Meta Linguistic Abstractions (Part 3)

Praseed Pai K.T.

Have u come across a declarative language inside an Procedural language?

- ????

The Answer is

- Language INtegrated Query (LINQ) inside C#
- T-SQL and PL/SQL is other way around (Imperative in Declarative)

What is a Procedural Programming language?

- The Program is modelled as Data and Procedures mutate data to perform Computation
- A language like Pascal, C++,C# all support Procedural Programming

What is a Declarative Programming language?

- You specify your intent and how to achieve that will taken care by the System
- SQL is a declarative Programming language
- HTML,CSS,XSLT are other declarative programming languages

LINQ

LINQ is a Query Subsystem integrated into the C# and VB.net programming languages to query data in a uniform manner against Object Hierarchies , Databases , XML and any kind of structured data.

LINQ Query Syntax

- Compreshension Syntax
- Lambda Syntax
- Mixed Mode Syntax

Lambda Syntax

```
static void LambdaSyntax() {
    char [] msg = { 'H' , 'e' ,'l','l' ,'o',' ',
                    'W' , 'o' ,'r','l','d' };
    IEnumerable<char> temp =
            msg.Where(( n) => (n !='l'));
    foreach (char ch in temp) { Console.Write(ch);}
}
```

Comprehension Syntax

```
static void ComprehensionSyntax(){
    char[] msg = { 'H' , 'e' ,'l','l' ,'o',' ',
                   'W' , 'o' ,'r','l','d' };
    IEnumerable<char> temp = from n in msg
                   where n != 'H'
                   select n.ToString().ToUpper()[0];
    foreach (char ch in temp){ Console.Write(ch);}
}
```

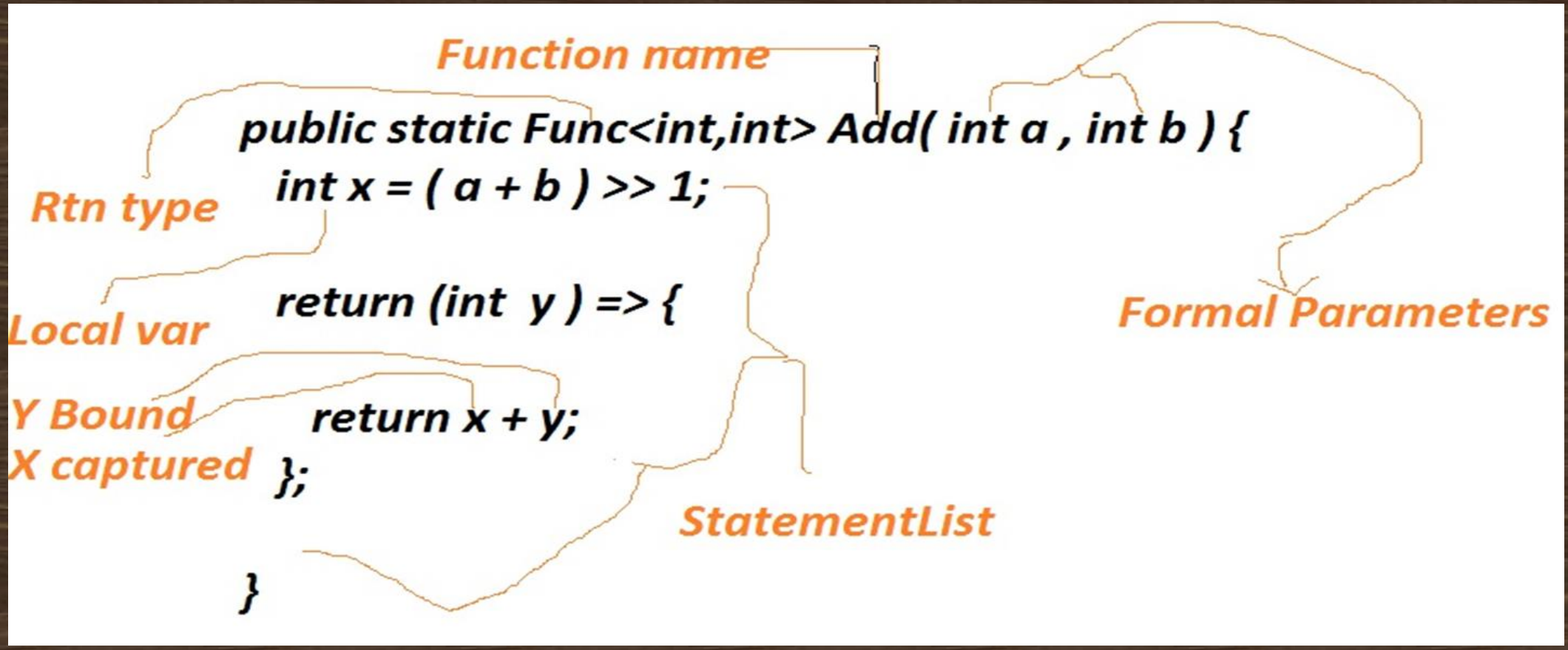
Mixed Mode Syntax

```
static void MixedModeSyntax(){
        char[] msg = { 'H' , 'e' ,'l','l' ,'o',' ',
                'W' , 'o' ,'r','l','d' };
        IEnumerable<char> temp = (from n in msg
                        where n >= 32
                        select n).Select((n) =>
                            n.ToString().ToUpper()[0]);
        foreach (char ch in temp){ Console.Write(ch);}
}
```

What the heck is Lambda?

- Lambda Syntax is based on Lambda Calculus
- Lambda Calculus was invented way before a physical computer was engineered
- Alonzo Church , was trying to solve Hilbert's 10th problem
- All Functional Programming Languages are based on Lambda Calculus
- Some good examples are Scheme , F#
- Scheme uses untyped Lambda Calculus
- F# uses Typed Lambda Calculus (ML , OCCAML LINEAGE)

Lambda (Function) in C#



Map function in Scheme

```
( map (lambda(m) ( * m m ) ) '( 1 2 3 4 ) )
(define sqr  ( lambda(m) (* m m )) )
( map sqr '(4 5))
```

Map/Reduce function in Scheme

```
( define  ( custom_map f x )
     ( cond (( null? x ) '() )
          (else ( cons (f (car x ))
                       (custom_map f ( cdr x ))))))
( define ( reduce f x v ) ( cond ((null? x ) v)
                         (else (f (car x ) (reduce f
                                  (cdr x ) v )))))
(reduce ( lambda(a b ) ( + a b ) )
(custom_map (lambda(a) ( * a a ) ) '(1 2 3)) 0 )

(reduce ( lambda(a b ) ( * a b ) )
(custom_map (lambda(a) ( * a a ) ) '(1 2 3)) 2 )
```

Map/Reduce in C#

```
 public static IEnumerable<T>
          Maps<T>(this IEnumerable<T> x, Func<T, T> f) {
          List<T> n = new List<T>();
          foreach (T t in x) { n.Add(f(t));}
          return n;
}
 public static   T Reduce<T>(this IEnumerable<T> x,
                                        Func<T, T, T> f, T init)   {
          T s = init;
          foreach (T t in x) { s = f(s, t); }
          return s;
}
```

Applying Map/Reduce

```
static void Main(string[] args){
        Func<double, double, double>
                adder = (x, y) => x + y;
        double[] arr = { 10, 20, 30 };
        IEnumerable<double> mapped =
                arr.Maps<double>( (x) => x * x);
        double n = mapped.Reduce<double>( adder, 0);
        n = mapped.Reduce<double>( (x, y) => x + y,0);
        Console.WriteLine(n);
        Console.Read();
}
```

Q&A

- If any?