Bài: Cấu trúc lệnh cơ bản trong C# Console Application

Xem bài học trên website để ủng hộ Kteam: Cấu trúc lệnh cơ bản trong C# Console Application

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Để viết được một chương trình phần mềm đầu tiên, chúng ta sẽ làm quen với cấu trúc lệnh cơ bản của C#. Cụ thể để đơn giản chúng ta sẽ làm quen với **CẤU TRÚC LỆNH CỦA C# VIẾT TRÊN NỀN CONSOLE APPLICATION**.

Nội dung

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

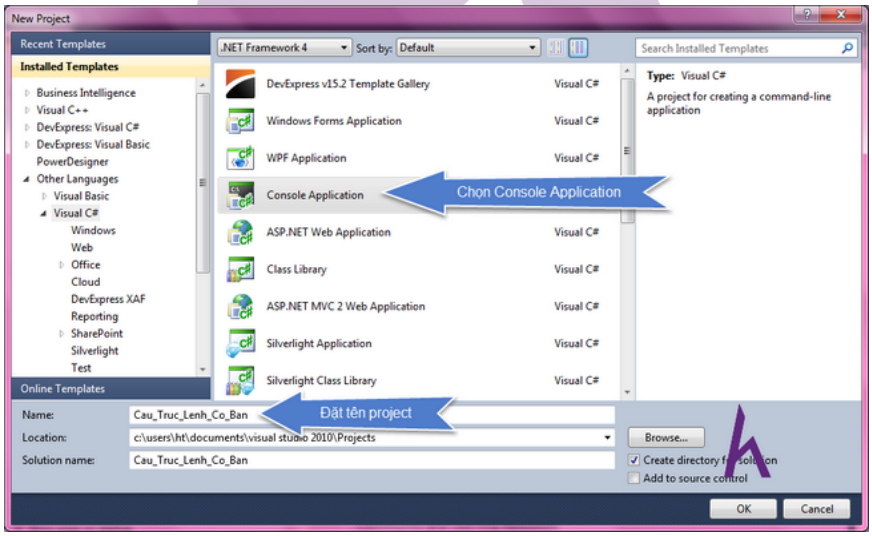
- Cấu trúc cơ bản của một chương trình trong C#.
- Giải thích ý nghĩa một số từ khóa được sử dụng trong chương trình đầu tiên.
- Cách viết comment trong C#.
- Ví dụ chương trình đầu tiên bằng C#.

Các bạn có thể tải và cài đặt visual studio theo bài hướng dẫn CÀI ĐẶT VISUAL STUDIO 2015 hoặc CÀI ĐẶT VISUAL STUDIO 2019

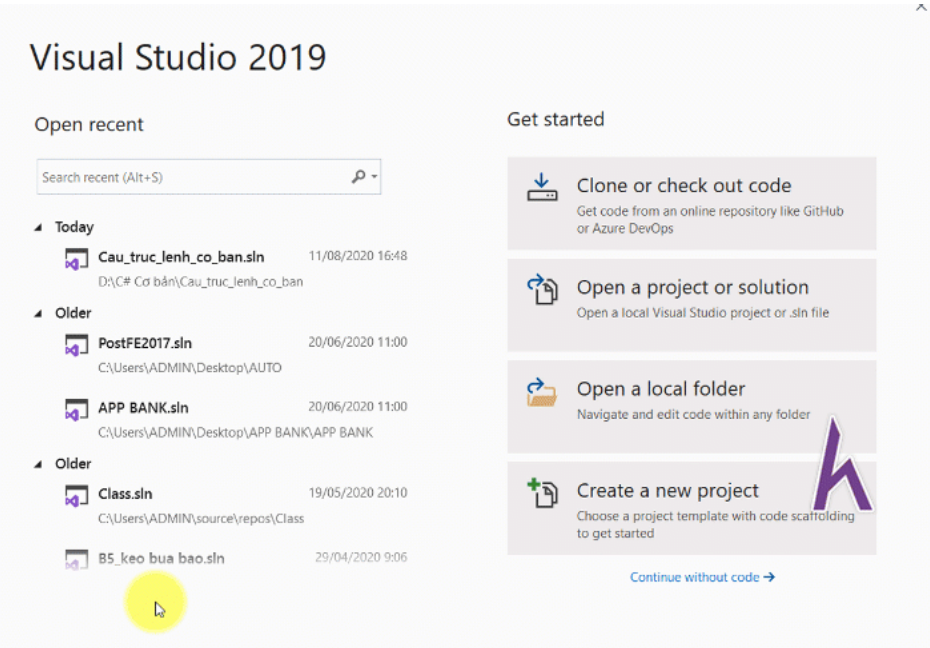
Cấu trúc cơ bản của một chương trình trong C#

