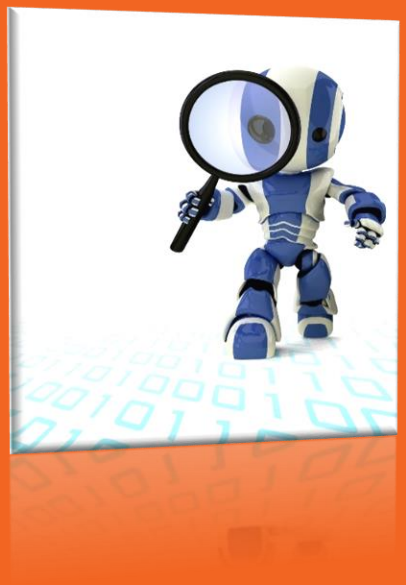
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 5: LINQ AGGREGATE FUNCTIONS & OPERATORS

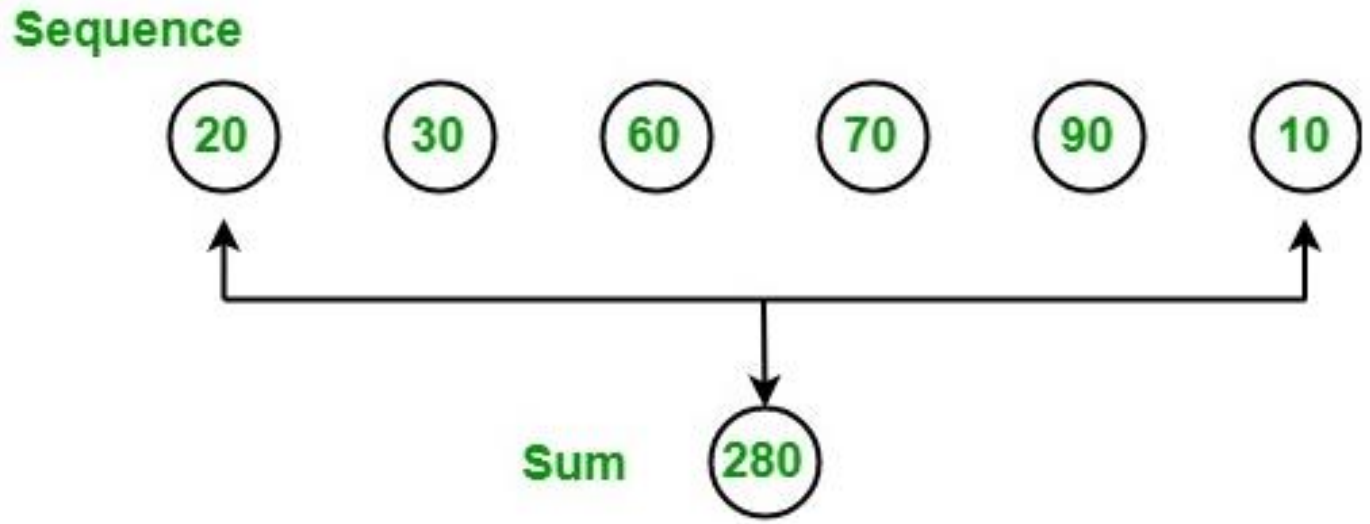
www.poly.edu.vn

Mục tiêu

- LINQ Aggregate Functions
- LINQ Operators

AGGRETGATE FUNCTION

- Aggretgate Function được dùng trong các trường hợp tính toán một giá trị tổng hợp từ một tập các giá trị



- Một số phương thức thông dụng: Min, Max, Sum, Count, Aggregate…

Ví dụ LINQ Min() Function

```
class Program
{
static void Main(string[] args)
{
int[] Num = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
Console.WriteLine("The Minimum value in the given array is:");
int minimumNum = Num.Min();
Console.WriteLine("The minimum Number is {0}", minimumNum);
Console.ReadLine();
}
}
```

Ví dụ LINQ Sum() Function

```
class Program
{
    static void Main(string[] args)
    {
//create array num with the initializing value
        int[] Num = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
        Console.WriteLine("Calculating the sum of all the elements of the array :");
//Num.Sum() is used to add the value of the Num array
        int Sum = Num.Sum();
        Console.WriteLine("The Sum is {0}", Sum);
        Console.ReadLine();
    }
}
```

