Linguagens de Programação – Lista 2

Aluno: Paulo Fábio dos Santos Ramos

**1)** Programação funcional consiste no uso de funções matemáticas, sendo que diferentemente da programação imperativa, nesse paradigma não existem variáveis de estado, ou seja, se um valor qualquer aparece em algum momento qualquer da computação, considera-se que aquele valor é único e imutável durante todo o processo, onde para cada tipo de entrada existe apenas um elemento de saída Além disso, as funções são normalmente expressas por meio de outras funções - de modo que obter o valor da função para um determinado conjunto de parâmetros envolve não só aplicar as regras daquela função, mas também fazer uso de outras funções.

**Vantagens:** Alto nível de abstração; ausência de operações de atribuição o que torna os programas funcionais muito mais simples para provas e análises matemáticas do que os programas procedurais; Composição de funções é uma forma simples de otimização.

**Desvantagens:** Menor eficiência; Problemas que envolvam muitas variáveis ou muitas atividades sequenciais; Difícil de prever performance e requisitos.

**2)** Funções puras: Se uma função for chamada múltiplas vezes com os mesmos argumentos, ela sempre irá retornar o mesmo valor. Ex.: Soma a b = a + b.

Estados imutáveis: Uma vez que o valor de uma variável for definido ele não pode ser modificado. Ex. x = 10

x = div 6 3

Funções de ordem maior: Funções podem retornar funções e receber funções como argumentos.

Ex.: Calcular as raízes de uma função do segundo grau.

```
delta a b c = (b*b) - (4 * a * c)
x1 a b c = (-b + sqrt(delta a b c)) / ( 2 * a)
x2 a b c = (-b - sqrt(delta a b c)) / ( 2 * a)

resultado a b c = ((x1 a b c), (x2 a b c))

main = do
        print (resultado 1 3 (-4))
```

Composição de funções: A partir de duas funções é possível gerar uma nova função que executa as duas de forma simultânea.

Ex.:

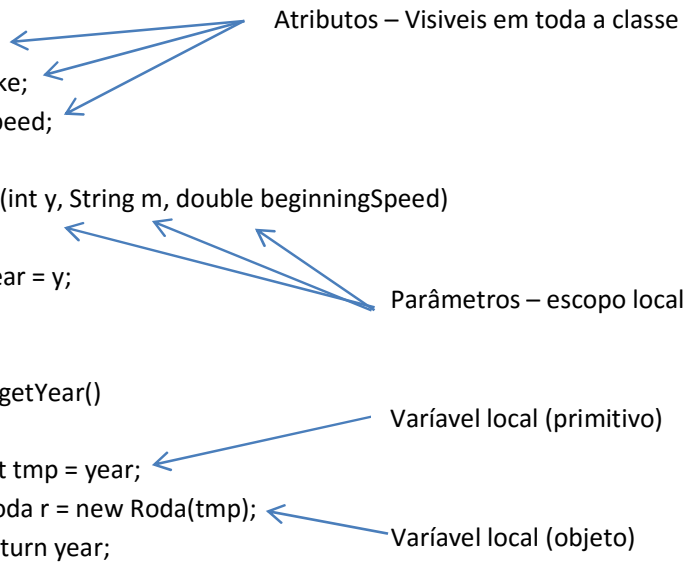
```
soma a b = a + b
subi a b = b - a
resultado a b = ((soma a b)*(subi a b))

main = do
        print (resultado 5 7)
```

**3)** Facebook: Atualmente utiliza Haskell para o estabelecimento de regras para o filtro de spam, malware e outras coisas abusivas.
Twitter: Utiliza Scala para escrita de seu backend
GitHub: até 2010 ultilizava Erlang no seu sistema de backend, lidando com milhares de transações concorrentes.

**4)** O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados. Consiste no uso da abstração de dados para descrever algo do mundo real durante o desenvolvimento do programa, por exemplo, podemos definir uma classe carro como algo que possui um motor, rodas, cor, fabricante, modelo, placa de identificação, etc.
Principais Vantagens: Reusabilidade de código; Escalabilidade de aplicações; Mantenabilidade; Apropriação.

**5)** 1.public class Car

```
2. {
3.      int year;
4.      String make;
5.       double speed;
6.
7.      public Car(int y, String m, double beginningSpeed)
8.      {
9.              year = y;
10.     }
11.
12.     public int getYear()
13.     {
14.             int tmp = year;
15.             Roda r = new Roda(tmp);
15.             return year;
16.     }
17.}
```




**6)** Na linha 21 o endereço do ponteiro b sofre um incremento o que faz com que ele não aponte mais para um espaço alocado dinamicamente, tornando o free(b) inutil pois o endereço do que estava sendo alocado dinamicamente anteriormente agora não existe.

**7) a) Haskell**

```
main = do
        inputjar <- getLine
        let nota1 = read inputjar::Double
        inputjar <- getLine
        let nota2 = read inputjar::Double
        inputjar <- getLine
        let nota3 = read inputjar::Double
        inputjar <- getLine
        let nota4 = read inputjar::Double
        let media = (nota1+nota2+nota3+nota4)/4
        if media>=7 then do
                print("Aluno Aprovado")
                print(media)
        else do
                inputjar <- getLine
                let ex = read inputjar::Double
                let fin = (media+ex)/2
                if fin>=5 then do
                        print("Aluno Aprovado em Exame")
                        print(fin)
                else do
                        print("Aluno Reprovado")
                        print(fin)
```

**b)Haskell**

```
main = do
        inputjar <- getLine
        let n = read inputjar::Int
        if n `mod` 2 == 0 then print("Par")
        else print("Impar")
```