ESTRUTURAS DE DADOS (SCC0122) – TRABALHO 2015

Prof. Ricardo J. G. B. Campello

Existem várias formas sofisticadas de se implementar computacionalmente um *Dicionário*, estrutura na qual pares de informações associadas quaisquer (p. ex. nome – endereço, palavra inglês – palavra português, etc) são armazenados de forma que uma das informações do par (p. ex. endereço) possa ser acessada através da outra informação (p. ex. nome), esta última servindo como uma *chave* para a busca. Quando não são permitidos múltiplos elementos compartilhando uma mesma chave (p. ex. vários endereços associados a um mesmo nome), denomina-se o dicionário de *Mapa*. Dentre diversas possíveis variantes, o TAD Mapa pode ser definido com as seguintes operações (funções) básicas:

- **search**(k, M): retorna o elemento com chave k do mapa M, caso exista. Caso contrário, retorna elemento nulo.
- **insert**(x, M): insere o elemento x no mapa M e retorna *true*, caso não exista elemento em M com a mesma chave de x. Se já existir elemento em M com a mesma chave de x, retorna *false*.
- **replace**(x, M): Substitui por x o elemento do mapa M com a mesma chave de x, se tal elemento existe em M, retornando *true*. Caso contrário retorna *false*.
- **remove**(k, M): remove e retorna o elemento com chave k, caso este exista no mapa M. Caso contrário retorna elemento nulo.

Uma das possíveis formas de implementação de um mapa (nem sempre eficiente, porém relativamente simples) é aquela que utiliza uma *Árvore Binária de Busca* (ABB). De acordo com a terminologia trabalhada no curso, o elemento a ser manipulado pelo Mapa pode ser um registro do tipo *tipo_elem*, que já contém um campo *chave* que pode ser utilizado pelo mapa para acessar a informação associada (campo *info*). Sua tarefa é implementar um Mapa através de uma ABB. Para tanto:

- Implemente em *linguagem C* uma **Árvore Binária de Busca** na forma *modular* discutida no curso, ou seja, composta de: (i) um arquivo de cabeçalho "ABB.h" (TAD) com os tipos de dados e protótipos das funções exportados para os clientes da ABB; e (ii) um arquivo de implementação "ABB.c" com a implementação das funções descritas no arquivo de cabeçalho e também com as funções e tipos de dados de uso exclusivamente interno (não exportados para o cliente). Implemente a árvore binária de forma **dinâmica** (encadeamento com ponteiros), conforme discutido em aula. Considere que a árvore armazena registros do tipo *tipo_elem* com ambos os campos *chave* e *info* do tipo **string** com no máximo 30 caracteres.
- Implemente em C um **Mapa** na forma modular discutida no curso, ou seja, composto de um arquivo de cabeçalho "Mapa.h" (TAD) e de um arquivo de implementação "Mapa.c". Mapa.h deve importar, via comando "#include", os tipos de dados e protótipos de funções em ABB.h. Por sua vez, Mapa.c (que importa Mapa.h) deve utilizar esses tipos de dados e funções para implementar as suas funções. Não reimplemente partes do código da ABB do item anterior para fazer o mapa, faça chamadas para as funções da árvore (buscar na árvore, inserir na árvore, remover da árvore, ...) que já estarão disponíveis no módulo ABB !
- Faça um programa teste que permita ao usuário, através de um **Menu**, realizar pelo console operações de inserir, remover, substituir e buscar por itens no Mapa, além de uma opção de percurso inter-fixado pela árvore que possibilite exibir ordenadamente ao usuário todas as chaves armazenadas na árvore.
- **Nota 1:** Observe que as funções **insert**, **replace** e **remove** do TAD Mapa devem executar uma busca no mapa antes de realizarem as respectivas operações. A função **search** do próprio TAD, no entanto, não é apropriada nesses casos, pois retorna apenas o elemento e não a sua localização na árvore. Para essa finalidade, implemente uma função de **busca auxiliar** que retorne um *ponteiro* para o nodo do respectivo elemento na árvore dinâmica.
- **Nota 2:** Tome como referência os exemplos do capítulo 9 do livro *Introdução a Estruturas de Dados - com Técnicas de Programação em C*, W. Celes, R. Cerqueira & J. L. Rangel, Ed. Campus, 2004 (vide material auxiliar no STOA), para fazer a modularização do código de forma correta.

Continua...

OBSERVAÇÕES:

- Os trabalhos deverão ser feitos em **duplas** e devidamente identificados com os nomes e números USP de cada um dos alunos. Trabalhos individuais ou com mais de 2 alunos **não serão aceitos**.
- É imprescindível explicar/documentar tudo o que está sendo apresentado. Utilize texto e/ou diagramas e/ou figuras e/ou exemplos para descrever o que foi feito e como foi feito de forma clara e didática: Trabalhos compostos basicamente de código **não serão considerados**.
- A avaliação abrangerá aspectos como exatidão, qualidade do conteúdo e do documento, esforço e participação de ambos os alunos (a ser averiguada via apresentação e arguição oral).
- Serão considerados como **não entregues** aqueles trabalhos nos quais forem detectados quaisquer tipos de cópia ou plágio, não importa a origem.
- Os relatórios deverão ser entregues na forma **impressa** para o **estagiário PAE**, junto a uma cópia eletrônica dos códigos **fonte**. No momento da entrega, a dupla deverá mostrar ao estagiário que o código fonte ao menos compila e executa. Testes detalhados do programa e análise do código serão feitos posteriormente pelo estagiário PAE. **Consultem o estagiário** no horário de atendimento para se certificarem que seu código está funcionando corretamente, antes de entregá-lo!
- A data limite para entrega é **19/11/2015**.
- A apresentação e arguição oral de cada dupla serão feitas nos dias **25/11/2015** (quarta-feira, no local / horário de atendimento PAE) e **26/11/2015** (quinta-feira, no local / horário de aula). A escolha da data e horário exato de apresentação/arguição será realizada com o estagiário PAE no momento da entrega do trabalho. A preferência de escolha será dada às duplas por ordem de entrega dos trabalhos.

Bom trabalho!