Ví dụ LINQ Aggregate() Function

```
classProgram
{
staticvoid Main(string[] args)
{
int[] Num = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
Console.WriteLine("Find the Product of the elements:");
double Average = Num.Aggregate((a, b) => a * b);
Console.WriteLine("The Product is {0}", Average); //Output 362880 (((((((1*2)*3)*4)*5)*
6)*7)*8)*9)
string[] charlist = { "a", "b", "c", "d" };
var concta = charlist.Aggregate((a, b) => a + ',' + b);
Console.WriteLine("Concatenated String: {0}",concta); // Output a,b,c,d
Console.ReadLine();
}
}
```

DEMO

- Hiện thực các ví dụ

SORTING OPERATORS IN LINQ

- Sorting Operators in LINQ dùng thay đổi thứ tự tăng dần hoặc giảm dần các các phần tử trong tập hợp theo một hoặc nhiều tiêu chí

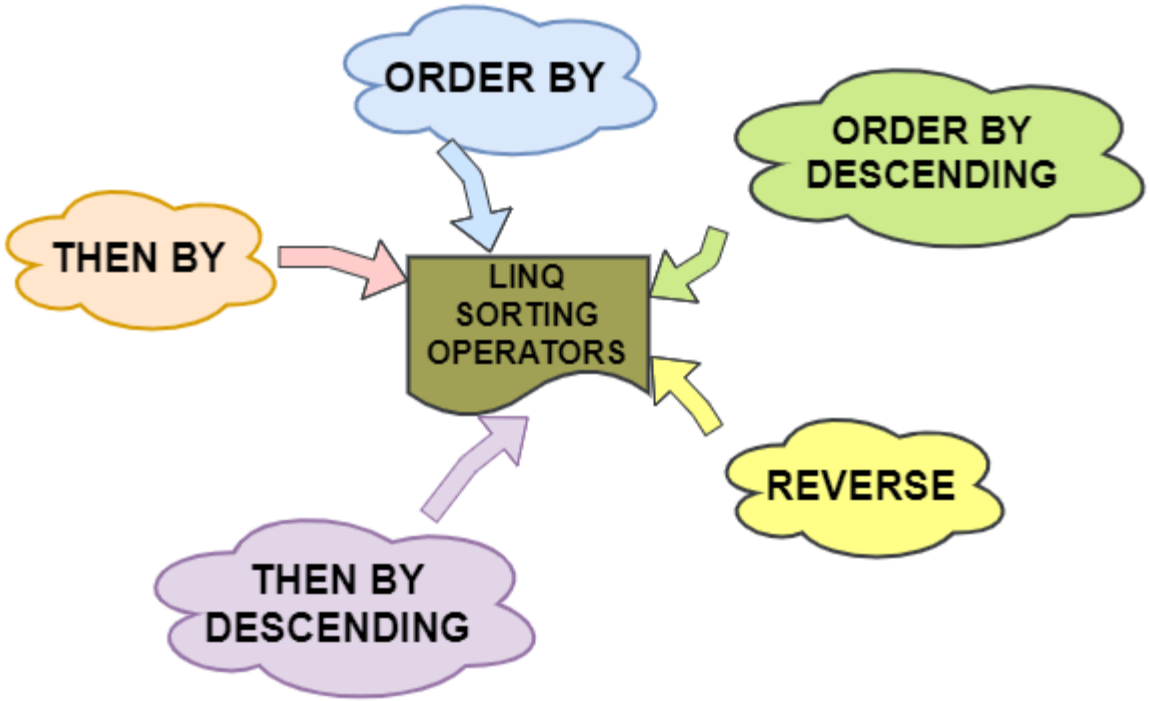


FIG: LINQ SORTING OPERATORS

SORTING OPERATORS IN LINQ

- LINQ OrderBy Operator: sắp xếp dữ liệu trong danh sách hoặc tập hợp

```
static void Main(string[] args)
{
   List<Student> Objstudent = new List<Student>(){
new Student() { Name = "Suresh Dasari", Gender = "Male", Subjects = new List<string> { "Mathematics", "Physics" } }
new Student() { Name = "Rohini Alavala", Gender = "Female", Subjects = new List<string> { "Entomology", "Botany" }
new Student() { Name = "Praveen Kumar", Gender = "Male", Subjects = new List<string> { "Computers", "Operating Sy
new Student() { Name = "Sateesh Chandra", Gender = "Male", Subjects = new List<string> { "English", "Social Studies'
new Student() { Name = "Madhav Sai", Gender = "Male", Subjects = new List<string> { "Accounting", "Charted" } }
};
   var studentname = Objstudent.OrderBy(x => x.Name);
   foreach (var student in student name)
   {
      Console.WriteLine(student.Name);
   }
      Console.ReadLine();
```

```
class Student
{
   public int RoleId { get; set; }
   public string Name { get; set; }
   public string Gender { get; set; }
   public List<string> Subjects { get; set; }
}
```

SORTING OPERATORS IN LINQ

- LINQ OrderBy Operator: sắp xếp dữ liệu trong danh sách hoặc tập hợp

```
static void Main(string[] args)
{
//create object of Student class and create a list of the student information
        List<Student> Objstudent = new List<Student>()
        {
            new Student() { Name = "Akshay", Gender = "Male", Subjects = new List<string> { "Mathematics", "Physics" } },
            new Student() { Name = "Vaishali", Gender = "Female", Subjects = new List<string> { "Computer", "Botany" } },
