Licenciatura em Engenharia Informática e de Computadores

Fundamentos de Programação

Projecto - Segunda Parte

15 de Novembro de 2011

O objectivo da segunda parte do projecto é a definição e implementação de alguns dos tipos abstractos de informação que serão usados na realização do projecto final.

Os tipos implementados nesta segunda parte vão ser usados não só pelo programa que desenvolverá na segunda parte, como também por outros programas, nomeadamente o programa de avaliação automática. Assim, deve respeitar escrupulosamente as definições que são dadas de seguida, tanto no que diz respeito aos nomes dos procedimentos, como no que diz respeito ao número e ordem dos parâmetros formais.

Para além dos tipos abaixo descritos, deverá incluir no seu código a implementação do tipo lista simplificada apresentada no livro; não precisa, no entanto, de incluir a definição abstracta do tipo lista simplificada no seu relatório.

Para não sobrecarregar desnecessariamente o código, apenas os construtores de cada tipo devem verificar a validade dos argumentos que recebem.

1 O tipo conjunto

O tipo conjunto deve disponibilizar as seguintes operações básicas.[1]

- Construtores.
  - $novo\text{-}conj : \{\} \mapsto conjunto$
    $novo\text{-}conj()$ devolve o conjunto vazio.
  - $insere\text{-}conj : universal \times conjunto \mapsto conjunto$
    $insere\text{-}conj(e, c)$ devolve o conjunto resultante de inserir o elemento e no conjunto c. Se e já pertencer a c, $insere\text{-}conj(e, c)$ tem como valor c.
  - $faz\text{-}conj : universal^n \mapsto conjunto$
    $faz\text{-}conj(e_1, \ldots, e_n)$ devolve o conjunto de elementos e_1, ..., e_n. Por exemplo, $faz\text{-}conj(1, 2, 3, 4, 2, 2)$ devolve o conjunto $\{1, 2, 3, 4\}$.[2]
- Selector.
  - $retira\text{-}conj : universal \times conjunto \mapsto conjunto$
    $retira\text{-}conj(e, c)$ devolve o conjunto resultante de remover o elemento e do conjunto c. Se e não pertencer a c, $retira\text{-}conj(e, c)$ tem como valor c.

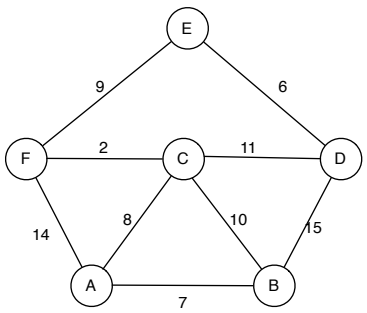
[1] Se desejar, pode basear-se na solução do exercício 5.7, apresentada no Apêndice C do livro. Utilize o procedimento primitivo `equal?` para comparar os elementos de um conjunto.

[2] A ordem dos elementos do conjunto é irrelevante.

- Reconhecedor.
  - $conjunto?$: $universal \mapsto lógico$
    $conjunto?(u)$ tem o valor $verdadeiro$ se u é um conjunto, e tem o valor $falso$, em caso contrário.
  - $conj\text{-}vazio?$: $conjunto \mapsto lógico$
    $conj\text{-}vazio?(c)$ tem o valor $verdadeiro$ se c é o conjunto vazio, e tem o valor $falso$, em caso contrário.
- Testes.
  - $conjuntos{=}?$: $conjunto \times conjunto \mapsto lógico$
    $conjuntos{=}?(c_1, c_2)$ tem o valor $verdadeiro$ se c_1 e c_2 são conjuntos iguais, e tem o valor $falso$, em caso contrário.
  - $pertence?$: $universal \times conjunto \mapsto lógico$
    $pertence?(e, c)$ tem o valor $verdadeiro$ se o elemento e pertence ao conjunto c, e tem o valor $falso$, em caso contrário.
  - $todos\text{-}elementos\text{-}conjunto\text{-}satisfazem?$: $conjunto \times predicado \mapsto lógico$
    $todos\text{-}elementos\text{-}conjunto\text{-}satisfazem?(c, p)$ tem o valor $verdadeiro$ se todos os elementos do conjunto c satisfazem o predicado p, e tem o valor $falso$, em caso contrário.
- Transformador.
  - $conj\text{->}lista$: $conjunto \mapsto lista$
    $conj\text{->}lista(c)$ devolve uma lista constituída pelos elementos do conjunto c. A ordem dos elementos na lista não é relevante.
- Operação adicional.
  - *interseccao* : $conjunto \times conjunto \mapsto conjunto$
    interseccao(c_1, c_2) devolve o conjunto intersecção de c_1 e c_2, ou seja, o conjunto formado por todos os elementos que pertencem simultaneamente a c_1 e c_2.

