# Bài 5: STACK TRONG C#

*Xem bài học trên website để ủng hộ Kteam: Stack trong C#*

*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

Ở các bài học trước, chúng ta đã cùng nhau tìm hiểu về SORTEDLIST TRONG C#. Hôm nay chúng ta sẽ cùng tìm hiểu về **Stack trong C#**.

---

## Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- BIẾN , KIỂU DỮ LIỆU, TOÁN TỬ trong C#
- CÂU ĐIỀU KIỆN trong C#
- Cấu trúc cơ bản của VÒNG LẶP, HÀM trong C#
- MẢNG trong C#
- LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#

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

- Stack là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong Stack.

---

## Stack là gì?

**Stack** (hay còn gọi là ngăn xếp) là một cấu trúc dữ liệu hoạt động theo nguyên lý **LIFO** (**L**ast **I**n **F**irst **O**ut). Người ta hay gọi nó là ngăn xếp bởi vì nó hoạt động giống như ngăn xếp trong thực tế vậy.

Giả sử bạn xếp chồng các chiếc đĩa lên nhau như hình dưới:


Khi đó chiếc đĩa được xếp đầu tiên sẽ nằm dưới cùng và chiếc đĩa được xếp sau cùng sẽ nằm trên đầu. Nếu bạn muốn lấy đĩa ra sử dụng chắc chắn bạn sẽ lấy chiếc đĩa nằm trên cùng ra. Đây chính là nguyên lý **Last In First Out** (vào cuối ra đầu) của Stack.

Quay lại vấn đề lập trình, bạn tưởng tượng cả chồng đĩa đó chính là 1 danh sách (Stack), các chiếc đĩa là các phần tử của danh sách. Thế là ta đã có cấu trúc dữ liệu Stack trong lập trình rồi đó.

Trong C#, Stack là một Collections đại diện cho một danh sách hoạt động theo nguyên lý **LIFO** đã trình bày ở trên.

Vì C# đã hỗ trợ sẵn cấu trúc dữ liệu Stack rồi nên chúng ta chỉ tìm hiểu cách sử dụng nó thôi. Còn cách tổ chức nó như thế nào sẽ được trình bày trong serial về cấu trúc dữ liệu và giải thuật.

---

# Đặc điểm của Stack

- Là một danh sách lưu trữ các đối tượng nhưng không thể truy cập các phần tử thông qua chỉ số phần tử được.
- Hành động thêm phần tử vào Stack được gọi là **Push** (đẩy vào).
- Hành động lấy phần tử ra khỏi Stack được gọi là **Pop** (đẩy ra). Và luôn luôn lấy ra phần tử được thêm vào cuối cùng.

Do Stack cũng là 1 Collections nên để sử dụng ta cần thêm thư viện System.Collections bằng câu lệnh:

```
using System.Collections;
```

Trước khi sử dụng ta cần khởi tạo vùng nhớ bằng toán tử new:

```
Stack MyStack = new Stack(); // khởi tạo 1 Stack rỗng
```

Bạn cũng có chỉ định sức chứa (Capacity) ngay lúc khởi tạo bằng cách thông qua **constructor** được hỗ trợ sẵn:

```
Stack MyStack = new Stack(5); // khởi tạo 1 Stack và chỉ định sức chứa ban đầu là 5
```

Bạn cũng có thể khởi tạo 1 **Stack** chứa các phần tử được sao chép từ một danh sách khác:

```
Stack MyStack = new Stack(); // khởi tạo 1 Stack rỗng
Bạn cũng có chỉ định sức chứa (Capacity) ngay lúc khởi tạo bằng cách thông qua constructor được hỗ trợ sẵn:
Stack MyStack = new Stack(5); // khởi tạo 1 Stack và chỉ định sức chứa ban đầu là 5
Bạn cũng có thể khởi tạo 1 Stack chứa các phần tử được sao chép từ một danh sách khác:
// khởi tạo 1 mảng bất kỳ
ArrayList MyArray = new ArrayList();
MyArray.Add(5);
MyArray.Add(9);
MyArray.Add(10);

// Khởi tạo 1 Stack và sao chép giá trị của các phần tử từ MyArray vào Stack.
Stack MyStack3 = new Stack(MyArray);
```

# Một số thuộc tính và phương thức hỗ trợ sẵn trong Stack

Một số thuộc tính thông dụng trong Stack:

| TÊN THUỘC TÍNH | Ý NGHĨA |
|---|---|
| Count | Trả về 1 số nguyên là **số phần tử hiện có** trong Stack. |

Một số phương thức thông dụng trong Stack:

