Vinícius Machado da Rocha Viana - 2111343
João Ricardo Malta de Oliveira - 2112714

Struct no

é uma struct que representa o nó de uma árvore. Ele possui 3 atributos: uma chave (valor inteiro) que representa o valor desse nó e 2 ponteiros para nós, que representam seus filhos na esquerda e na direita.

Funções:

No *cria(int c, No *p1, No *p2)

A função "cria" tem a finalidade de criar um novo nó para a árvore binária. Ela recebe três argumentos:
"c" (int): Representa o valor da chave que será armazenada no novo nó.
"p1" (No *): É um ponteiro para o filho esquerdo do novo nó.
"p2" (No *): É um ponteiro para o filho direito do novo nó.

Dentro da função fizemos a alocação dinâmica de memória usando a função "malloc(sizeof(No))". Isso cria espaço na memória para armazenar as informações do novo nó. Passamos o valor da chave do novo nó como o valor fornecido em "c", os ponteiros "esq" e "dir" do novo nó para apontar para os nós representados por "p1" e "p2", que atuam como os filhos esquerdo e direito do novo nó e por fim a função retorna um ponteiro para o novo nó, que agora possui a chave e as ligações para os seus filhos, permitindo a construção da estrutura da árvore binária.

No* insereElemento(No* p, int chave)

A função "insereElemento" segue a seguinte lógica, primeiro verificamos se o nó atual "p" é nulo, se for nulo, cria um novo nó com a chave passada como argumento usando a função "cria" e retorna esse novo nó, se o nó atual "p" não for nulo, a função verifica se pelo menos um dos filhos (esquerdo ou direito) do nó atual é nulo. Se um dos filhos for nulo, a função insere o novo nó como filho nulo do nó atual, dependendo de qual filho estiver vazio.
Se ambos os filhos do nó atual não forem nulos, a função chama recursivamente a função "insereElemento" para um dos filhos (esquerdo ou direito) do nó atual, dependendo de qual filho ainda possui pelo menos um filho nulo.
Por fim, a função retorna o nó atual "p" após ter feito as inserções necessárias, mantendo a estrutura da árvore balanceada.

int verificaAbb(No* P)

É uma função que recebe um ponteiro “p” para um nó de uma árvore e retorna 1 ou 0, indicando respectivamente se a árvore é ou não é uma árvore binária de busca (ABB). Primeiro a função faz uma verificação para saber se “p” é nulo, se for ele retorna 1, pois uma árvore vazia pode ser considerada ABB. Caso contrário verifica-se se o filho da esquerda é diferente de nulo e se o valor da esquerda é maior que o valor do nó ou se o filho da direita é diferente de nulo e o valor da direita é menor que o valor do nó.
Se alguma dessas condições for cumprida (nó da direita diferente de nulo e com valor menor ou nó da esquerda diferente de nulo e com valor maior), ele encerra a função retornando 0

Caso nenhuma das duas verificações iniciais tenha sido sucedida, ele então chama a função recursivamente para seus “filhos” da direita e da esquerda e então retorna se ambos os filhos são ABB.

int altura(No *p)

A função "altura" calcula a altura de uma árvore binária. Ela começa verificando se a árvore está vazia, ou seja, se o ponteiro para o nó "p" é nulo. Se for o caso, a altura da árvore é 0. Caso contrário, a função calcula a altura das subárvores esquerda e direita, chamando a função "altura" recursivamente para essas subárvores. A altura da árvore é o máximo entre as alturas dessas subárvores, acrescido de 1, que representa a altura do nó raiz.

void exibePreOrdem(No *p)

A função "exibePreOrdem" imprime os valores dos nós da árvore em pré-ordem. Ela começa verificando se o nó atual (representado por "p") não é nulo. Se o nó não for nulo, a função imprime o valor do nó atual, então chama-se recursivamente a função "exibePreOrdem" para a subárvore esquerda e, em seguida, para a subárvore direita. Essa ordem de impressão significa que a raiz é visitada antes de seus filhos, seguindo o padrão de pré-ordem.

int verificaAVL(No* p)

É uma função que recebe um ponteiro “p” para um nó de uma árvore e retorna 1 ou 0, indicando respectivamente se a árvore é ou não é uma árvore balanceada (AVL)

Primeiro, a função verifica se o ponteiro recebido é nulo, se for ele retorna 1 pois uma árvore vazia pode ser considerada AVL em seguida se a árvore recebida é abb pela função "verificaABB" e se nao for retorna 0.

Se as condições não forem cumpridas, ele salva a altura de cada lado da árvore em duas variáveis (alturaEsq e alturaDir), chamando a função “altura” e passando seus nós esquerdo e direito para cada uma das variáveis.
Em seguida ele faz uma comparação utilizando a função “abs” que retorna o módulo da operação, comparando se a diferença de altura entre os nós esquerdos e direitos são maiores que 1. Se a diferença for maior do que 1, a função é encerrada retornando 0.

Se nenhuma das condições acima for cumprida, ele chama a função recursivamente para seus nós direito e esquerdo, verificando se ambos são AVL. Se algum dos nós não for, ele retorna 0 e a função é encerrada. Caso contrário, ele retorna 1, indicando que a árvore é AVL.

int main(void)

Na primeira linha, inicializa a árvore binária representada por um ponteiro chamado "arvore" como nula, indicando uma árvore vazia.
Em seguida, utilizamos a função "srand(time(0))" para inicializar a semente do gerador de números aleatórios com o valor atual do relógio. Isso garante que os números aleatórios gerados sejam diferentes em cada execução do programa.
Em um loop que se repete 20 vezes, geramos números aleatórios entre 1 e 100 (inclusive) e os armazenamos na variável "numero". Em cada iteração do loop, exibimos o número sorteado no console.
Chamamos a função "insereElemento" para inserir o número sorteado na árvore "arvore".
Após o loop, utilizamos a função "exibePreOrdem" para exibir os valores dos nós da árvore em pré-ordem, o que significa que a raiz é visitada antes de seus filhos esquerdo e direito.
Em seguida, utilizamos a função "verificaAbb" para verificar se a árvore é uma Árvore Binária de Busca (ABB) e imprime o resultado no console e fizemos o mesmo para verificar se é uma arvore avl usando a função "verificaAVL" .