UNIVERSIDADE ABERTA
www.uab.pt

e-Fólio B

U.C. 21046

Estruturas de Dados e Algoritmos Fundamentais

2025-2026

INSTRUÇÕES

1. Este e-fólio tem uma cotação global de 5 valores.

2. Este e-fólio deve ser submetido em dois locais distintos:

    - A resolução do programa é realizada na plataforma moodle, no recurso VIRTUAL PROGRAMMING LAB e-fólio B, disponibilizado no espaço central da unidade curricular.

    - A resolução deve ainda ser acompanhada por um relatório constituído por um único ficheiro em **formato pdf,** entregue única e exclusivamente através do recurso "E-Fólio B", da turma respetiva.

3. O nome do ficheiro PDF deve seguir a norma "eFolioB" + <nº estudante> + <primeiro nome> + <último nome>.

4. Não são aceites e-fólios manuscritos, i.e. tem penalização de 100%.

5. Na primeira página do e-fólio deve constar o nome completo do estudante bem como o seu número. Penalização de 10% a 100%.

6. Durante a realização do e-fólio, os estudantes devem concentrar-se na resolução do seu trabalho individual, não sendo permitida a colocação de perguntas ao professor ou entre colegas.

7. A interpretação das perguntas também faz parte da sua resolução, se encontrar alguma ambiguidade deve indicar claramente como foi resolvida.

ENUNCIADO

Pretende-se desenvolver um programa em linguagem C++11 padrão que aceite comandos para a gestão de uma árvore de pesquisa binária (BST – Binary Search Tree) para armazenar itens que são números inteiros (positivos ou negativos). Neste caso os itens representam ambos os papéis de chave e de informação.

Os comandos devem permitir inserir, remover, procurar e listar itens na árvore, além de outros comandos mais específicos.

A árvore é inicializada por defeito vazia e é alterada com comandos especificados num ficheiro de entrada fornecido ao programa pela entrada padrão (stdin em C e cin em C++), um comando por linha, podendo um comando ter vários argumentos, com o seguinte formato,

```
cmd    arg0  arg1  ...
```

onde "cmd" indica o nome do comando a executar, "arg" é um tipo de argumento do comando e "..." a seguir a um argumento indica que este pode ser repetido várias vezes indefinidamente. Na descrição dos comandos, um tipo de argumento pode ser representado por: "item" o item a armazenar (inteiro).

A lista dos comandos a implementar é a indicada a seguir. No caso de mensagens de saída de dados, o seu formato é indicado em estilo da linguagem C.

Operações principais

Comando	Descrição
insert item ...	Insere itens na BST pela ordem apresentada. Itens repetidos não devem ser inseridos; se um item já existir, esse item é ignorado e o comando prossegue.
print_in	Imprime os itens da árvore em travessia inorder. Formato "BST(in)= %d %d ...\n".
print_pre	Imprime os itens da árvore em travessia preorder. Formato "BST(pre)= %d %d ...\n".
print_post	Imprime os itens da árvore em travessia postorder. Formato "BST(post)= %d %d ...\n".
print_min	Imprime o menor item da árvore. Formato "Min= %d\n".
print_max	Imprime o maior item da árvore. Formato "Max= %d\n".

Consultas e atualização da árvore

Comando	Descrição
dim	Imprime o número de itens da árvore. Formato "BST tem %d itens\n".
height	Imprime a altura da árvore. Formato "Altura= %d\n". Considere que a árvore vazia tem altura -1 e uma árvore com apenas a raiz tem altura 0.
find item	Procura o item na árvore. Se existir, imprime "Item %d encontrado\n". Se não existir, imprime "Item %d nao encontrado!\n".
count_leaves	Imprime o número de folhas da árvore. Formato "BST tem %d folhas\n".
delete item	Remove o item utilizando obrigatoriamente o algoritmo de remoção por cópia (deletion by copying). Se o item não existir, imprime "Comando %s: Item nao encontrado!\n".
clear	Remove todos os nós da árvore, inicializando novamente a BST como vazia.

Mensagens de erro e comportamento esperado

Todos os comandos que não possam ser executados por a BST estar vazia devem imprimir uma mensagem com o formato "Comando %s: BST vazia!\n".

Esta condição aplica-se aos comandos print_in, print_pre, print_post, print_min, print_max, height, count_leaves, delete e clear. O comando find item, se executado numa árvore vazia, deve imprimir "Item %d nao encontrado!\n".

É da responsabilidade do programa assegurar que nenhuma operação que comprometa a estabilidade do programa é executada, por exemplo tentar aceder ao mínimo de uma BST vazia.

P

Requisitos de implementação

Projete as estruturas de dados (classes) adequadas ao programa que se pretende desenvolver. No que respeita aos atributos obrigatórios, a BST deve possuir um apontador para a raiz e um contador que a cada momento indica quantos nós tem a árvore. Todos os nós e árvores utilizados no programa devem ser instâncias de classes/estruturas (objetos) e não um conjunto de variáveis "soltas" ou isoladas.

No desenvolvimento do programa deve utilizar os algoritmos próprios para cada operação efetuada.

