Introduction to Functional Programming in C#

Worawit - Jetabroad (Thailand)
worawit.so@bkk.jetabroad.com

Overview

- What is Functional Programming?
  - C#? Is that for Functional Programming?
- Paradigm Shift Imperative vs Functional
  - The Functional Programming
  - Demo
  - Thinking in Functional Programming
- What still missing in C#?

What is Functional Programming?

- Functional Programming is a programming paradigm that will help reducing complexity in Coding by introducing the following
  - Function as a first-class members
  - Higher Order Functions
  - Referential Transparency
    - expression always evaluates to the same result in any context.
  - Function Honesty
    - What is told, is what is got
  - Immutability
    - No side effect

Function as a First Class member

- A function can be pass along like a value

Example in Haskell (Including Implementation of Map)

```
map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs
```

Example in C# (Definition Only)

```
public static IEnumerable<TResult>

    Select<TSource, TResult>(this IEnumerable<TSource> source,

        Func<TSource, TResult> selector);
```

C#? Is that for Functional Programming?

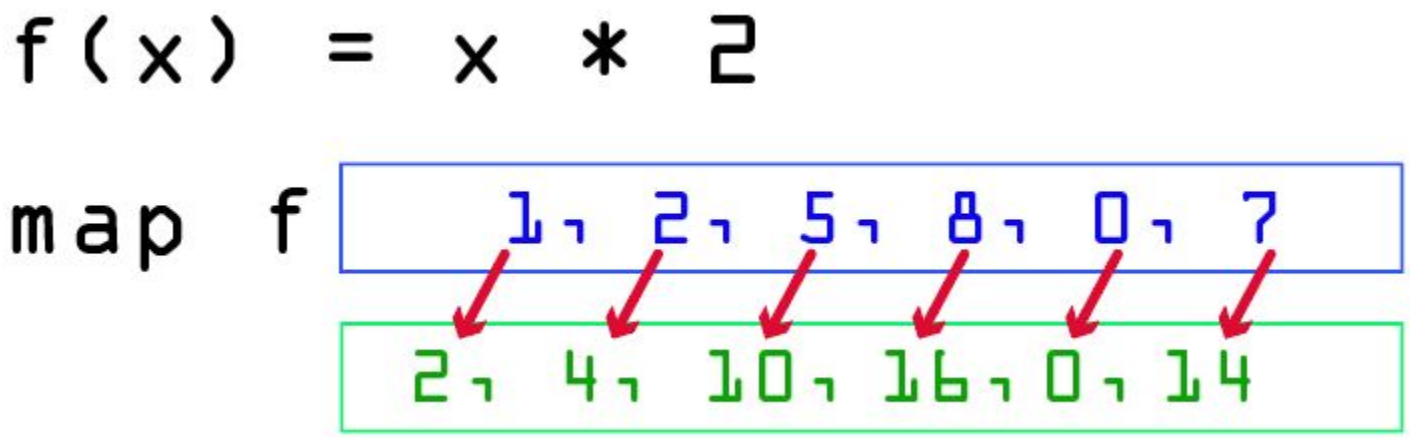
- C# implements many FP paradigm
- Using Linq arguably, means using some FP paradigm

C#? Is that for Functional Programming?

Some C# Higher Order Function Comparison

Higher Order Function	Haskell	C# Linq	Description
Mapping	map	Select	Transforming element in the list
reduce	foldl, foldr	Aggregate	Aggregate list into a result with given function
filter	filter	Where	filter only in the criteria

C#? Is that for Functional Programming?

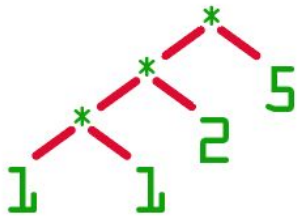


In C#

```
public static int F(int x) { return x * 2; }

…
  var mapped = new [] { 1, 2, 5, 8, 0, 7 }.Select(F);
…
  var mapped = new [] { 1, 2, 5, 8, 0, 7 }.Select(x => x * 2);
```

C#? Is that for Functional Programming?

```
f(x,y) = x * y

foldl  f 1  1, 2, 5
```