            new Student() { Name = "Arpita", Gender = "FMale", Subjects = new List<string> { "Economics", "Operating Syste
            new Student() { Name = "Shubham", Gender = "Male", Subjects = new List<string> { "Account", "Social Studies",
            new Student() { Name = "Himanshu", Gender = "Male", Subjects = new List<string>{ "English", "Charted" } }
    };
    /*OrderByDescending() operator is used to print
    the name of the student in the descending form*/
    var studentname = Objstudent.OrderByDescending(x => x.Name);
    //foreach loop is used to print the name of the student
    foreach (var student in studentname)
    {
    Console.WriteLine(student.Name);
    }
    Console.ReadLine();
```

SORTING OPERATORS IN LINQ

- LINQ OrderBy viết theo cách LINQ queries

```
string[] words = { "cherry", "apple", "blueberry" };
var wordsSortedByLength =
    from word in words
    orderby word.Length descending
    select word;
foreach (var word in wordsSortedByLength)
{
    Console.WriteLine(word);
}
```

The result is:

```
blueberry
cherry
apple
```

SORTING OPERATORS IN LINQ

- LINQ ThenBy Operator: sắp xếp dữ liệu trong danh sách hoặc tập hợp theo nhiều tiêu chí

```
//create an object Objstudent of the class Student, and create a list of the information of the student
        List<Student> Objstudent = new List<Student>()
        {
            new Student() { RoleId=1, Name = "Ak", Gender = "Male", Subjects = new List<string> { "Mathematics", "Physics
            new Student() { RoleId=2, Name = "Shalu", Gender = "Female", Subjects = new List<string> { "Computers", "Bot
            new Student() { RoleId=3, Name = "Shubham", Gender = "Male", Subjects = new List<string> { "Economics", "Op
            new Student() { RoleId=4, Name = "Rohit", Gender = "Male", Subjects = new List<string> { "Accounting", "Social
            new Student() { RoleId=5, Name = "Shivani", Gender = "FeMale", Subjects = new List<string> { "English", "Chart
        };
    //ThenBy() operator is used here to sort the Information of the student in ascending form on the behalf of the RollNumber
            var studentname = Objstudent.OrderBy(x => x.Name).ThenBy(x => x.RoleNumber Id);
    //foreach loop is used to print the information
            foreach (var student in studentname)
            {
                Console.WriteLine("Name={0} studentid={1}", student.Name, student.Roleid);
            }
                Console.ReadLine();
```

