Engenharia de Software

Paradigmas de Programação

- 2ª Entrega -

Paradigma Funcional

Professora: Milene Serrano

Grupo: Álex Mesquita

Jefferson Xavier

Priscilla Gonçalves

Ramom Cruz

**Relatório**

1. **Introdução**

Neste tópico serão demonstradas questões relacionadas ao Paradigma Funcional, ao qual se relaciona este trabalho.

* O paradigma funcional é utiliza aplicação de funções através da avaliação de [funções matemáticas](http://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_matem%C3%A1tica) e evita estados ou dados mutáveis; por isso, a linguagem oferece funções básicas que não requerem definições adicionais.
* Uma função pode ser considerada a unidade básica na programação funcional, e, pode ter ou não [parâmetros](http://pt.wikipedia.org/wiki/Par%C3%A2metro_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)) (valores de entrada da função utilizados como referência para que ela seja executada) e um valor de retorno (valor de retorno). A definição de uma função descreve como a função será avaliada em termos de outras funções.
* As funções podem ser manipuladas em uma grande variedade de formas, pois são tratadas como valores de primeira importância, o que é o mesmo que dizer que funções podem ser parâmetros ou valores de entrada para outras funções e podem ser os valores de retorno ou saída de uma função. Então podemos entender paradigma funcional como um mapeamento dos valores de entrada nos valores de retorno, através de funções.
* Funções podem ser nomeadas, como em outras linguagens, ou definidas anonimamente (algumas vezes durante a execução do programa) usando uma abstração [lambda](http://pt.wikipedia.org/wiki/C%C3%A1lculo_lambda) e usadas como valores em outras funções. Permite reescrita de funções com múltiplos parâmetros como a composição de funções de um parâmetro. O resultado é uma função onde os parâmetros neste subconjunto são agora fixados como constantes, e os valores do resto dos parâmetros ainda não são especificados.
* Algumas linguagens notáveis usadas na indústria e no comércio incluem [Erlang](http://pt.wikipedia.org/wiki/Erlang_(linguagem_de_programa%C3%A7%C3%A3o)), [cálculo lambda](http://pt.wikipedia.org/wiki/C%C3%A1lculo_lambda), as linguagens de programação [APL](http://pt.wikipedia.org/wiki/APL) e [Lisp](http://pt.wikipedia.org/wiki/Lisp), e mais recentemente [ML](http://pt.wikipedia.org/wiki/ML_(linguagem_de_programa%C3%A7%C3%A3o)),[Haskell](http://pt.wikipedia.org/wiki/Haskell_(linguagem_de_programa%C3%A7%C3%A3o)), [OCaml](http://pt.wikipedia.org/wiki/OCaml) e [F#](http://pt.wikipedia.org/wiki/F_Sharp).

1. **Descrição do programa**
   1. **Objetivo**

O programa se trata de um jogo dinâmico onde várias situações cotidianas são apresentadas, e para cada uma delas várias alternativas são exibidas, a respeito de ações que poderiam ser tomadas na situação exposta. O usuário então pode escolher a opção desejada, correspondente ao que ele faria se estivesse na situação descrita (ou perto disso), e é levado para outra situação, a partir da escolha realizada.

O usuário só sai da aplicação quando desejar, através do comando indicado no terminal.

* 1. **Estrutura**

O jogo foi feito com base na estrutura de dados **grafo,** que relaciona informações através de nós (vértices) e arestas. Diferentemente de árvores, que são estruturas de dados hierárquicas e possuem fluxo de relacionamento de um nó pai para um nó filho, nos grafos os vértices podem ter relacionamentos direcionados ou não, e os relacionamentos podem ocorrer com qualquer um dos outros vértices.

Um grafo G = (V, E) é um conjunto não-vazio V de vértices, e um conjunto E de arestas. Cada aresta é um par (vi, vj), sendo vi e vj elementos de V.

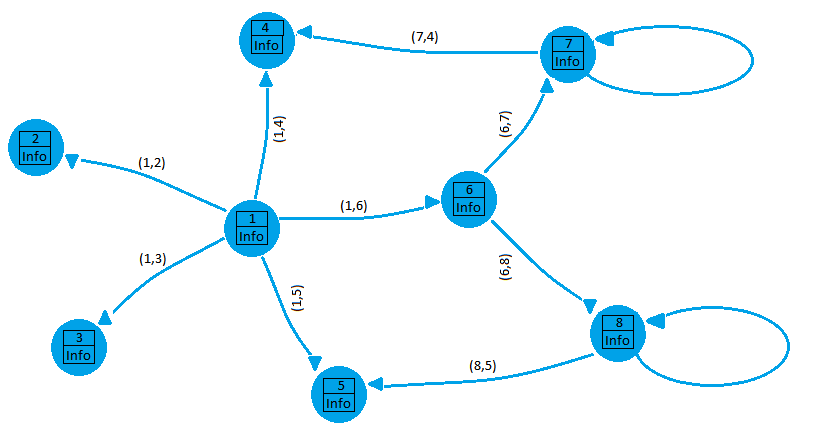
Como exemplo de um grafo direcionado, podemos considerar a malha aérea de uma região, de forma que os vértices do grafo representem os aeroportos, que são conectados por vôos.

Um grafo em linguagem C pode ser definido utilizando-se listas encadeadas que registram para cada vértice do grafo uma Lista de Adjacências. Desta forma, para cada vértice, podemos registrar à quais outros ele está ligado.

Na imagem abaixo, cada vértice do grafo é representado pelas esferas azuis, e as arestas pelas setas.

No grafo implementado no jogo, cada vértice é composto por uma tupla, contendo as duas arestas e a alternativa aser escolhida, um identificador numérico e uma informação. As arestas são representadas por pares ordenados no formato (x,y), onde **x** representa o vértice de origem e **y** representa o vértice de destino. O jogo é iniciado no vértice 1: a informação contida nele (situação) é exibida e as alternativas a serem escolhidas pelo usuário descrevendo ações que podem ocorrer na situação descrita representam outros vértices que estão ligados por arestas ao vértice 1.

Quando qualquer uma delas é escolhida, o fluxo é direcionado ao nó escolhido, que possui outra informação (situação) que levará a outra opções.



1. **Tutoriais de instalação**

**4.1. Sistema Operacional**

Os sistemas operacionais utilizados no trabalho foram o Linux/Ubuntu e o Windows (alguns integrantes utilizaram Windows e outros, Linux). Porém não houve necessidade de instalação, pois todos os integrantes já possuíam esses SO’s.

**4.2. Sublime (Utilizado no Linux)**

Instalação via linha de comando no terminal:

* sudo add–apt–repository ppa:webupd8team/sublime-text-2
* sudo apt-get update
* sudo apt-get install sublime-text

**4.3. Git**

Não foi necessário realizar instalação do Git, pois todos os integrantes já possuiam.

1. **Acesso ao repositório no Github**

< https://github.com/ParadigmasUnBFGA22014/haskell.git >