Atividade de Árvore Binária

Observação: As instruções sobre entrega, prazo e avaliação estão nas páginas 2-3 do documento.

Objetivo: Implementar uma árvore binária em C++.

(1) Crie uma classe C++ para o nó da árvore binária. O nó deve armazenar apenas uma string, além dos ponteiros para outros nós (atributos privados).

Essa classe deve conter as seguintes operações (métodos públicos):

- Métodos *getters*/*setters* dos atributos privados do nó (dados e ponteiros dos nós);
- Um método que verifica se o nó é raiz (true se é raiz, false caso contrário);
- Um método que verifica se o nó é folha (true se é folha, false caso contrário);
- Um método que retorna o grau do nó (int);
- Um método que retorna o nível do nó (int);
- Um método que retorna a altura do nó (int).

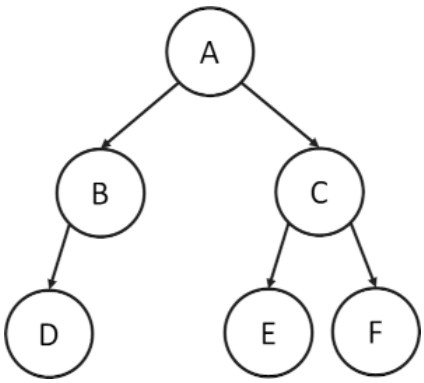
(2) Crie uma classe C++ para a árvore binária. Essa classe deve ter um ponteiro para a raiz da árvore (atributo privado).

Essa classe deve conter as seguintes operações (métodos públicos):

- Métodos *getters*/*setters* dos atributos privados da árvore binária.
- Um método que verifica se a árvore está vazia (true se vazia, false caso contrário);
- Um método que verifica o grau da árvore (int);
- Um método que verifica a altura da árvore (int);
- Um método que percorre a árvore em ordem;
- Um método que percorre a árvore em pré-ordem;
- Um método que percorre a árvore em pós-ordem.
- (Bônus opcional – nota extra) Um método que percorre a árvore por nível.

"Percorrer uma árvore", nessa atividade, significa visitar os nós da árvore e exibir o conteúdo de cada nó no terminal (ou seja, exibir o conteúdo da árvore). Você pode fazer isso usando std::cout já quando visita um nó ou criando uma string com std::ostringstream, para depois exibir a string final com std::cout.

(3) Para testar o seu código, construa a seguinte árvore na memória e use todos os métodos implementados para validação.



(4) A sua main deve ter código que exiba todas as informações de cada nó (se é raiz, se é folha, grau, nível e altura) e todas as informações da árvore (se está vazia, grau e altura da árvore, percurso em ordem, percurso em pré-ordem e percurso em pós-ordem).

Desenvolvimento (10,0 pontos)

- Sua solução deve ser escrita apenas com a linguagem C++, sem uso da STL.
- Declare as suas classes em headers (arquivos .h) e implemente-as nos arquivos .cpp.
- Lembre-se do include guard do header.
- Para alocação dinâmica, use new e delete (lembre-se que, para arrays, o delete inclui o [] antes do nome do ponteiro a ser liberado).
- A correta execução do código da main também entra como critério de avaliação (haverá desconto na nota caso falte exibir as informações pedidas no item 4 do enunciado).

Identificação e referências

- Coloque sua identificação - nome e TIA - no início de cada arquivo de código, como comentário (use // no começo de cada linha que queira comentar).
- Inclua como comentário quaisquer referências (livros, artigos, sites, entre outros) usadas para solucionar o problema.

Entrega

- **Código:** Compacte todos os arquivos .cpp/.h ou o projeto completo criado na IDE que você está usando (mas sem os intermediários como bin e obj) no formato zip OU comite todos os arquivos .cpp/.h ou o projeto completo criado na IDE que você está usando (mas sem os intermediários como bin e obj) em um repositório git.
- **Arquivo texto (.txt):**
  - Se o código está em um repositório git, envie um arquivo txt no Moodle contendo sua identificação e o link do repositório.
- **Prazo de entrega:** via link do Moodle até 26/09/2021 23:59.

Informações importantes sobre critérios de avaliação

Embora essa atividade seja uma avaliação da disciplina, sempre considero que as atividades também podem ser usadas para nos acostumarmos com o mercado de trabalho. Portanto, leve em consideração os seguintes critérios que vou aplicar na avaliação:

- Será descontado 1,0 (um) ponto caso a entrega não respeite o enunciado. Exemplos:
  - O enunciado pede para enviar um arquivo compactado no formato zip, mas o arquivo enviado está no formato rar.
  - O enunciado pede um arquivo texto no formato txt, mas foi enviado um documento do Word.
  - Não há identificação nem referências (caso aplicável) nos arquivos de código.
- Será descontado 1,0 (um) ponto caso o arquivo zip OU o repositório git contenha pastas e arquivos desnecessários.

Exemplo:
  - Pastas intermediárias criadas no processo de compilação (Debug, obj, bin, ...).
- O projeto deve ser desenvolvido em linguagem C++ e não em linguagem C. Caso a solução apresentada use funcionalidades da linguagem C e que tenham equivalentes em C++, será descontado 2,0 (dois) pontos.
  Atente-se a esse detalhe quando estiver pesquisando e verificando exemplos na internet e outros materiais, principalmente de assuntos que não vimos até o momento (essa atividade pode ser resolvida só com o que foi visto em aula, com suas devidas adaptações).
  Exemplo:
  - Declarar arrays de tamanho variável (padronizado no C99, mas erro em C++ pois não há suporte para VLA), ex. int n = 10; char arr[n];.
- Projeto que possui erros de compilação ou que trava durante a execução automaticamente perde 50% da nota máxima.
  Sobre erros de compilação: considero apenas erros, não há problema se o projeto tiver warnings (apesar que warnings podem avisar possíveis travamentos em tempo de execução, como loop infinito, divisão por zero etc.).
  Quando há necessidade de entrada de dados por parte do usuário, assumo que o usuário vai inserir as informações corretas (ex. tipos de dados corretos), a menos que o enunciado explicite que você deve garantir que os dados de entrada estejam corretos.

Em uma situação profissional, os itens indicados acima atrapalham (e muito) o trabalho da equipe. E o último item é gravíssimo (o ideal também é remover todos os warnings e sempre validar os dados).