SORTING OPERATORS IN LINQ

- LINQ ThenBy Operator: sắp xếp dữ liệu trong danh sách hoặc tập hợp theo nhiều tiêu chí

```
//Create object ObjStudent of the Student class having the list of the student information
        List<Student> Objstudent = new List<Student>()
        {
            new Student() { RoleId=1, Name = "Suresh Dasari", Gender = "Male", Subjects = new List<string> { "Mathematic
            new Student() { RoleId=2, Name = "Rohini Alavala", Gender = "Female", Subjects = new List<string> { "Entomolo
            new Student() { RoleId=3, Name = "Praveen Kumar", Gender = "Male", Subjects = new List<string> { "Computers
            new Student() { RoleId=4, Name = "Sateesh Chandra", Gender = "Male", Subjects = new List<string> { "English",
            new Student() { RoleId=5, Name = "Madhav Sai", Gender = "Male", Subjects = new List<string> { "Accounting", "
        };
    //ThenByDescending() operator is used to sort the information of the student in the descending form
            var studentname = Objstudent.OrderBy(x => x.Name).ThenByDescending(x => x.RoleId);
            foreach (var student in studentname)
            {
                Console.WriteLine("Name={0} StudentId={1}", student.Name, student.RoleId);
            }
                Console.ReadLine();
```

PARTITION OPERATORS

- Partition Operators dùng phân chia danh sách, tập hợp thành các phần và trả về một phần dữ liệu

PARTITION OPERATORS

- Take Partition Operator: Toán tử này sẽ lấy N phần tử đầu tiên trong một tập dữ liệu

```
namespace LINQExamples
{
class Program
{
static void Main(string[] args)
{
string[] countries = { "India", "USA", "Russia", "China", "Australia", "Argentina" };
IEnumerable<string> result = countries.Take(3);
foreach (string s in result)
{
Console.WriteLine(s);
}
Console.ReadLine();
}
}
}
```

PARTITION OPERATORS

- LINQ TakeWhile Partition Operator: chỉ lấy các phần tử thõa mãn một điều kiện nhất định nào đó trong một dãy

```
class Program
{
    static void Main(string[] args)
    {
//Array countries is created of string type.
        string[] countries = { "US", "UK", "Russia", "China", "Australia", "Argentina" };
/*TakeWhile operator is used which will print
the values until the specified condition is satisfied.*/
        IEnumerable<string> result = countries.TakeWhile(x => x.StartsWith("U"));
  //foreach loop will print the value of the result
        foreach (string s in  result)
        {
            Console.WriteLine(s);
        }
            Console.ReadLine();
    }
}
```

PARTITION OPERATORS

- LINQ Skip Operator: bỏ đi các phần tử được chỉ định

```
class Program
{
   static void Main(string[] args)
   {
//create array of string type countries with the initialization
        string[] countries = { "US", "UK", "India", "Russia", "China", "Australia", "Argentina" };
/*skip method is used to with the IEnumerable to return
 the value which skip the third element of the array*/
        IEnumerable<string> result = countries.Skip(3);
  //foreach loop is used to print the element of the array
        foreach (string s in result)
        {
           Console.WriteLine(s);
        }
           Console.ReadLine();
   }
}
```

PARTITION OPERATORS

- LINQ SkipWhile Operator: bỏ đi các phần tử được chỉ định có điều kiện

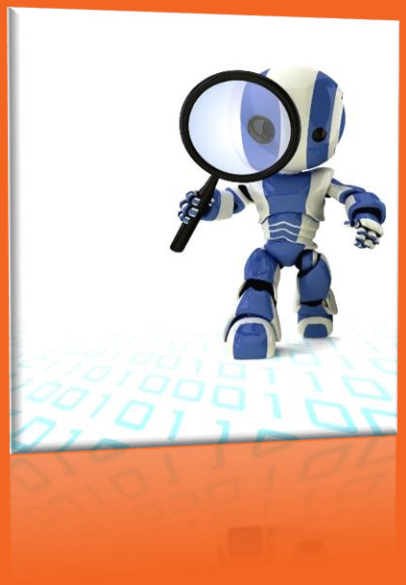
```
classProgram
{
staticvoid Main(string[] args)
{
string[] countries = { "US", "UK","India", "Russia", "China", "Australia", "Argentina"
};
IEnumerable<string> result = countries.SkipWhile(x => x.StartsWith("U"));
foreach (string s in result)
{
Console.WriteLine(s);
}
Console.ReadLine();
}
}
```

