1. **Chuyển đổi vòng lặp sang một LINQ query.**

Một vòng lặp có 3 thành phần là phần khởi tạo, điều kiện dừng và bước nhảy của vòng lặp, để chuyển đổi một vòng lặp sang một câu truy vấn LINQ bạn phải xác định 3 thành phần sau:

1. Xác định dãy giá trị của vòng lặp (ví dụ một mảng các phần tử thì chính là số lượng phần tử của mảng đó đang chứa).
2. Xác định điều kiện dừng của vòng lặp.
3. Tìm một toán tử phù hợp để thay thế cho điều kiện dừng đó.

Ví dụ: ta có một vòng lặp sau:

var numbers = new int[] { 1, 2, 3, 4, 6, 7, 8, 9, 10 };

// To assume new array the capacity is 100 items.

var goodNumber = new List<int>();

for (int i = 0; i < numbers.Length; i++)

{

if (numbers[i] % 2 == 0)

goodNumber.Add(numbers[i]);

}

Từ vòng lặp bên trên ta có thể dễ dàng chuyển đổi về vòng lặp foreach trong C# nhằm bỏ đi các biến tạm được dùng trong vòng lặp.

foreach (var item in numbers)

{

if (item % 2 == 0)

goodNumber.Add(item);

}

Đến lúc này ta có thể từ vòng lặp foreach trên chuyển về một câu query trong LINQ như sau:

goodNumber = Enumerable.Range(0, numbers.Length).Where(n => n % 2 == 0).ToList();

1. **Sử dụng toán tử Any trong LINQ.**

***Toán tử Any sẽ trả về treu nếu có ít nhất một giá trị trong một tập hợp mà đáp ứng được điều kiện mà ta chỉ ra.***

Trong đoạn code bên dưới giúp ta kiểm tra xem có bất kỳ phần tử nào trong một danh sách lớn hơn hoặc bằng 150.

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

bool isAny = false;

for (int i = 0; i < nums.Length; i++)

{

if (nums[i] >= 150)

{

isAny = true;

break;

}

}

Bằng sách sử dụng toán tử Any của LINQ ta sẽ có đoạn code như sau

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

bool isAny = nums.Any(n => n >= 150);

1. **Toán tử All trong LINQ.**

***Bạn sử dụng toán tử All khi bạn muốn kiểm tra liệu rằng tất cả các phần tử trong một tập hợp có đáp ứng một điều kiện của bạn.***

Trong đoạn code bên dưới, bạn kiểm tra xem liệu rằng tất cả các phần tử của mảng nums có nhỏ hơn 150 hay hoặc bằng 150 hay không.

// THE ALL OPERATOR.

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

bool isAll = true;

for (int i = 0; i < nums.Length; i++)

{

if (nums[i] < 150)

{

isAll = false;

break;

}

}

Bằng cách sử dụng toán tử All trong LINQ ta sẽ có được kết quả tương tự như sau:

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

bool isAll = nums.All(n => n < 150);

1. **Toán tử Take trong LINQ.**

***Toán tử này sẽ lấy ra một tập phần tử đầu, số lượng phần tử lấy ra sẽ do người dùng truyền vào.***

Trong đoạn code bên dưới, việc của ta là tử một mảng cho trước lấy ra 4 phần tử đầu tiên của mảng này.

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

int[] first4 = new int[4];

for (int i = 0; i < 4; i++)

{

first4[i] = nums[i];

}

Bằng cách sử dụng toán tử Take trong LINQ ta sẽ có được kết quả tương tự như sau:

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

int[] first4 = new int[4];

first4 = nums.Take(4).ToArray();

1. **Toán tử Skip trong LINQ.**

***Toán tử này sẽ lấy ra tất cả phần tử nhưng sẽ bỏ qua K phần tử đầu, với K là giá trị bạn truyền vào.***

Trong đoạn code bên dưới, từ một tập các phần tử cho trước, ta muốn lấy ra tất cả phần tử, nhưng bỏ qua 4 phần tử đầu.

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

int[] skip4 = new

int[nums.Length - 4];

for (int i = 4, j = 0; i < nums.Length; i++, j++)

{

skip4[j] = nums[i];

}

Bằng cách sử dụng toán tử Skip trong LINQ ta có kết quả như sau:

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

int[] skip4 = new int[nums.Length - 4];

skip4 = nums.Skip(4).ToArray();

1. **Toán tử TakeWhile.**

***Toán tử này cho phép bạn lấy ra những phần tử của một tập hợp với một điều kiện cho trước, khi gặp đến điều kiện nó sẽ dừng ngay lập tức.***

Đoạn code bên dưới dùng để lấy ra các phần tử đùa tiên mà nhỏ hơn 50.

Bằng cách sử dụng toán tử TakeWhile trong LINQ ta viết lại như sau:

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

List<int> until50 = new

List<int>();

for (int i = 0; i < nums.Length; i++)

{

if (nums[i] < 50)

{

until50.Add(nums[i]);

}

else

break;

}

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

List<int> until50 = new List<int>();

until50 = nums.TakeWhile(n => n < 50).ToList();

1. **Toán tử SkipWhile trong LINQ.**

***Toán tử này sẽ bỏ qua các phần tử bắt đầu danh sách tới phần tử làm cho điều kiện mà bạn truyền vào bị false (gặp phần tử làm cho điều kiện false đầu tiên sẽ dừng lại).***

Ta có đoạn chương trình bên dưới nhằm lấy tra các phần tử mà không chia hết cho 7.