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**.



Nếu bạn dùng phiên bản VISUAL STUDIO 2019 sẽ có chút khác biệt trong khởi tạo project như bên dưới:



Sau khi tạo xong project **Cosole Application** thì ta nhận đoạn mã sau:

C#:

```
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)
        {
        }
    }
}
```

Đây chính là cấu trúc cơ bản của một chương trình C# trên nền Console được công cụ hỗ trợ tạo sẵn. Sau đây chúng ta sẽ cùng tìm hiểu từng thành phần của chương trình trên.

Using

Cú pháp:

> **using** <tên thư viện>

Ý nghĩa:

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 viên nào trong chương trình của mình.

Ví dụ: Khai báo một số thư viện

C#:

```
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
}
```

Ý nghĩa: 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 về namespace:

C#:

```
namespace Cau_Truc_Lenh_Co_Ban
{
    public class Action { }

    public delegate void Art();

    namespace Sub_Namespace { }
}
```

Với khai báo trên thì ta thấy các thành phần trong namespace Cau_Truc_Lenh_Co_Ban sẽ thuộc namespace Cau_Truc_Lenh_Co_Ban.

Class

Cú pháp:

```
    class <Tên lớp> { }
```

Ý nghĩa: 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 về lớp:

C#:

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

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

Hàm (Phương thức) Main

Đây là hàm được tạo sẵn khi tạo project với cấu trúc như sau:

```
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

Khi viết code nhu cầu chú thích ý nghĩa đoạn code cũng rất thiết thực.

- 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 sài lại.

Chúng ta hãy cùng tìm hiểu về comment nhé!

C#:

```
using System;

// Commnet không được biên dịch khi dịch chương trình
namespace Cau_Truc_Lenh_Co_Ban
{
    /// <summary>
    /// Comment cho class
    /// </summary>
    class Program
    {
        /// <summary>
        /// Comment cho hàm
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            // Comment cho 1 dòng
            Console.Write("K Team");    // Hoàn toàn có thể comment như thế này.

            Console //à há .WriteLine("Test comment")  cái này lỗi xóa dòng này sẽ chạy được;

            Console.ReadKey(/*haha đoạn comment này không được biên dịch*/);

            /*Comment*/
            /*
             * Hay như thế này
             */

        }   // đoạn code/chữ bạn viết phía sau dấu // sẽ không được biên dịch. Nhưng đoạn code phía trước đó vẫn được biên dịch
bình thường
    }
}
```

Có 3 cách để comment code trong Visual Studio:

Sử dụng ký tự //

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

C#:

```
// Comment cho 1 dòngConsole.Write("K team");    // Hoàn toàn có thể comment như thế này.
```

Nhưng với cách comment như thế này thì sẽ không được. Đoạn code không còn hoàn chỉnh, biến thành comment vì nằm phía sau ký tự //

C#:

```
Console //à há .WriteLine("Test comment");
```

Sau dòng //, dòng tiếp theo sẽ không còn là dòng comment nữa.

C#:

```
// Comment cho 1 dòngConsole.Write("K team");
```

Sử dụng ký tự /**/

Vậy nếu vẫn muốn comment nhưng comment giữa đoạn code. Hay các dòng comment khác sẽ liên tiếp nhau dễ đọc hơn. Thì chúng ta cùng tìm hiểu ký tự comment tiếp theo **/**/**.

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

C#:

```
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:

C#:

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

Bạn cũng có thể comment với ký tự **///** tương tự ký tự **//**. Nhưng có vẻ không được đẹp mắt nhỉ?