DEMO

- Hiện thực các ví dụ

FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 5: LINQ AGGREGATE FUNCTIONS & OPERATORS (P2)

www.poly.edu.vn

CONVERSION OPERATORS

- Sử dụng các operators này để ép kiểu các phần tử trong danh sách hoặc tập hợp

Operator	Description
ToList	It converts a collection to List
ToArray	It converts a collection to an array
ToLookUp	It converts group of elements into Lookup<tkey, telement>
Cast	It converts non-generic list to generic list (IEnumerable to IEnumerable)
OfType	ItIt is used to filters collection based on specified type
AsEnumerable	It is used to convert input elements as IEnumerable
AsQueryable	It converts IEnumerable to IQueryable
ToDictionary	It converts input elements into dictionary based on key selector function

CONVERSION OPERATORS

- ToList() operator: ép kiểu các phần tử về List

```
class Program
{
    static void Main(string[] args)
    {
/create array countries of type string containing the collection of data
        string[] countries = { "US", "UK", "India", "Russia", "China", "Australia", "Argentina" };
/countries.ToList() convert the collection of data into the list.
        List<string> result = countries.ToList();
  //foreach loop is used to print the information of the student
        foreach (string s in result)
        {
            Console.WriteLine(s);
        }
            Console.ReadLine();
    }
}
```

CONVERSION OPERATORS

- LINQ ToArray(): ép kiểu các phần tử sang Array

```
        static void Main(string[] args)
        {
//Create array countries of string type containing the data.
            string[] countries = { "Uk", "Us", "Russia", "India", "Argentina", "Australia", "China" };
//countries.ToArray() is used to convert the collection of data into the form of array
            string[] countryarray = countries.ToArray();
//foreach loop is used to print the name of the countries
            foreach (string s in countryarray)
            {
                Console.WriteLine(s);
            }
                Console.ReadLine();
        }
    }
```

CONVERSION OPERATORS

- LINQ ToLookup() Operator: ép các giá trị trong danh sách/tập hợp về định dạng kiểu key/value

```
staticvoid Main(string[] args)
{
List<Employee> objEmployee = new List<Employee>()
{
new Employee(){ Name="Ashish Sharma", Department="Marketing", Country="India"},
new Employee(){ Name="John Smith", Department="IT", Country="Australia"},
new Employee(){ Name="Kim Jong", Department="Sales", Country="China"},
new Employee(){ Name="Marcia Adams", Department="HR", Country="USA"},
new Employee(){ Name="John Doe", Department="Operations", Country="Canada"}
};
var emp = objEmployee.ToLookup(x => x.Department);

Console.WriteLine("Grouping Employees by Department");
Console.WriteLine("---------------------------------");

foreach (var KeyValurPair in emp)
{
Console.WriteLine(KeyValurPair.Key);

// Lookup employees by Department

foreach (var item in emp[KeyValurPair.Key])
{
Console.WriteLine("\t" + item.Name + "\t" + item.Department + "\t" + item.Country);
}
```

```
class Employee
{
public string Name { get; set; }
public string Department { get; set; }
public string Country { get; set; }
}
```

