Centro Universitário Barão de Mauá

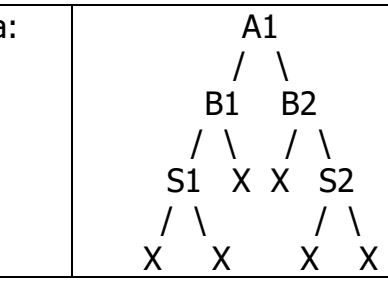
Estrutura de Dados
Laboratório de Estrutura de Dados

Trabalho II

O trabalho consiste em implementar uma árvore de decisão em C++.

O programa deverá ler os dados de entrada a partir de um arquivo com extensão *.tree*, no formato descrito a seguir, que permitirá a construção da árvore. Os dados dos arquivos *.tree* permitirão a construção da árvore a partir da raiz em direção às folhas. Cada linha do arquivo possui uma tripla de strings separadas por espaços e terminadas com o caractere de final de linha (chave1 chave2 chave3 \n).

Exemplo de uma árvore, criada a partir de um arquivo:

A1 B1 B2 B1 S1 X B2 X S2 S1 X X S2 X X X X X	Gera a árvore binária:	

Cada linha do arquivo é uma tripla de dados, separados por espaço em branco, contendo o dado do vértice (chave). A chave pode ser um número, ou um texto, como no exemplo (“A1”). Uma tripla que contém uma chave (do vértice a ser inserido), seguido de duas outras chaves que são, respectivamente, as chaves da esquerda, e a chave da direita. Existem chaves especiais, que são os chaves do tipo “X” e que denotam um vértice vazio (sem filhos naquela sub-árvore). Portanto, um vértice do tipo “C1 X X” é um nó folha, pois não tem nem um filho a esquerda e nem um filho a direita.

Portanto temos: chave1(Pai) chave2(FilhoEsq) chave3(FilhoDir). Se as chaves forem “X”, significa que não temos um filho na respectiva sub árvore onde está indicado o “X”. A última linha do arquivo contém 3 chaves “X” separados por espaço, indicando que não é para inserir mais vértices na árvore (fim da entrada de dados). Assuma que os valores informados no arquivo estão sintaticamente e estruturalmente corretos, codificando uma árvore sem erros e sem ciclos.

Processo de criação e inserção de vértices na árvore:

i) Se a árvore estiver vazia: a primeira linha do arquivo é usada para criar o nó raiz e seus filhos.

ii) Se a árvore já tiver nós, deve ser buscado de forma recursiva a chave do nó em questão que será inserido. Por exemplo na linha 2 do exemplo acima, deve-se buscar recursivamente na árvore o nó B1. Uma vez encontrado este nó, deve então ser inseridos seus 2 filhos na estrutura. No exemplo da linha 2: busca por B1. Ao encontrar B1, insere S1 como filho à esquerda e indica que não há filho à direita (devido ao X).

iii) Repete este processo de busca e construção da árvore, até que encontre o identificador "X X X".

Assim que a estrutura for construída, será necessário imprimir na tela algumas informações sobre a mesma. **O processo de visitação dos nós e exibição da saída a seguir deve ser recursivo**.

As informações solicitadas serão obtidas a partir do resultado do percurso em pré-ordem, indicando o número de filhos que o nó tem e se o nó tem filhos só a esquerda, só a direita, em ambos os lados (direita e esquerda) ou se é nó folha.

Veja a saída esperada para o exemplo apresentado previamente:

```
A1 2 ED
B1 1 E
S1 0 F
B2 1 D
S2 0 F
5 2
```

A saída indica portanto, o conteúdo do nó (string, por exemplo "A1", "S1", ...), seguida de um número inteiro que indica o número de filhos (0 filhos é folha; 1, só um filho; 2 dois filhos), sempre separados por um espaço em branco. Cada nó da árvore é exibido em uma linha da saída (com "\n" após cada linha com os dados do nó). Quando o nó não tem filhos, exibe a seguir a letra "F" indicando que é folha. Quando o nó tem 2 filhos, exibe a seguir as letras "ED" indicando que tem filhos tanto na esquerda quanto na direita. Quando o nó tem 1 filho apenas, indica na letra qual dos lados que está este

filho, “E” se só tem filho à esquerda e “D” se só tem filho à direita. **Lembrete: esta exibição deve ser implementada de modo recursivo.**

Por fim, na última linha de saída na tela é escrito o número total de nós que a árvore possui, onde no caso do exemplo acima são 5 nós, e o número de nós sem filhos, considerados como nós de saída – S1 e S2 (escreve o valor 5 seguido de 2 e do “\n”)

Atenção!

- Um mesmo trabalho poderá ser feito por **até 3** estudantes matriculados na disciplina;
- **Entrega: até as 23:59h do dia 19/11/2023 contendo todos os códigos fontes devidamente documentados, via Portal (SAV da disciplina LED);**
- Não serão aceitos trabalhos e modificações após a data e horário de entrega;
- **Não serão aceitos trabalhos que utilizem *containers* (*vector, queue, stack, priority_queue, list, set, map...*), bem como declarações de variáveis/ponteiros com auto type.**
- Não serão aceitos códigos em outra linguagem de programação;
- Todas as estruturas utilizadas no trabalho devem ser alocadas dinamicamente!
- O código deve ser organizado usando divisão das classes entre arquivos de cabeçalho (.h) e implementação (.cpp);
- Obviamente seu programa deverá ser bem documentado, e a forma de utilizá-lo também será avaliada.
- Os programas resultantes do trabalho poderão serem testados na presença do professor durante o horário da aula de LED seguinte à data de entrega;
- Todo arquivo de código fonte deve conter, nas suas primeiras linhas, um campo de comentário com o nome e o número institucional dos responsáveis pelo trabalho;
- Trabalhos reconhecidos como ‘muito semelhantes’ pela sua estrutura de programação serão desconsiderados. Lembrem-se, variáveis com nomes diferentes, mas em códigos com a mesma estrutura, são considerados ‘muito semelhantes’;
- Ressalto que estas propostas de trabalhos podem envolver alguns conhecimentos da linguagem de programação C++ que não foram

cobertos pelos exemplos ou pelos exercícios realizados em aula. No entanto, estes conhecimentos estão disponíveis nos livros referenciados como material de apoio na ementa da disciplina.

Bom trabalho!