

Bài: List trong C#

Xem bài học trên website để ủng hộ Kteam: [List 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 [GENERIC TRONG C#](#). Hôm nay chúng ta sẽ cùng tìm hiểu về **List 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:

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

List là gì?

List là 1 [Generic Collections](#) đưa ra như một sự thay thế [ArrayList](#) vì thế về khái niệm cũng như sử dụng nó hoàn toàn giống với [ArrayList](#). (bạn có thể tham khảo chi tiết trong bài [ARRAYLIST TRONG C#](#))

Ở đây mình chỉ trình bày lại một số ý để những bạn nào không theo dõi những bài trước vẫn có thể hiểu được.

List trong C# là một [Generic Collections](#) giúp lưu trữ và quản lý một danh sách các đối tượng theo kiểu mảng (truy cập các phần tử bên trong thông qua chỉ số **index**).

Để sử dụng các Collections trong .NET ta cần thêm thư viện [System.Collections.Generic](#) bằng câu lệnh:

```
using System.Collections.Generic;
```

Vì List là một lớp nên trước khi sử dụng ta cần khởi tạo vùng nhớ bằng toán tử **new**:

C#:

```
// khởi tạo 1 List các số nguyên rỗng  
List<int> MyList = new List<int>();
```

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

C#:

```
// khởi tạo 1 List các số nguyên và chỉ định Capacity ban đầu là 5  
List<int> MyList2 = new List<int>(5);
```

Ngoài ra bạn cũng có thể khởi tạo 1 List chứa các phần tử được sao chép từ một [Generic Collections](#) khác (lưu ý là có cùng kiểu dữ liệu truyền vào):

C#:

```
/*
 * Khởi tạo 1 List số nguyên có kích thước bằng với MyList2.
 * Sao chép toàn bộ phần tử trong MyList2 vào MyList3.
 */
List<int> MyList3 = new List<int>(MyList2);
```

Một số thuộc tính và phương thức hỗ trợ sẵn trong List

Một số thuộc tính thông dụng trong List:

TÊN THUỘC TÍNH	Ý NGHĨA
Count	Trả về 1 số nguyên là số phần tử hiện có trong List .
Capacity	Trả về 1 số nguyên cho biết số phần tử mà List có thể chứa (sức chứa). Nếu số phần tử được thêm vào chạm sức chứa này thì hệ thống sẽ tự động tăng lên. Ngoài ra ta có thể gán 1 sức chứa bất kỳ cho List .

Một số phương thức thông dụng trong List:

TÊN PHƯƠNG THỨC	Ý NGHĨA
Add(object Value)	Thêm đối tượng Value vào cuối List .
AddRange(ICollection ListObject)	Thêm danh sách phần tử ListObject vào cuối List .
BinarySearch(object Value)	Tìm kiếm đối tượng Value trong List theo thuật toán tìm kiếm nhị phân. Nếu tìm thấy sẽ trả về vị trí của phần tử ngược lại trả về giá trị âm. Lưu ý: List phải được sắp xếp trước khi sử dụng hàm.
Clear()	Xoá tất cả các phần tử trong List .
Contains(T Value)	Kiểm tra đối tượng Value có tồn tại trong List hay không.
CopyTo(T[] array, int Index)	Thực hiện sao chép tất cả phần tử trong List sang mảng một chiều array từ vị trí Index của array. Lưu ý: array phải là mảng kiểu T tương ứng.
IndexOf(T Value)	Trả về vị trí đầu tiên xuất hiện đối tượng Value trong List . Nếu không tìm thấy sẽ trả về -1 .
Insert(int Index, T Value)	Chèn đối tượng Value vào vị trí Index trong List .
InsertRange(int Index, IEnumerable<T> ListObject)	Chèn danh sách phần tử ListObject vào vị trí Index trong List .
LastIndexOf(T Value)	Trả về vị trí xuất hiện cuối cùng của đối tượng Value trong List . Nếu không tìm thấy sẽ trả về -1 .

Remove(T Value)	Xoá đối tượng Value xuất hiện đầu tiên trong List .
Reverse()	Đảo ngược tất cả phần tử trong List .
Sort()	Sắp xếp các phần tử trong List theo thứ tự tăng dần.
ToArray()	Trả về 1 mảng kiểu T chứa các phần tử được sao chép từ List .

Sử dụng [List](#) hoàn toàn tương tự như sử dụng [ArrayList](#). Một ví dụ đơn giản về sử dụng List:

C#:

```
/*
 * Tạo 1 List các kiểu string và thêm 2 phần tử vào List.
 */
List<string> MyList4 = new List<string>();
MyList4.Add("Free");
MyList4.Add("Education");

// In giá trị các phần tử trong List
Console.WriteLine(" List ban dau: ");
Console.WriteLine(" So luong phan tu trong List la: {0}", MyList4.Count);
foreach (string item in MyList4)
{
    Console.Write(" " + item);
}
Console.WriteLine();

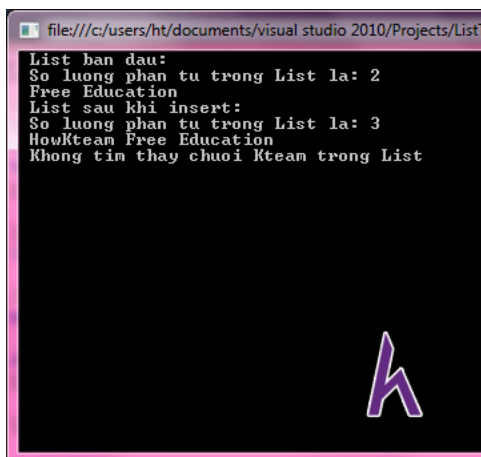
// Chèn 1 phần tử vào đầu List.
MyList4.Insert(0, "HowKteam");

// In lại giá trị các phần tử trong List để xem đã chèn được hay chưa
Console.WriteLine(" List sau khi insert: ");
Console.WriteLine(" So luong phan tu trong List la: {0}", MyList4.Count);
foreach (string item in MyList4)
{
    Console.Write(" " + item);
}
Console.WriteLine();

// Kiểm tra 1 phần tử có tồn tại trong List hay không.
bool isExists = MyList4.Contains("Kteam");

if (isExists == false)
{
    Console.WriteLine(" Không tìm thấy chuỗi Kteam trong List");
}
```

Kết quả: khi chạy đoạn chương trình trên là:



```
file:///c:/users/ht/documents/visual studio 2010/Projects/List/
List ban dau:
So luong phan tu trong List la: 2
Free Education
List sau khi insert:
So luong phan tu trong List la: 3
Howkteam Free Education
Khong tim thay chuoi Kteam trong List
```

Kết luận

Nội dung bài này giúp các bạn nắm được:

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

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