CONVERSION OPERATORS

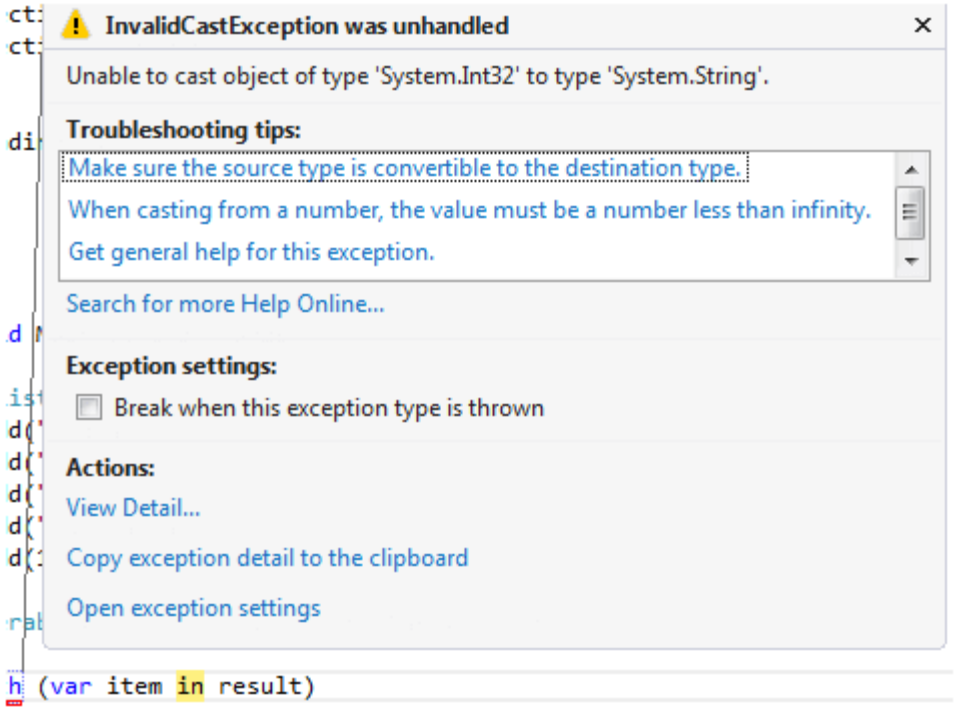
- LINQ Cast Conversion Operator: ép các giá trị trong danh sách/tập hợp sang kiểu mới

```
static void Main(string[] args)
{
ArrayList obj = new ArrayList();
obj.Add("India");
obj.Add("USA");
obj.Add("UK");
obj.Add("Australia");
IEnumerable<string> result = obj.Cast<string>();
foreach (var item in result)
{
Console.WriteLine(item);
}
Console.ReadLine();
}
}
}
```

CONVERSION OPERATORS

- LINQ Cast Conversion Operator: Sẽ bị văng Exception khi ép các kiểu ko phù hợp

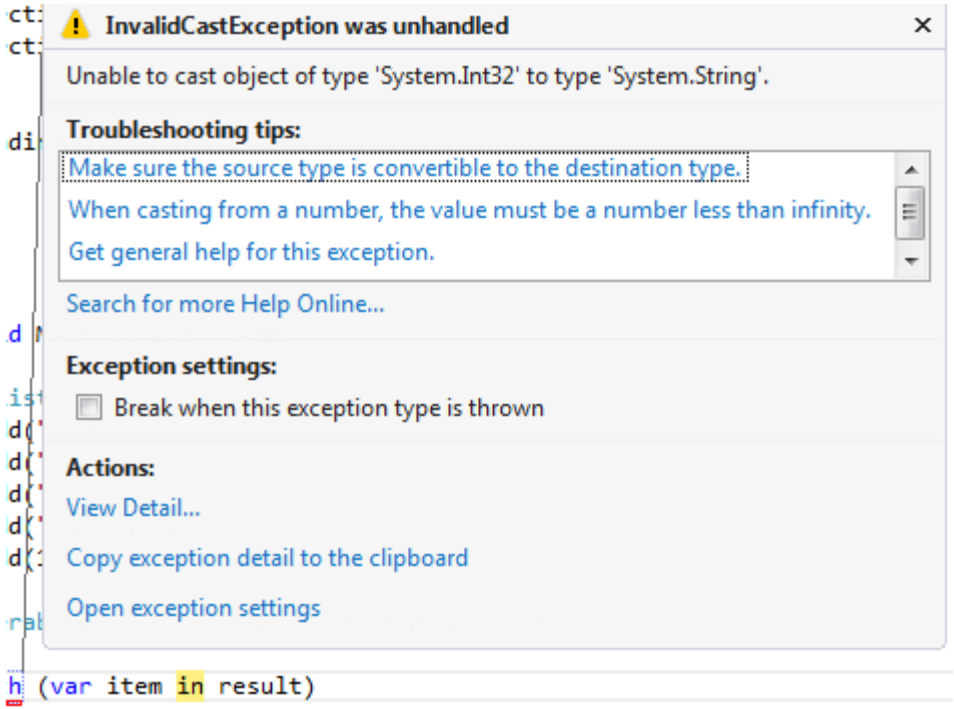
```
static void Main(string[] args)
{
    ArrayList obj = new ArrayList();

    obj.Add("India");

    obj.Add("USA");

    obj.Add("UK");

    obj.Add("Australia");

    obj.Add(1);

    IEnumerable<string> result = obj.Cast<string>

    foreach (var item in result)
    {
        Console.WriteLine(item);
    }
    Console.ReadLine();
}
```