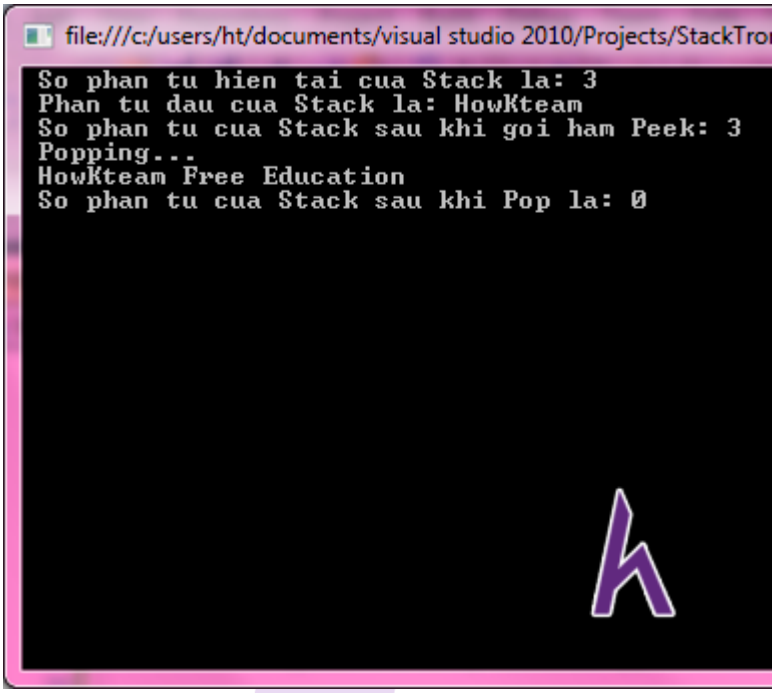
| TÊN PHƯƠNG THỨC | Ý NGHĨA |
|---|---|
| Clear() | Xoá tất cả các phần tử trong Stack. |
| Clone() | Tạo 1 bản sao từ Stack hiện tại. |
| Contains (object Value) | Kiểm tra đối tượng Value có tồn tại trong Stack hay không. |
| CopyTo(Array array, int Index) | Thực hiện sao chép tất cả phần tử trong Stack sang mảng một chiều array từ vị trí Index của array. |
| Peek() | Trả về giá trị của đối tượng tại vị trí trên cùng trong Stack (phần tử được thêm vào cuối cùng) nhưng không xoá phần tử khỏi Stack. |
| Pop() | Trả về giá trị của đối tượng tại vị trí trên cùng trong Stack (phần tử được thêm vào cuối cùng) đồng thời xoá phần tử khỏi Stack. |
| Push(object Value) | Thêm một phần tử có giá trị Value vào vị trí trên cùng trong Stack. |
| ToArray() | Tạo ra 1 mảng các object chứa tất cả các phần tử trong Stack và trả về mảng đó. |

# Một ví dụ đơn giản về sử dụng Stack

```
// Tạo 1 Stack rỗng
Stack MyStack4 = new Stack();
```

```
// Thực hiện thêm vài phần tử vào Stack thông qua hàm Push.
MyStack4.Push("Education");
MyStack4.Push("Free");
MyStack4.Push("HowKteam");

// Thử sử dụng các phương thức của Stack.
Console.WriteLine(" So phan tu hien tai cua Stack la: {0}", MyStack4.Count);

// Lưu ý ở đây ta chỉ muốn xem giá trị mà không muốn nó khỏi Stack thì ta sẽ dùng Peek.
Console.WriteLine(" Phan tu dau cua Stack la: {0}", MyStack4.Peek());

// Thử kiểm tra lại số phần tử để chắc chắn rằng hàm Peek không xoá phần tử ra khỏi Stack.
Console.WriteLine(" So phan tu cua Stack sau khi goi ham Peek: {0}", MyStack4.Count);

// Thực hiện xoá các phần tử ra khỏi Stack.
Console.WriteLine(" Popping...");
int Length = MyStack4.Count;
for (int i = 0; i < Length; i++)
{
    Console.Write(" " + MyStack4.Pop());
}
Console.WriteLine();

// Kiểm tra lại số phần tử của Stack sau khi Pop
Console.WriteLine(" So phan tu cua Stack sau khi Pop la: {0}", MyStack4.Count);
```

Kết quả khi chạy đoạn chương trình trên:




# Kết luận

Nội dung bài này giúp các bạn nắm được:

- Stack là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong Stack.

Bài học sau chúng ta sẽ cùng tìm hiểu về QUEUE 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**".