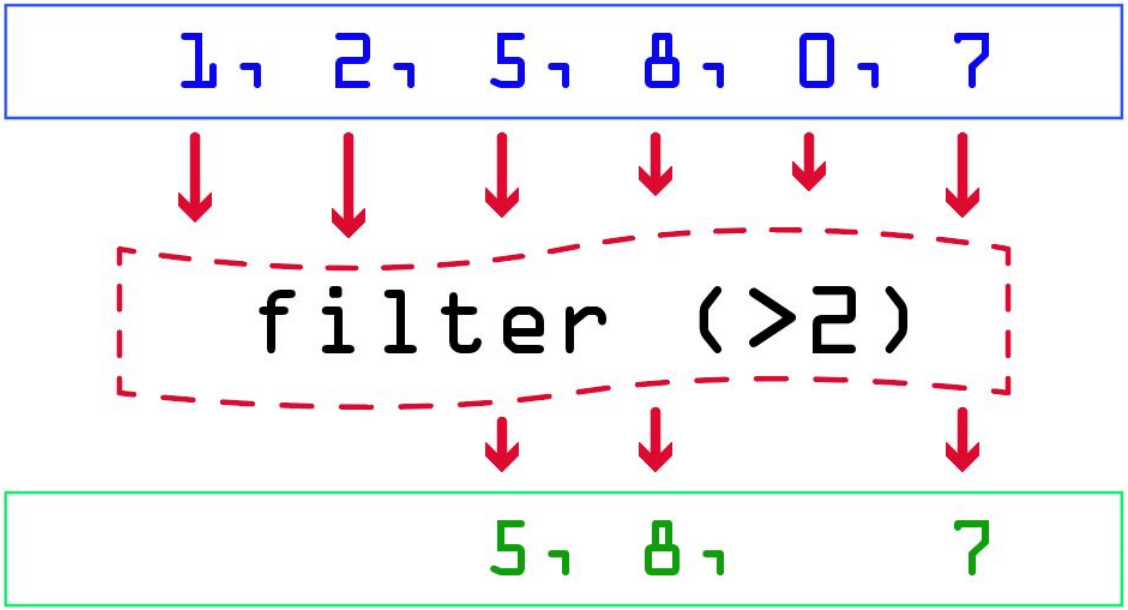



```
(((1*1)*2)*5)
```

In C#

```
public static int F(int x, int y) { return x * y; }
…
  var folded = new [] { 1, 2, 5 }.Aggregate(1, F);
…
  var folded = new [] { 1, 2, 5 }.Aggregate(1, (x, y) => x * y);
```

C#? Is that for Functional Programming?



In C#

```
public static bool F(int x) { return x > 2; }
…
  var filtered = new [] { 1, 2, 5 }.Where(F);
…
  var filtered = new [] { 1, 2, 5 }.Where(x => x > 2);
```

Referential Transparency

```
Console.WriteLine("Age {0}",
    Aging(new DateTime(2017, 1, 1)));
```

```
public static TimeSpan Aging(DateTime fromDate)
{
    return DateTime.Now - fromDate;
}
```

```
Output
Age 205.10:44:34.5871568
Age 205.10:44:34.7056017
Age 205.10:44:34.8250461
Age 205.10:44:34.9405167
Age 205.10:44:35.0560095
Age 205.10:44:35.1571520
```

```
Console.WriteLine("Age {0}",
    Aging(new DateTime(2017, 1, 1), DateTime.Now);
```

```
3 references | 0 authors, 0 changes
public static TimeSpan Aging(DateTime fromDate, DateTime toDate)
{
    return toDate - fromDate;
}
```

```
Output
Age 205.10:50:33.2709742
Age 205.10:50:33.3712868
Age 205.10:50:33.4933271
Age 205.10:50:33.6092376
Age 205.10:50:33.7095493
Age 205.10:50:33.8250187
```

Function Honesty

```
Utility.SeqFromTo(10, 0, -1)
    .Select(elem => elem.Reciprocal());
```

```
1 reference | 1 author, 1 change
public static double Reciprocal(this BigInteger number)
{
    return 1.0 / (double)number;
}
```

```
2 references | 1 author, 1 change
public static Nullable<double> Reciprocal(this BigInteger number)
{
    if (number == 0) return null;
    return 1.0 / (double)number;
}
```

Output

```
[0.1,
0.111111111111111,
0.125,
0.142857142857143,
0.166666666666667,
0.2,
0.25,
0.333333333333333,
0.5,
1,
Throw exception]
```

Output

```
[0.1,
0.111111111111111,
0.125,
0.142857142857143,
0.166666666666667,
0.2,
0.25,
0.333333333333333,
0.5,
1,
null]
```

Immutability

```
6 references | 0 authors, 0 changes
public static class Immutability
{
    public static BigInteger accumulated = 0;
    3 references | 0 authors, 0 changes
    public static BigInteger AddAndStore(BigInteger number)
    {
        accumulated += number;
        return accumulated;
    }
}
```

```
TestAndAdjustData();
Immutability.AddAndStore(5);
Immutability.AddAndStore(5);
Immutability.AddAndStore(5);

/// What should this print?
Console.WriteLine("Print Current = {0}", Immutability.accumulated);
```

Output

```
Print Current = 26
```

Immutability

Something hidden here

```
1 reference | 0 authors, 0 changes
private static void TestAndAdjustData()
{
    Console.WriteLine("Testing Data = {0}", Immutability.accumulated);
    Immutability.accumulated = 11;
}
```

C#? Is that for Functional Programming?

