FEUP

Licenciatura em Engenharia Electrotécnica e de Computadores
Algoritmos e Estruturas de Dados
Miniteste nº 2
Versão para alunos com frequência regular

2º ano
2005-06-09
Duração: 1H30m
Com consulta

Atenção: Este miniteste é composto por 4 questões em 3 páginas.
▷ A pergunta **4** deve ser respondida directamente no enunciado. Não se esqueça de colocar o nome nessa folha e de a entregar juntamente com as restantes respostas.

(5 val.) 1. Pretende-se implementar uma estrutura do tipo lista, mas em que seja possível determinar o menor elemento em tempo constante. Considere que para tal é usada a classe `Listam`, parcialmente definida a seguir:

```
class Erro {};
template <class T>
class Listam {
        VList <T> elementos;
        T menor;    // contém sempre uma cópia do menor elemento da lista
public:  ....
};
```

Assuma que o tipo T implementa os operadores de comparação.

(a) Implemente o membro-função:

```
template <class T> T  Listam<T>::getMenor() const;
```

Este membro-função retorna o menor elemento da lista (operação realizada em tempo constante). Se a lista estiver vazia, deve lançar uma excepção do tipo `Erro`.

(b) Implemente o membro-função:

```
template <class T> void Listam<T>::inserir(const T &el1, int pos=0);
```

Este membro-função insere o novo valor `el1` na lista, na posição `pos`.

(c) Implemente o membro-função:

```
template <class T> void Listam<T>::remover(const T &el1);
```

Este membro-função remove o valor `el1` da lista.

(5 val.) 2. Uma árvore binária de mínimo é aquela em que todos os nós, excepto a raiz, verificam a seguinte propriedade: *O valor do pai do nó X é menor ou igual ao valor de X.*

Considere a classe `BinaryTree<T>` aumentada com dois novos membros-função, um público e outro privado, conforme indicado no código apresentado a seguir:

```
template <class T> class BinaryTree {
 public:
  // ... conforme versão original
  bool isMinTree() const { return root == 0 ? true : isMinTree(root) ; }
   private:
  // ... conforme versão original
   bool isMinTree(const BTNode<T> *t) const;
};
```

(a) Desenhe as árvores binárias `bt` e `bt1` produzidas pelo código indicado a seguir. Indique se `bt` e `bt1` são ou não árvores binárias de mínimo. Justifique todas as respostas.

```
BinaryTree<char> bt1 ('a', BinaryTree<char> ('b', 'c', 'f'),  'd');
BinaryTree<char> bt ('k', bt1, BinaryTree<char>('x','y', 'z'));
```

(b) Implemente a função-membro

```
template <class T> bool BinaryTree<T>::isMinTree(const BTNode<T> *t) const;
```

(5 val.) 3. Pretende-se implementar um programa para fazer substituições rápidas de palavras em ficheiros de texto. O programa usa uma tabela de dispersão que contém pares de palavras: a palavra a ser substituída e a palavra substituta. A classe `ParPal` é usada para representar pares de palavras, conforme indicado a seguir:

```
struct ParPal {
  string texto1 /* palavra original */, texto2 /* palavra substituta */;
  bool operator !=(const ParPal &p) const
  { return texto1 != p.texto1; }
};
```

(a) Defina uma função de dispersão apropriada.

(b) Implemente a função

```
int fillTable(istream &is, HashTable<ParPal> &htp);
```

que preenche a tabela a partir de um ficheiro de texto que contém duas palavras por linha; a segunda palavra é a substituta da primeira. A função deve retornar o número de pares lidos. Para uma dada palavra, apenas deve ser considerada a primeira substituição especificada no ficheiro.

(c) Implemente a função

```
int textSubst(istream &is, ostream &os, const HashTable<ParPal> &htp);
```

que escreve no fluxo de saída as palavras que obtém do fluxo de entrada, fazendo as substituições especificadas por `htp`. A função deve retornar o número de substituições efectuadas.

Nome: _____

Responda directamente sobre o enunciado e entregue este folha com as restantes respostas. Não se esqueça de preencher o nome.

Versão ♠

(5 val.) 4. As questões seguintes têm resposta Verdadeiro/Falso (uma resposta errada tem cotação negativa de valor igual a 50 % de uma resposta certa; a cotação mínima é 0).

(a) Numa fila, cuja implementação é baseada em listas, é possível remover um qualquer elemento da fila. ☐

(b) Numa pilha, cuja implementação é baseada em listas, a operação de inserção de um novo elemento é sempre realizada em tempo constante. ☐

(c) A visita por nível a uma árvore binária de pesquisa, origina sempre uma sequência de valores crescente. ☐

(d) A profundidade de uma árvore binária com 6 nós é sempre inferior a 3. ☐

(e) Em uma árvore binária, o número de folhas é sempre superior ou igual ao número de nós que não são folhas. ☐

(f) Uma sequência de valores correspondete a uma visita em pré-ordem a uma árvore binária, não define uma árvore única. ☐

(g) Uma tabela de dispersão com resolução de colisões por sondagem linear garante que se encontra sempre uma posição livre na inserção de um novo elemento (desde que a tabela não esteja cheia). ☐

(h) Uma tabela de dispersão com resolução de colisões por sondagem quadrática tem sempre um factor de carga inferior a 1.0. ☐

Fim.