2 Os tipos grafo e ramo

Um grafo rotulado é um tipo estruturado constituído por um conjunto de nós e um conjunto de ramos entre os nós. Cada ramo tem um rótulo associado. Por exemplo, o grafo abaixo



é constituído pelo conjunto de nós

$$\{A, B, C, D, E, F\},$$

e pelo conjunto de ramos

$$\{(A\ B\ 7), (A\ C\ 8), (A\ F\ 14), (B\ C\ 10), (B\ D\ 15), (C\ D\ 11), (C\ F\ 2), (D\ E\ 6), (E\ F\ 9)\}.$$

Embora em geral os nós possam ter informação associada, consideramos que um nó é equivalente ao seu identificador, podendo este ser de qualquer tipo.[3]

2.1 O tipo ramo

Um ramo é constituído por um conjunto de dois nós, os *extremos* do ramo, e por um inteiro, o *rótulo* do ramo. O tipo ramo deve disponibilizar as seguintes operações básicas.

- Construtor.
  - $faz\text{-}ramo : nó \times nó \times inteiro \mapsto ramo$
    $faz\text{-}ramo(n_1, n_2, r)$ devolve o ramo com extremos n_1 e n_2, e rótulo r.
- Selectores.
  - $extremos\text{-}ramo : ramo \mapsto conjunto\ de\ nós$
    $extremos\text{-}ramo(r)$ devolve o *conjunto* contendo os extremos do ramo r.
  - $rotulo\text{-}ramo : ramo \mapsto inteiro$
    $rotulo\text{-}ramo(r)$ devolve o rótulo do ramo r.
  - $outro\text{-}no\text{-}ramo : nó \times ramo \mapsto nó$
    $outro\text{-}no\text{-}ramo(n, r)$, sendo n um dos extremos do ramo r, devolve o outro extremo do ramo r.
- Reconhecedor.
  - $ramo? : universal \mapsto lógico$
    $ramo?(u)$ tem o valor $verdadeiro$ se u é um ramo, e tem o valor $falso$, em caso contrário.
- Testes.
  - $ramos{=}? : ramo \times ramo \mapsto lógico$
    $ramos{=}?(r_1, r_2)$ tem o valor $verdadeiro$ se r_1 e r_2 são ramos iguais, e tem o valor $falso$, em caso contrário. Dois ramos são considerados iguais se tiverem os mesmos extremos e o mesmo rótulo. Note que não existe uma ordem entre os extremos, pelo que se deverá verificar o seguinte axioma:
    $ramos{=}?(faz\text{-}ramo(n_1, n_2, r), faz\text{-}ramo(n_2, n_1, r)) = verdadeiro$
  - $extremo\text{-}ramo? : nó \times ramo \mapsto lógico$
    $extremo\text{-}ramo?(n, r)$ tem o valor $verdadeiro$ se n é um dos extremos do ramo r, e tem o valor $falso$, em caso contrário.

[3] Para efeitos de implementação, assuma que os nós podem ser comparados com o procedimento primitivo `equal?`.

2.2 O tipo grafo

O tipo grafo deve disponibilizar as seguintes operações básicas.