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

List<int> skipWhileDivisibleBy7 = new

List<int>();

for (int i = 0; i < nums.Length; i++)

{

if (nums[i] % 7 == 0)

{

continue;

}

else

skipWhileDivisibleBy7.Add(nums[i]);

}

Bằng cách sử dụng SkipWhile ta được như sau:

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

List<int> skipWhileDivisibleBy7 =

nums.SkipWhile(n => n % 7 == 0).ToList();

Ta lưu ý rằng ở đoạn code LINQ bên trên khi gặp phần tử 24 trong array sẽ làm điều kiện sai và ngay lập tức toán tử SkipWhile sẽ dừng lại và lấy các giá trị từ vị trí phần tử đó đến hết mảng. Điều này sẽ sai nếu có phần tử nào chia hết cho 7 nằm sau phần tử làm điều kiện của SkipWhile bị sai.

1. Toán tử where.

***Toán tử where giúp ta tìm kiếm các phần tử mà thỏa mãn điều kiện mà ta truyền vào. Đây là một toán tử được sử dụng nhiều và quan trọng trong LINQ.***

Ở đoạn code bên dưới ta muốn rằng tạo ra một array với các phần tử lớn hown 50 từ một array ban đầu. Ta thấy rằng ta dùng một biến đếm J để đếm các phần tử lớn hơn 50, và khi kết thúc việc tìm kiếm ta sẽ resize lại mảng kết quả cho đúng với số phần tử (mặc định ban đầu ta đã lấy size của nó chính la size của mảng gốc.

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

int[] above50 = new int[nums.Length];

int j = 0;

for (int i = 0; i < nums.Length; i++)

{

if (nums[i] > 50)

{

above50[j] = nums[i];

j++;

}

}

Array.Resize(ref above50, j);

Bằng cách sử dụng toán tử where trong LINQ ta được như sau:

int[] nums = { 14, 21, 24, 51, 131, 1, 11, 54 };

int[]above50 = nums.Where(n => n > 50).ToArray();

1. Toán tử zip.

Toán tử này cho phép ta thiết đặt một function đến tương ứng của từng phần tử của 2 tập hợp để tạo ra kết quả tương ứng.

Ví dụ: Ta muốn in ra tên đầy đủ của tất cả các thành viên trong gia đình bao gồm tên, họ và title đích danh của họ thì ta làm như sau:

string[] salutations = {"Mr.", "Mrs.","Master.","Ms."};

string[] names = { "Patrick", "Nancy", "Jon", "Jane" };

List<string> allNames = new List<string>();

for (int i = 0; i < salutations.Length; i++)

allNames.Add(salutations[i] + " " +

names[i] + " Smith");

foreach (var item in allNames)

{

Console.WriteLine(item);

Bằng cách sử dụng toán tử zip của LINQ ta sẽ có đoạn chương trình như sau:

string[] salutations = { "Mr.", "Mrs.", "Master.", "Ms." };

string[] names = { "Patrick", "Nancy", "Jon", "Jane" };

List<string> allNames = new List<string>();

allNames = salutations.Zip(

names,

(salutation, name) => salutation + " " + name + " Smith"

).ToList();

foreach (var item in allNames)

{

Console.WriteLine(item);

}

1. OrderBy and OrderByDescending Operators

Hai toán tử này dùng để sắp xếp các giá trị trong một tập hợp tăng dần hoặc giảm dần.

public class StringLengthComparer :IComparer<string>

{

public int Compare(string x, string y)

{

return x.Length.CompareTo(y.Length);

}

}

void Main()

{

string[] codes =

{"abc","bc","a","d","abcd"};

StringLengthComparer slc = new

StringLengthComparer();

List<string> codesAsList = codes.ToList();

codesAsList.Sort(slc);

}

string[] codes = { "abc", "bc", "a", "d", "abcd" };

List<string> codesAsList = codes

.OrderBy(item => item.Length).ToList();

1. The Distinct Operator

Toán tử này dùng để lấy ra các giá trị duy nhất và lọc các giá trị bị trùng lặp trong một tập hợp.

Việc sử dụng vòng lặp để loại bỏ giá trị trùng.

string[] names = {"Sam","David","Sam","Eric", "Daniel","Sam"};

Array.Sort(names);

List<string> distinctNames = new List<string>();

for (int i = 0; i < names.Length - 1; i++)

{

if (names[i] != names[i + 1])

distinctNames.Add(names[i]);

else

{

if (distinctNames[distinctNames.Count -

1] != names[i])

distinctNames.Add(names[i]);

}

}

Bằng cách sử dụng toán tử Distinct trong LINQ

string[] names = {"Sam","David","Sam","Eric", "Daniel","Sam"};

List<string> distinctNames = names.Distinct().ToList();