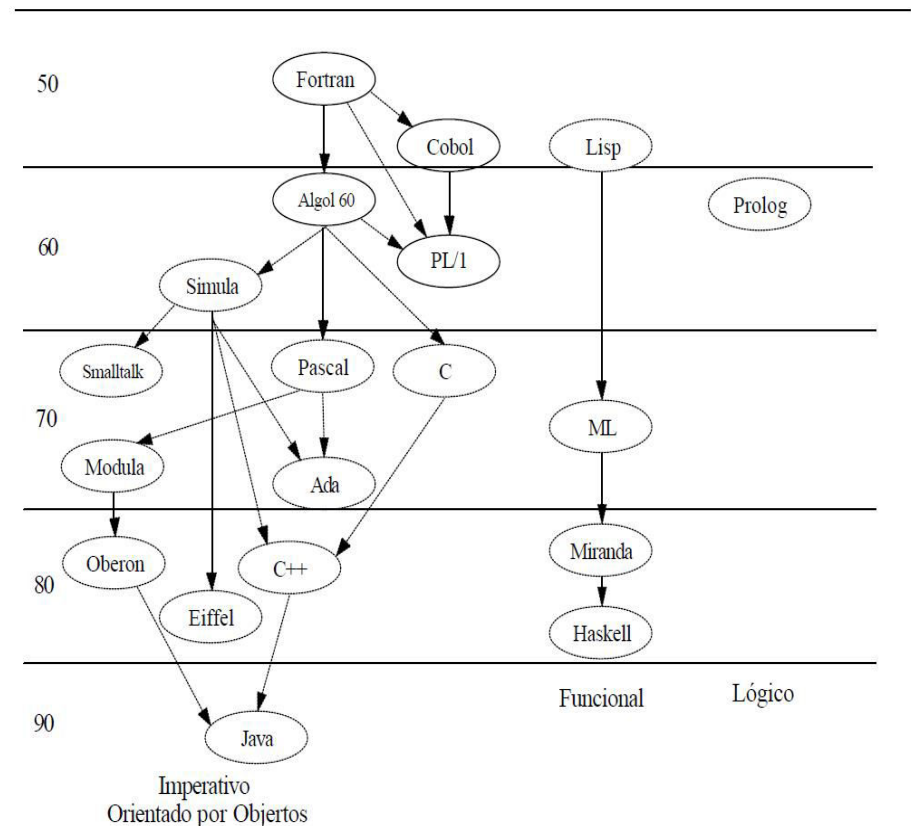Linguagem de Programação Haskell

Origem

- Final dos anos 80

- Linguagem Miranda

Classificação

- Funcional

- Estática

Writability

- Haskell vs Java

Haskell

```
let shoppingList = ["Eggs", "Milk"]
```

Java

```
ArrayList<String> shoppingList = new ArrayList<>();
shoppingList.add("Eggs");
shoppingList.add("Milk");
```

Compreensão de listas

- Haskell vs C#

```
npm - ghci

ghci>doublesList xs = [x*2 | x <- xs]
ghci>doublesList [1..10]
[2,4,6,8,10,12,14,16,18,20]
ghci>
```

```csharp
public class Program
{
    public static void Main(string[] args)
    {
        var list = new List<int>();
        list.AddRange(new int[] { 1,2,3,4,5,6,7,8,9,10 });
        var result = DoublesList(list);
    }

    public static List<int> DoublesList(List<int> list)
    {
        var list2 = new List<int>();
        list.ForEach(x => { list2.Add(x * 2); });
        return list2;
    }
}
```

Lazy Evaluation

```
Positives = [0..]
```

```
npm - ghci

ghci>list = [1..]
ghci>100 `elem` list
True
ghci>1000000000 `elem` list
True
ghci>
```

```
1    main = do
2        let myTuple = ("first", map (*2) [1,2,3,4])
3        print "Hello"
4        print $ fst myTuple
5        print $ snd myTuple
6
```

```
function1 :: Int
function1 = function2 exp1 exp2 exp3
  where
    exp1 = reallyLongFunction 1234
    exp2 = reallyLongFunction 3151
    exp3 = reallyLongFunction 8571

function2 :: Int -> Int -> Int -> Int
function2 exp1 exp2 exp3 = if exp1 < 1000
  then exp2
  else if exp1 < 2000
    then exp3
    else exp1
```

Lazy Evaluation

```csharp
public class Program
{
    public static void Main(string[] args)
    {
        var result = Function1(Function2(ReallyLongFunction(),ReallyLongFunction(),ReallyLongFunction()));

        Console.WriteLine($"Resultao = {result}");
        Console.ReadKey();
    }

    public static int Function1(int x)
    {
        return x;
    }

    public static int Function2(int exp1, int exp2, int exp3)
    {
        if (exp1 < 1000)
            return exp2;
        else if (exp1 < 2000)
            return exp3;
        else
            return exp1;
    }

    public static int ReallyLongFunction()
    {
        var random = new Random();
        Thread.Sleep(5000);

        return random.Next(0,3000);
    }
}
```

Pattern Matching

```haskell
numDaSorte :: (Integral a) => a -> String
numDaSorte 13 = "Você está com sorte !!"
numDaSorte x = "Opa, parece que você está com azar..."
```

Haskell

```java
public string NumeroDaSorte(int a){
        if(a == 7){
                return "Você está com sorte !!"
        } else {
                return "Opa, parece que você está com azar..."
        }
}
```

Java

Pattern Matching

```haskell
factorial :: (Integral a) => a -> a
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Haskell

```c
#include<stdio.h>

int fat, n;

int main()
{
  scanf("%d", &n) ;
  for(fat = 1; n > 1; n = n - 1)
  {
      fat = fat * n;
  }
  printf("\n%d", fat);
  return 0;
}
```

C

Currying Functions

Haskell

```
ghci> max 4 5
5
ghci> (max 4) 5
5
```

```
1    fatorial :: (Integral a) => a -> a
2    fatorial 0 = 1
3    fatorial n = n * fatorial (n -1)
4
5    somaDoFatorial :: [Int] -> Int
6    somaDoFatorial xs = sum [fatorial x | x <- xs]
7
8    main = print $ somaDoFatorial [1..5]
9
```

```
λ runhaskell curried.hs
153
```

```
npm - ghci

ghci>addThreeNumbers x y z = x + y + z
ghci>addThreeNumbers 2 4 6
12
ghci>
```

Currying Functions

```c
1    #include<stdarg.h>
2    #include<stdio.h>
3
4    long int factorial(int n){
5        if(n>1)
6            return n*factorial(n-1);
7        return 1;
8    }
9
10   long int sumOfFactorials(int num,...){
11       va_list vaList;
12       long int sum = 0;
13
14       va_start(vaList,num);
15
16       while(num--)
17           sum += factorial(va_arg(vaList,int));
18
19       va_end(vaList);
20
21       return sum;
22   }
23
24   int main()
25   {
26       printf("\nSum of factorials of [1,5] : %ld",sumOfFactorials(5,1,2,3,4,5));
27
28       return 0;
29   }
```

```
Sum of factorials of [1,5] : 153
--------------------------------
Process exited after 0.03017 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```