console.Write("TT Design "); Console.ReadKey();
// không nên viết nhiều đoạn code trên 1 hàng như vậy
```

❖ Ví dụ chương trình đầu tiên

Đầu tiên, các bạn tạo một project mới và nhập đoạn code sau vào:



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace FirstConsoleApp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.Write("TT Design");
14             Console.ReadKey();
15         }
16     }
17 }
18
19
```

- Sau đó nhấn F7 để biên dịch và nhấn F5 để chạy.
- Ta được kết quả là “TT Design” trên màn hình Console

III. NHẬP XUẤT CƠ BẢN TRONG C#

Trong C# có 5 lệnh dùng để nhập xuất đó là:

```
Console.Write();

Console.WriteLine();

Console.Read();

Console.ReadLine();

Console.ReadKey();
```

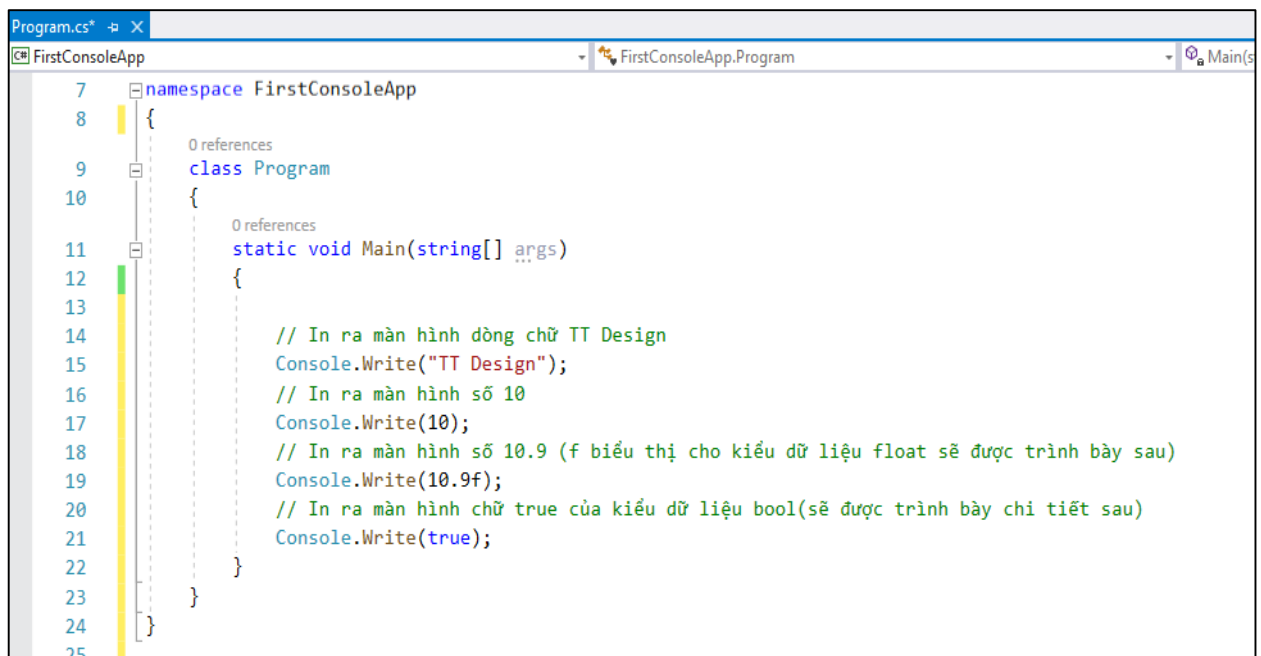
❖ Console.Write();

➤ Cú pháp:

```
Console.Write(<giá trị cần in ra màn hình>);
```

➤ In giá trị ra màn hình console. Giá trị này có thể là 1 ký tự, 1 chuỗi, một giá trị có thể chuyển về kiểu chuỗi (Trình bày sau)

➤ Ví dụ:



```
Program.cs
FirstConsoleApp
FirstConsoleApp.Program
Main(s)

7 namespace FirstConsoleApp
8 {
9     0 references
10    class Program
11    {
12        0 references
13        static void Main(string[] args)
14        {
15            // In ra màn hình dòng chữ TT Design
16            Console.Write("TT Design");
17            // In ra màn hình số 10
18            Console.Write(10);
19            // In ra màn hình số 10.9 (f biểu thị cho kiểu dữ liệu float sẽ được trình bày sau)
20            Console.Write(10.9f);
21            // In ra màn hình chữ true của kiểu dữ liệu bool(sẽ được trình bày chi tiết sau)
22            Console.Write(true);
23        }
24    }
25 }
```

➤ Thực hiện chạy chương trình thì ta thấy màn hình console vừa hiện lên đã tắt. Vậy làm sao để xem được kết quả?