- Construtor.
  - $faz\text{-}grafo : conjunto\ de\ nós \times conjunto\ de\ ramos \mapsto grafo$
    $faz\text{-}grafo(n, r)$ devolve o grafo cujo conjunto de nós é n e cujo conjunto de ramos é r.
- Selectores.
  - $nos : grafo \mapsto conjunto\ de\ nós$
    $nos(g)$ devolve o conjunto de nós do grafo g.
  - $ramos : grafo \mapsto conjunto\ de\ ramos$
    $ramos(g)$ devolve o conjunto de ramos do grafo g.
  - $vizinhos : nó \times grafo \mapsto conjunto\ de\ nós$
    $vizinhos(n, g)$ devolve o conjunto de nós de g ligados por um ramo ao nó n. Por exemplo, os vizinhos do nó A do grafo da figura anterior são os nós do conjunto $\{B, C, F\}$.
  - *distancia-entre* $: nó \times nó \times grafo \mapsto inteiro$
    distancia-entre(n_1, n_2, g) devolve o rótulo do ramo de extremos n_1 e n_2 se este ramo existir; em caso contrário, o valor devolvido é indefinido.
- Reconhecedor.
  - $grafo? : universal \mapsto lógico$
    $grafo?(u)$ tem o valor $verdadeiro$ se u é um grafo, e tem o valor $falso$, em caso contrário.
- Teste.
  - $grafos{=}? : grafo \times grafo \mapsto lógico$
    $grafos{=}?(g_1, g_2)$ tem o valor $verdadeiro$ se g_1 e g_2 são grafos iguais, e tem o valor $falso$, em caso contrário.

3 Classificação

A nota da segunda parte do projecto será baseada nos seguintes aspectos:

1. Execução correcta (40%).
2. Facilidade de leitura, nomeadamente abstracção procedimental, abstracção de dados, nomes bem escolhidos, paragrafação correcta, qualidade (e não quantidade) dos comentários[4] (25%).

[4]Todos os comentários do seu programa devem ser feitos utilizando a opção "Comment Out with Semicolons". Programas que utilizem a opção "Comment Out with a Box" ou caracteres acentuados serão penalizados com três valores.

3. Relatório (30%).
4. Estilo de programação (5%).

Os pesos das notas das três partes na nota final do projecto são os seguintes:

1. Primeira parte - 10%.
2. Segunda parte - 30%.
3. Terceira parte - 60%.

4 Condições de realização e prazos

O ficheiro contendo o código deverá conter na primeira linha `#lang scheme`, na segunda linha `(provide (all-defined-out))` e, a partir da terceira linha, em comentário, os números e os nomes dos alunos do grupo, bem como o número do grupo.

O projecto deve ser realizado pelos mesmos grupos da primeira parte.

A segunda parte do projecto deve ser entregue até às 15:00 horas do dia **2 de Dezembro de 2011**, na Reprografia do DEI ou na portaria do Tagus (consoante o campus que corresponda ao grupo do projecto), e deverá constar de um relatório, incluindo uma listagem do código (numa fonte "monospace" como, por exemplo, a fonte "courier"). Um modelo de relatório, que podem/devem adaptar de acordo com as vossas necessidades, será disponibilizado no sistema Fénix. Projectos em atraso serão aceites até ao dia 5 de Dezembro, sendo penalizados com 0.5 valores por cada dia de atraso (Sábado e Domingo contam como dias de atraso). A partir das 15:00 horas do dia 5 de Dezembro de 2011 não se aceitam quaisquer projectos, seja qual for o pretexto. O relatório deve ser entregue dentro de uma capa Roma (à venda na Reprografia do DEI), apresentando visivelmente o número do grupo e número e nome dos seus autores, na capa. Projectos que não sejam entregues nestas condições serão penalizados com três valores.

Para além disto, a submissão do código por via electrónica, *através do sistema Fénix*, é *obrigatória* e deverá ser feita nos mesmos prazos que a entrega do relatório. Se o código e o relatório forem entregues em horas diferentes, o desconto será o correspondente ao da última entrega. O código do projecto deve estar contido num único ficheiro. Se durante o desenvolvimento usaram vários ficheiros devem, no final, antes de submeter o código, colocar todo o código num único ficheiro, chamado `FP1112-parte2-grupo`n`.rkt`, em que n é o número do grupo. Por exemplo, o ficheiro do grupo nº 5 deverá chamar-se `FP1112-parte2-grupo5.rkt`.

Durante o desenvolvimento do programa é importante não se esquecer da *Lei de Murphy*:

1. Todos os problemas são mais difíceis do que parecem;
2. Tudo demora mais tempo do que nós pensamos;
3. Se alguma coisa puder correr mal, ela vai correr mal, na pior das alturas possíveis.

Pode ou não haver uma discussão oral do trabalho e/ou uma demonstração do funcionamento do programa (será decidido caso a caso).

Projectos iguais, ou muito semelhantes, serão considerados cópias. O corpo docente da cadeira será o único juiz do que se considera ou não copiar no projecto.