- Function As First Class member
  - delegate
  - Func<T1, T2>
  - Lambda
- Linq
  - Higher Order Function

C#? Is that for Functional Programming?

```
0 references | 1 author, 1 change
public static IEnumerable<BigInteger> NumberFactorize(this BigInteger number)
{
    var firstPart = Factorize
    (
        number,
        Utility.SeqFrom(2).TakeWhile(x => x + x <= number)
    ).ToList();

    return firstPart.Union
    (
        firstPart.Select(x => number / x)
    ).OrderBy(x => x).Prepend(1);
}
1 reference | 0 authors, 0 changes
public static IEnumerable<BigInteger> Factorize
(
    BigInteger number,
    IEnumerable<BigInteger> numberList
)
{
    return numberList.Where(x => number % x == 0);
}
```

```
Number Factor for 120 is 1,2,3,4,5,6,8,10,12,15,20,24,30,40,60
Press any key to continue . . .
```

Paradigm Shift Imperative vs Functional

```
Enumerable.Range(0, 1000)
        .Where(x => x % 2 == 0)
        .Where(x => x % 3 == 0)
        .Where(x => x % 5 == 0)
        .Take(20)
```

```
var itemList = new List<BigInteger>();
for (int i = 0; i < to && itemList.Count < 20; i++)
{
    if (i % 2 == 0
        && i % 3 == 0
        && i % 5 == 0)
    {
        itemList.Add(i);
    }
}
```

Paradigm Shift Imperative vs Functional

```
Enumerable.Range(0, 1000)
    .Aggregate((x, y) => x + y)
```

```
BigInteger sum = 0;
for (var i = from; i <= to; i++)
{
    sum += i;
}
```

```
Enumerable.Range(0, 1000)
    .Zip(someSeq.Reverse(), (x, y) => (x, y))
    .Take(10)
```

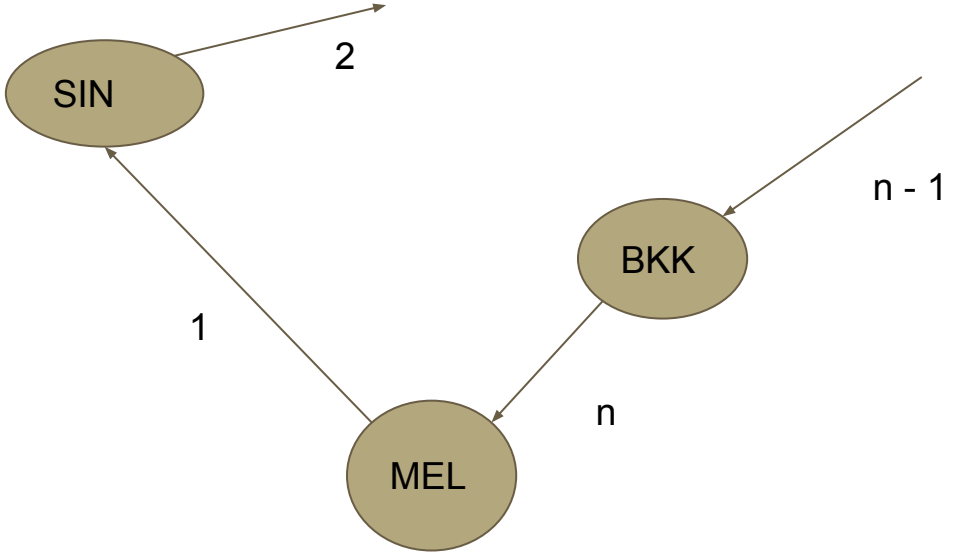
```
var toTake = 0;
var tupleList = new List<(BigInteger, BigInteger)>();
for (BigInteger i = from, j = to; i <= to; i++, j--)
{
    if (toTake < 10)
    {
        tupleList.Add((i, j));
    }

    toTake++;
}
```