- Ta thực hiện chạy chương trình bằng cách vào Debug > Run Without Debugging. (Phím tắt Ctrl + F5).
- Chúng ta chỉ cần thêm 1 trong 3 lệnh trên vào cuối chương trình là xong. Ý nghĩa của 3 lệnh trên sẽ được giải thích chi tiết ở phần sau trong bài học này.

```
Console.Read();  
Console.ReadLine();  
Console.ReadKey();
```

➤ Chạy lại Chương trình.

❖ Console.WriteLine();

➤ Cú pháp:

```
Console.WriteLine(<giá trị cần in ra màn hình>);
```

Lệnh này cũng tương tự như `Console.Write()`

Khác: khi in giá trị ra màn hình xong nó sẽ tự động đưa con trỏ xuống dòng.

Điều này giúp ta có thể giải quyết được vấn đề đã đặt ra ở phần trên.

Ngoài ra, để xuống dòng ta còn có nhiều cách khác như:

- Sử dụng ký tự đặc biệt: chúng ta sử dụng ký tự “\n” trong chuỗi in ra màn hình thì trình biên dịch sẽ tự động đổi nó thành ký tự xuống dòng.
- Như vậy thay vì dùng `Console.WriteLine(“TT Design”)` ta có thể dùng `Console.Write(“TT Design \n”)`.
- Các ký tự đặc biệt sẽ được giới thiệu trong phần sau của bài học.

➤ Ví dụ:

```
Program.cs* -# X
FirstConsoleApp FirstConsoleApp.Program
7 namespace FirstConsoleApp
8 {
9     0 references
10    class Program
11    {
12        0 references
13        static void Main(string[] args)
14        {
15            Console.WriteLine("TT Design \n"); // Sử dụng ký tự đặc biệt để xuống dòng
16            Console.WriteLine(5); // Sử dụng lệnh in ra màn hình có xuống dòng
17
18            Console.Write(6.5f); // In ra giá trị nhưng không xuống dòng
19            Console.Write(true);
20            Console.ReadLine();
21        }
22    }
23 }
```

+ **Console.Write**(<giá trị cần in ra màn hình>): in giá trị ra màn hình nhưng không đưa con trỏ xuống dòng.

+ **Console.WriteLine**(<giá trị cần in ra màn hình>): in giá trị ra màn hình và đưa con trỏ xuống dòng.

❖ Cộng dồn chuỗi in ra màn hình:

```
int a = 5; // khai báo biến kiểu nguyên có tên là a và khởi tạo giá trị là 5.
Console.Write("a = "); // In ra màn hình giá trị "a = ".
Console.Write(a); // In ra giá trị của a là 5

// Kết quả màn hình là: a = 5
```

Thì ta có thể viết gọn lại là **Console.Write("a = " + a);** vẫn in ra màn hình a = 5

❖ In ra giá trị của biến:

Ngoài ra ta cũng có thể chỉ định vị trí in ra giá trị của biến trong chuỗi bằng cú pháp {<số đếm>}.

```
int a1 = 5; // khai báo biến kiểu nguyên có tên là a và khởi tạo giá trị là 5.
Console.WriteLine("a = {0}", a); // In ra màn hình giá trị "a = 5".
```

Cú pháp:

```
Console.WriteLine("{0} {1} {2} {...}", <giá trị 0>, <giá trị 1>, <giá trị 2>, <giá trị n>);
```

❖ Console.Read();

➤ Cú pháp:

```
Console.Read();
```

Đọc 1 ký tự từ bàn phím và trả về **kiểu số nguyên** (là mã ASCII (American Standard Code for Information Interchange - Chuẩn mã trao đổi thông tin Hoa Kỳ, là bộ ký tự và bộ mã ký tự dựa trên bảng chữ cái La Tinh được dùng trong tiếng Anh hiện đại và các ngôn ngữ Tây Âu khác) của ký tự đó.

Link: <https://vi.wikipedia.org/wiki/ASCII>

Chú ý: lệnh này không đọc được các phím chức năng như Ctrl, Shift, Alt, Caps Lock, Tab,...

➤ Ví dụ:

```
{  
    Console.WriteLine(Console.Read());  
    /*  
    đọc 1 ký tự từ bàn phím bằng lệnh  
    Console.Read();  
    sau đó in ra ký tự vừa đọc được.  
    */  
    Console.ReadKey(); // lệnh này dùng với mục đích dừng màn hình để xem kết quả.  
}
```

❖ Console.ReadLine();

➤ Cú pháp:

```
Console.ReadLine();
```

Đọc dữ liệu từ bàn phím cho đến khi gặp ký tự xuống dòng thì dừng (Nói cách khác là đọc cho đến khi mình nhấn enter thì dừng) và giá trị đọc được luôn là một chuỗi.