CONVERSION OPERATORS

- LINQ Cast Conversion Operator: Sẽ bị văng Exception khi ép các kiểu ko phù hợp

```
static void Main(string[] args)
{
    ArrayList obj = new ArrayList();

    obj.Add("India");

    obj.Add("USA");

    obj.Add("UK");

    obj.Add("Australia");

    obj.Add(1);

    IEnumerable<string> result = obj.Cast<string>

    foreach (var item in result)
    {
        Console.WriteLine(item);
    }
    Console.ReadLine();
}
```



CONVERSION OPERATORS

- LINQ OfType Operator: Trả về các phần tử thỏa mãn kiểu dữ liệu chỉ định

```
        static void Main(string[] args)
        {
//Create an object of ArrayList and add the values
            ArrayList obj = new ArrayList();
            obj.Add("Australia");
            obj.Add("India");
            obj.Add("UK");
            obj.Add("USA");
            obj.Add(1);
//ofType() method will return the value only the specific type
            IEnumerable<string> result = obj.OfType<string>();
   //foreach loop is applied to print the value of the item
            foreach (var item in result)
            {
                Console.WriteLine(item);
            }
                Console.ReadLine();
        }
    }
```

CONVERSION OPERATORS

- LINQ AsEnumerable: Ép kiểu dữ liệu xác định sang kiểu IEnumerable

```
class Program1
{
    static void Main(string[] args)
    {
//here we are creating an array NumArray type of int
        int[] NumArray = new int[] { 1, 2, 3, 4,5};
//After applying the AsEnumerable method the output will be store in variable result
        var result = NumArray.AsEnumerable();
//Now we will print the value of variable result one by one with the help of foreach loop
        foreach (var number in result)
    {
        Console.WriteLine(number);
    }
        Console.ReadLine();
    }
    }
```

CONVERSION OPERATORS

- LINQ ToDictionary: Chuyển đổi kiểu dữ liệu xác định trong danh sách/tập hợp (IEnumerable<T>) sang (Dictionary<TKey,TValue>)

CONVERSION OPERATORS

LINQ ToDictionary

```
static void Main(string[] args)
{
//Create a object objStudent of Student class and add the information of student in the List
        List<Student> objStudent = new List<Student>()
        {
            new Student() { Id=1,Name = "Vinay Tyagi", Gender = "Male",Location="Chennai" },
            new Student() { Id=2,Name = "Vaishali Tyagi", Gender = "Female", Location="Chennai" },
            new Student() { Id=3,Name = "Montu Tyagi", Gender = "Male",Location="Bangalore" },
            new Student() { Id=4,Name = "Akshay Tyagi", Gender = "Male", Location ="Vizag"},
            new Student() { Id=5,Name = "Arpita Rai", Gender = "Male", Location="Nagpur"}
        };
   /*here with the help of ToDictionary() method we are converting the colection
   of information in the form of dictionary and will fetch only the required information*/
            var student = objStudent.ToDictionary(x => x.Id, x => x.Name);
   //foreach loop is used to print the information of the student
            foreach (var stud in student)
            {
                Console.WriteLine(stud.Key + "\t" + stud.Value);
            }
            Console.ReadLine();
}
```

```
class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Gender { get; set; }
    public string Location { get; set; }
}
```

DEMO

- Hiện thực các ví dụ

Tổng kết bài học

- LINQ Aggregate Functions
- LINQ Operators

KẾT THÚC