The Functional Programming in C#

Let’s do the Demo

The Functional Programming in C#

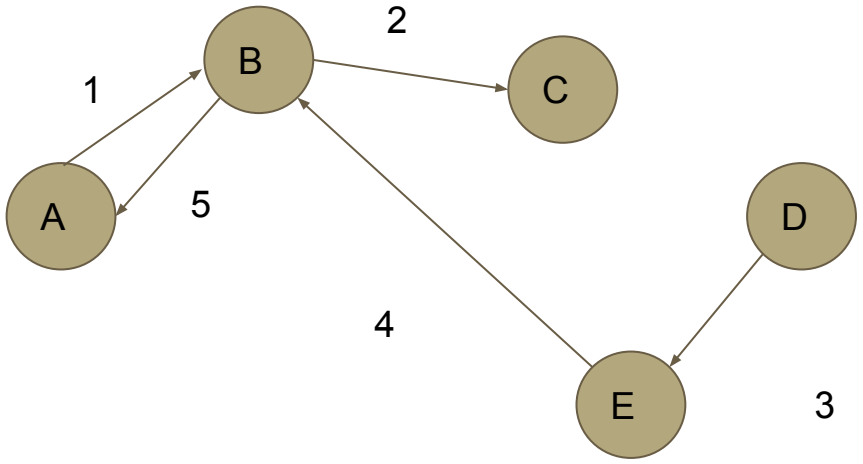
- Functional Programming in Real World Example
- Open Jaw flight rule



The Functional Programming in C#

- For the List of Itinerary
- Get the first on in the list
- Find the matched Open Jaw from the List
- Once found the Open Jaw, find more Open Jaw from the rest of the itinerary
- Otherwise, Find the Open Jaw with the next city in the itinerary

The Functional Programming in C#




```
FlightSegmentList = [1, 2, 3, 4, 5]
OpenJaw(1, [2, 3, 4, 5]) -> [1, 5],[2, 3, 4]
OpenJaw(2, [3, 4]) -> [2, 4], [3]
OpenJaw(3, []) => [], []
OpenJaw = [1, 5], [2, 4]
NonOpenJaw = [3]
```

The Functional Programming in C#

```
1 reference | 1 author, 1 change
public static (IEnumerable<IEnumerable<FlightSegment>> OJ, IEnumerable<FlightSegment> Other) AllOpenJaws(this FlightQuery flightQuery)
{
    var allOpenJaws = AllOpenJawsInternal(flightQuery);

    var allOpenJawsFlatten = allOpenJaws.SelectMany(x => x);
    IEnumerable<FlightSegment> allSegments = flightQuery.FlightSegments;
    return (allOpenJaws, allSegments.Except(allOpenJawsFlatten));
}
3 references | 1 author, 3 changes
private static IEnumerable<IEnumerable<FlightSegment>> AllOpenJawsInternal(this FlightQuery flightQuery)
{
    var head = flightQuery.FlightSegments.First();
    var tail = flightQuery.FlightSegments.Skip(1).ToList();

    if (tail.Count() >= 1)
    {
        // Modify this part to change the logic for the OpenJaw
        var openJaw = tail.Where(elem => elem.Destination == head.Origin && elem.Departure > head.Departure).ToList();
        if (openJaw.Any())
        {
            var theRest = tail.Except(openJaw).ToList();
            return new[] { openJaw.Prepend(head) }.Concat(
                theRest.Any()
                    ? new FlightQuery { FlightSegments = theRest.ToArray() }.AllOpenJawsInternal()
                    : new List<IEnumerable<FlightSegment>>().ToArray());
        }
        else
        {
            return new FlightQuery { FlightSegments = tail.ToArray() }.AllOpenJawsInternal();
        }
    }

    return new List<IEnumerable<FlightSegment>>().ToArray();
}
```

The Functional Programming in C#

Full Itinerary

[(MEL,BKK),(BKK,PAR),(SIN,LON),(LON,BKK),(PEK,MEL)]

OpenJaws

[(MEL,BKK),(PEK,MEL)]

OpenJaws

[(BKK,PAR),(LON,BKK)]

Other

[(SIN,LON)]

What still missing in C#?

- Tail call optimization
  - Prevent Stack Overflow
- Better Type inference

```
(*
// first-class function.
Func<int, int, int> add = (a, b) => a + b;

// higher-order functions.
Func<int, Func<int, int>> createAdder =
    x => new Func<int, int>(
        y => x + y
    );

var f = createAdder(2); // f(x) = x + 2
Console.WriteLine(f(1)); // 3
*)

let add a b = a + b

let createAdder x = add x
```

Q & A