➤ Ví dụ:

```
static void Main(string[] args)
{
    Console.WriteLine(Console.ReadLine()); // đọc dữ liệu từ bàn phím cho
    đến khi gặp ký tự xuống dòng thì dừng. Sau đó in giá trị đã nhập ra màn
    hình.
    Console.ReadKey(); // lệnh này dùng với mục đích dừng màn hình để xem
    kết quả.
}
```

❖ Console.ReadKey();

➤ Cú pháp:

```
Console.ReadKey();
```

- Lệnh này cũng dùng để đọc một ký tự từ bàn phím nhưng trả về kiểu **ConsoleKeyInfo** (là một kiểu dữ liệu có cấu trúc được định nghĩa sẵn để chứa những ký tự của bàn phím bao gồm các phím chức năng).
- Tham số kiểu bool bao gồm 2 giá trị: **true hoặc false**.
- Nếu truyền vào **true** thì phím được ấn sẽ không hiển thị lên màn hình console mà được đọc ngấm ngược lại thì phím được ấn sẽ hiển thị lên màn hình console
- Nếu không truyền tham số vào thì mặc định sẽ là false.

➤ Ví dụ:

```
Console.WriteLine("TT Design");
Console.ReadKey(); // không truyền tham số vào thì mặc định là false.
Console.ReadKey(false); // hiển thị phím ấn lên màn hình.
Console.ReadKey(true); // Không hiển thị phím ấn lên màn hình.
```

BÀI TẬP:

1. Viết chương trình cho phép người dùng nhập tên của mình và hiển thị câu: TTDesign.com xin chào <Tên vừa nhập>

2. Viết chương trình nhập vào các thông tin:

- Tên
- Tuổi
- Địa chỉ

Xuất ra màn hình theo định dạng: Bạn tên <Tên>, <Tuổi> tuổi, ở <Địa chỉ>

IV. BIẾN TRONG C#

❖ Biến là gì?

- Là một giá trị dữ liệu có thể thay đổi được.
- Là tên gọi tham chiếu đến một vùng nhớ nào đó trong bộ nhớ.
- Là thành phần cốt lõi của một ngôn ngữ lập trình

➤ Mục đích

- Lưu trữ dữ liệu và tái sử dụng.
- Thao tác với bộ nhớ một cách dễ dàng:
 - 🚦 Cấu trúc của bộ nhớ bao gồm nhiều ô nhớ liên tiếp nhau, mỗi ô nhớ có một địa chỉ riêng (địa chỉ ô nhớ thường mã hex (thập lục phân)).
 - 🚦 Khi muốn sử dụng ô nhớ nào (cấp phát, hủy, lấy giá trị, . . .) bạn phải thông qua địa chỉ của chúng. Điều này làm cho việc lập trình trở nên khó khăn hơn.
 - 🚦 Thay vào đó bạn có thể khai báo một biến và cho nó tham chiếu đến ô nhớ bạn cần quản lý rồi khi sử dụng bạn sẽ dùng tên biến bạn đặt chứ không cần dùng địa chỉ của ô nhớ đó nữa. Rất tiện lợi phải không nào!

❖ Cú pháp:

<Kiểu dữ liệu> <Tên biến>;

➤ Kiểu dữ liệu gồm:

- Kiểu dữ liệu cơ bản.
- Kiểu dữ liệu có cấu trúc (trình bày sau).

➤ Tên biến:

- Là tên do người dùng đặt.
- Phải tuân thủ theo quy tắc đặt tên (sẽ được trình bày ngay sau đây)

➤ Khai báo:

```
int BienKieuSoNguyen;  
float BienKieuSoThuc;  
string BienKieuChuoai;  
bool BienKieuLuanLy;  
char BienKieuKyTu;
```

❖ **Sử dụng biến:**

➤ Để sử dụng biến ta cần phải gán giá trị cho nó trước. Có 2 cách gán giá trị:

- Khởi tạo giá trị lúc khai báo:
- Gán giá trị theo cú pháp: <Tên biến> = <Giá trị>;

➤ Ví dụ:

```
int BienKieuSoNguyen = 10;  
string BienKieuChuoai = "TT Design";  
  
BienKieuSoNguyen = 9;  
BienKieuKyTu = 'TTD';
```

➤ Còn khi bạn muốn gọi một biến ra để lấy giá trị thì bạn chỉ cần gọi tên của nó ra là được.

```
Console.WriteLine(BienKieuSoNguyen);  
// In giá trị của biến tên là BienKieuSoNguyen ra màn hình. Kết quả là 9
```

```
int a = 1, b = 2 ;  
int c = a + b;  
// Biến a và biến b được gọi để lấy giá trị sau đó cộng chúng lại rồi gán cho biến c.
```

❖ **Quy tắc đặt tên biến**

➤ Một số quy tắc khi đặt tên biến cũng như là các định danh khác:

- Tên biến là một chuỗi ký tự liên kết (**không có khoảng trắng**) và **không chứa ký tự đặc biệt**.
- Tên biến không được đặt bằng **tiếng việt có dấu**.
- Tên không được **bắt đầu bằng số**.

- Tên biến không được **trùng nhau**.
- Tên biến không được **trùng với từ khóa**:
- Quy tắc Lạc Đà: Ví dụ: phamHoan, tongKhoiLuong, . . .