Documentação - Projeto EDA

João Pedro Soares Matias

May 2023

1 Classes e Funções

Implementação de **4** Classes templatezadas: Node, Pessoa, AVL e Date.

1.1 Node.h

A Classe é a mesma feita durante todo o período em que estudamos AVLs com exceção de uma única modificação que além do nó receber chave(**key**), altura (**height**), ponteiro para direita e esquerda (***left** e ***right**, respectivamente), o nó irá receber um atributo ponteiro para pessoa (***pes**), que me auxiliará a retornar a pessoa tratada nas consultas posteriores. Meu construtor é do tipo const passando **key** e ***pes** como atributos, sendo os outros passados por default.

1.2 Pessoa.h

A classe Pessoa possui 5 atributos, sendo eles: string nome, string sobrenome, string cidade, string cpf, Date data; Um construtor default e um construtor passando os 5 atributos como parâmetros, além de uma função print que printa os atributos separados por vírgula.

1.3 Date.h

Minha Classe Date recebe como atributos inteiros, **year**, **month** e **day**, representando respectivamente, ano, mês e dia. Com um Construtor que recebe os três inteiros como parâmetro, um Construtor default, e um Construtor que recebe uma data no formato de string e distribui aos atributos no tipo int Recebendo datas com intervalos de '/' e '.' como 2/2/2000 e 2.2.2000.

Sobrecarga dos operadores $>, <, >=, <=, ==, !=, <<, >>$.

E uma função isBetween que retorna true caso a minha data esteja entre duas datas passadas como parâmetro.

Obs: Tratei uma data como sendo maior que a outra pela idade da Pessoa, pessoas mais velhas tem datas maiores que pessoas mais novas que ela.

1.4 AVL.h

Minha Classe AVL é a mesma fornecida pelo senhor durante a cadeira com algumas funções extras, possui um construtor default.

Funções Públicas:

- Add para adicionar um nó na árvore.
- bshow para print da árvore.
- clear para deletar todos os nós da árvore.
- destrutor.
- searchNameCpf para consulta tanto de nomes quanto de cpf único.
- searchDate para consulta de intervalo de datas.
- *minimum para receber o minimo nó de uma árvore.
- *maximum para receber o máximo nó de uma árvore.

Além de um ponteiro público pra root da árvore.

Funções Privadas:

- rightRotation para rotação a direita.
- leftRotation para rotação a esquerda.
- add privada.
- fixup-node para balanceamento da árvore.
- bshow privada.
- clear privada.
- height para calcular a altura.
- balance para checar o balanceamento da árvore.

Falando sobre a Main eu fiz basicamente toda minha interface nela, além de também ler o arquivo e add nas árvores nela também, qualquer dúvida sobre alguma função estará explicada em forma de comentário no projeto.

código de compilação: **g++ *.cpp -o main ./main**