Sau này khi tái sử dụng lại bạn sẽ thấy đoạn comment của bạn hiện lên ở phần chú thích của code.

Phím tắt Comment nhanh

Để thao tác nhanh trong lúc code, bạn cần **tô đen** đoạn code cần comment và dùng tổ hợp phím

- **Ctrl + K + C**: đóng comment đoạn code
- **Ctrl + K + U**: để mở đoạn comment

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 **Kteam** khuyến cáo không nên để code rõ ràng.

C#:

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

- Mỗi dòng code là 1 hàng.
- Các đoạn code con thì để trong khối lệnh **{ }**.

Ví dụ chương trình đầu tiên trong C#

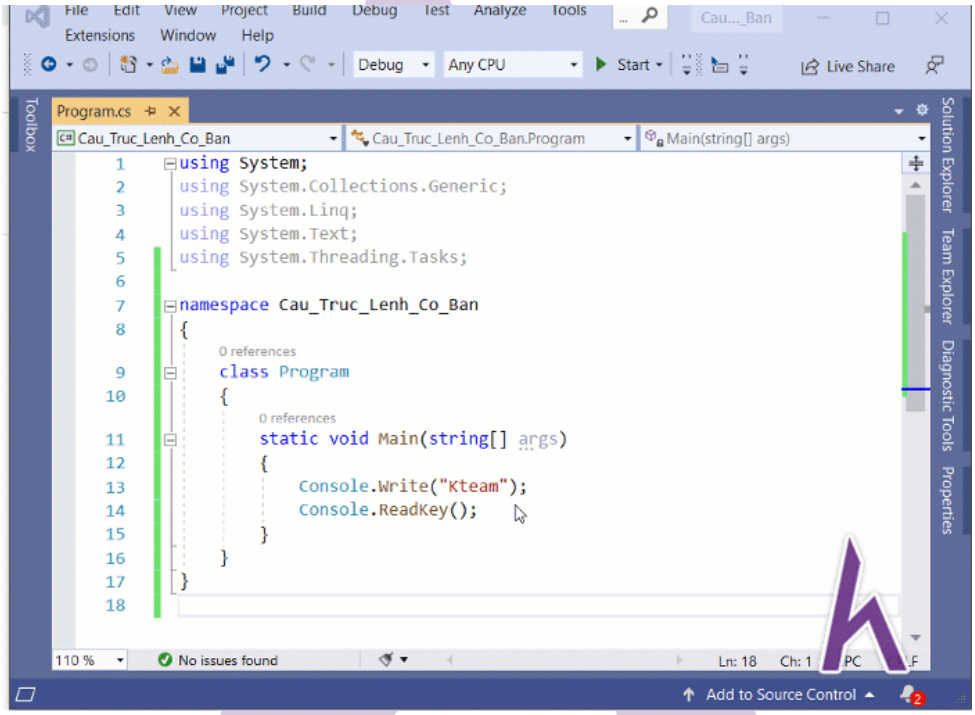
Đầu tiên, các bạn tạo một project mới và nhập đoạn code sau vào:

C#:

```
using System;

namespace Cau_Truc_Lenh_Co_Ban
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Kteam");
            Console.ReadKey();
        }
    }
}
```

- Sau đó nhấn **F7** để biên dịch và nhấn **F5** để chạy.
- Ta được kết quả là **Kteam** trên màn hình Console.



Kết luận

Nội dung bài này giúp các bạn nắm được:

- Qua bài này chúng ta đã nắm được cấu trúc cơ bản của một chương trình C#.
- Giải thích một số từ khóa xuất hiện trong chương trình.
- Viết chương trình đầu tiên và chạy thử.

Bài học sau chúng ta sẽ đi vào chi tiết chương trình đầu tiên cũng như các hàm **NHẬP XUẤT DỮ LIỆU TRONG C#** .

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".