Os métodos (funções membro) são livres. Deverá definir e criar os métodos e construtores que considerar mais adequados. O código dos métodos deve ser definido fora das classes, que apenas devem ter os respetivos protótipos. Os métodos e os comandos devem ser implementados também tendo em conta a sua eficiência.

No recurso VPL encontram-se templates de 3 ficheiros para o desenvolvimento do programa com instruções, às quais deve ser acrescentado o código necessário. Para cada caso de teste, o programa é executado com um ficheiro de entrada de dados redirecionado para a entrada padrão, sendo equivalente ao seguinte exemplo na linha de comandos, onde < é o operador de redireção da entrada padrão:

```
$ prog < input.txt
```

sendo posteriormente os dados de saída do programa comparados com a saída esperada.

Leitura da entrada

O ficheiro de entrada (fornecido automaticamente pela entrada padrão) pode ter linhas em branco e o nº de espaços que separa o comando e os argumentos entre si pode ser qualquer. Também podem existir linhas de comentário, caso em que começam obrigatoriamente pelo caracter '#' na 1ª posição.

Projete e teste uma versão do programa que implemente as especificações e comandos pedidos. O programa deve ler e processar o ficheiro de entrada uma linha inteira de cada vez, num ciclo ler-linha processar-linha até chegar ao fim do ficheiro de entrada. É estritamente proibido ler o ficheiro de entrada todo de uma vez para memória.

É importante saber ler o ficheiro de entrada corretamente e eficientemente sem usar código complexo. Sugere-se que seja lida uma linha de cada vez para uma string e

analisado o primeiro carater para ver se é um '#', caso em que a linha é um comentário. Caso não seja comentário, criar uma string stream e usar o operador extrator >> para obter o nome do comando. Terá sucesso se não for uma linha em branco. Após saber o nome do comando, sabe-se quais os tipos de argumentos do comando que se seguem.

Programas que processem uma linha de entrada carater a carater serão severamente penalizados devido à sua complexidade.

Restrições

No desenvolvimento do programa em C++11 não devem ser utilizadas variáveis globais nem ser utilizada a STL, nomeadamente no que respeita às estruturas de dados e algoritmos estudados, devendo o aluno escrever o próprio código.

Restrições aplicam-se aos includes <array> <deque> <forward_list> <list> <map> <queue> <set> <stack> <unordered_map> <unordered_set> <vector> <utility> e em parte de <algorithm>. Em caso de dúvida, questionar o seu uso. Não existem restrições para <string>. Também não devem ser usados smart pointers.

Exemplo de dados entrada / saída padrão

```
Input:
# exemplo
insert 6 11 9 2 10 13 7 1 4 8
print_in
print_pre
print_post
print_min
print_max
dim
height
count_leaves

Output:
BST(in)= 1 2 4 6 7 8 9 10 11 13
BST(pre)= 6 2 1 4 11 9 7 8 10 13
BST(post)= 1 4 2 8 7 10 9 13 11 6
Min= 1
Max= 13
BST tem 10 itens
Altura= 4
BST tem 5 folhas
```

Critérios de correção

- O programa desenvolvido difere significativamente das especificações e instruções do enunciado => 0 valores.

- O programa utiliza variáveis globais ou componentes da STL não permitidas no enunciado do VPL => 0 valores.

- O programa altera o nome das variáveis e classes definidas nas templates de código do VPL => 0 valores.

- O programa não compila ou produz avisos com g++ -Wall -std=c++11 => 0 valores.

- O código do programa não está correta e uniformemente indentado de modo a permitir a sua leitura fácil => 0 valores.

- O programa não está comentado => 0 valores. Os comentários no programa devem elucidar questões relevantes do código locais ao comentário e não o funcionamento geral do programa, que deve ser explicado no relatório.

- O programa é avaliado tendo como ponto de partida a percentagem de testes com resultado positivo. O nível de simplicidade e qualidade do código também é avaliado. Programas considerados mal estruturados, demasiado complexos, confusos ou ineficientes podem ser penalizados até 50%.

- Apenas são considerados os resultados e o código dos programas constantes no recurso VPL.

- **O e-fólio só é considerado entregue com a submissão do relatório em formato pdf no respetivo recurso no espaço turma**. O relatório tem formato livre, com um máximo de 2 páginas úteis além da capa/cabeçalho, com letra mínima 11. **Deve explicar sucintamente o funcionamento do programa, os pontos principais e opções tomadas, e indicar a complexidade assintótica das operações** insert, find, print_in, delete, height e count_leaves.

- Trabalhos com plágio comprovado ou muito similares entre si, independentemente da origem, serão anulados.

Nota ética: Nunca é de mais referir que o código a apresentar como solução para este e-fólio deve ser 100% original do aluno. A probabilidade de duas pessoas que efetivamente não comunicaram entre si apresentarem programas "quase iguais" é considerada nula. Isto é válido para qualquer par de alunos, assim como entre um aluno e qualquer outra pessoa, em particular através da Internet, onde existem inúmeras soluções e código para os mais variados problemas, em sites, fóruns, blogs, IA, etc.

Bom trabalho!

FIM