universidade jorge amado

ESTRUTURA DE DADOS

montival alcantara da silva junior

**Entrega de avaliação - Trabalho da Disciplina [AVA2]**

Lauro de Freitas

2020

MONTIVAL ALCANTARA DA SILVA JUNIOR

**Entrega de avaliação - Trabalho da Disciplina [AVA2]**

Versão original

Lauro de Freitas

2020

**BUSCA EM ARVORES BINÁRIAS**

As árvores binárias de busca são estruturas de dados de árvore binária baseada em nós em que todos os nós da subárvore esquerda possuem um valor numérico inferior ao nó raiz e todos os nós da subárvore direita possuem um valor superior ao nó raiz (FORBELLONE; EBERSPACHER, 2005).

A Empresa Renalf S/A solicita a você, membro da equipe de programadores de computador em linguagem C, que desenvolva um programa que seja capaz de manipular uma árvore binária (inclusão e remoção de nós), admitindo a escolha dos três processos de busca em profundidade.

**Procedimentos para elaboração do TD**

Nesse contexto, escreva um programa em linguagem C que apresente um menu de opções que possibilite executar as seguintes opções:

\* \* \* MENU DE OPÇÕES \* \* \*

1. Incluir nó.
2. Remover nó.
3. Buscar pré-ordem.
4. Buscar em ordem.
5. Buscar pós-ordem.

Opção [0 para encerrar]

**RESOLUÇÃO**

struct Valor{

char prefixo[11] ;

};

typedef struct codigo{

int digito;

struct codigo \*direita;

struct codigo \*esquerda;

} Cod;

void criaNovoCodigo(Cod \*\*T){

\*T = NULL;

}

int verificaCodigoNulo(Cod\* T){

return T == NULL;

}

void insereCodigo(Cod \*\*t, int digito){

if(\*t == NULL){

\*t = (Cod\*)malloc(sizeof(Cod));

(\*t)->esquerda = NULL;

(\*t)->direita = NULL;

(\*t)->digito = digito;

}else{

if(digito < (\*t)->digito){

insereCodigo(&(\*t)->esquerda, digito);

}if(digito > (\*t)->digito){

insereCodigo(&(\*t)->direita, digito);

}

}

}

Cod \*PraDireita(Cod \*\*cod){

if((\*cod)->direita != NULL){

return PraDireita(&(\*cod)->direita);

}else{

Cod \*aux = \*cod;

if((\*cod)->esquerda != NULL){

\*cod = (\*cod)->esquerda;

}else{

\*cod = NULL;

return aux;

}

}

}

Cod \*PraEsquerda(Cod \*\*cod){

if((\*cod)->esquerda != NULL){

return PraEsquerda(&(\*cod)->esquerda);

}else{

Cod \*aux = \*cod;

if((\*cod)->direita != NULL){

\*cod = (\*cod)->direita;

}else{

\*cod = NULL;

return aux;

}

}

}

void removeValor(Cod \*\*T, int digito){

if(\*T == NULL){

printf("\n Opa, parece que você digitou uma valor que não existe na árvore!\n");

return;

}else{

if(digito > (\*T)->digito){

removeValor(&(\*T)->direita, digito);

}else{

Cod \*aux = \*T;

if (((\*T)->esquerda == NULL)&&((\*T)->direita == NULL)){

free(aux);

printf("\n Valor removido com sucesso! \n");

(\*T) = NULL;

}else{

if ((\*T)->esquerda == NULL){

(\*T) = (\*T)->direita;

aux->esquerda = NULL;

free(aux);

aux = NULL;

printf("\n Valor removido com sucesso! \n");

}else{

aux = PraDireita(&(\*T)->esquerda);

aux->esquerda = (\*T)->esquerda;

aux->direita = (\*T)->direita;

(\*T)->esquerda = (\*T)->direita = NULL;

free((\*T));

\*T = aux;

aux = NULL;

printf("\n Valor removido com sucesso! \n");

}

}

}

}

}

void imprimePreOrdem(Cod \*\*T){

if((\*T) != NULL){

printf("%i\n", (\*T)->digito);

imprimePreOrdem(&(\*T)->esquerda);

imprimePreOrdem(&(\*T)->direita);

}

}

void imprimeEmOrdem(Cod \*\*T){

if((\*T) != NULL){

imprimeEmOrdem(&(\*T)->esquerda);

printf("%i\n", (\*T)->digito);

imprimeEmOrdem(&(\*T)->direita);

}

}

/\*

\* Função Percurso Pós-Ordem

\*/

void imprimePosOrdem(Cod \*\*T){

if((\*T) != NULL){

imprimePosOrdem(&(\*T)->esquerda);

imprimePosOrdem(&(\*T)->direita);

printf("%i\n", (\*T)->digito);

}

}

/\*

\* Função que verifica o maior parâmetro

\*/

int retornaMaiorValor(int a, int b){

if(a > b)

return a;

else

return b;

}

/\*

\* Função de imprimir Árvore

\*/

void imprimeArvoreSeNaoVazia(Cod \*\*T){

if((\*T) != NULL){

if((\*T) != NULL){

printf("\nRaíz %i\n",(\*T)->digito);

if((\*T)->esquerda != NULL)

printf("esquerda: %i\t",(\*T)->esquerda->digito);

else

printf("esquerda: NULL\t");

if((\*T)->direita != NULL)

printf("direita: %i\t",(\*T)->direita->digito);

else

printf("direita: NULL\t");

if((\*T)->esquerda != NULL)

imprimeArvoreSeNaoVazia(&(\*T)->esquerda);

if((\*T)->direita != NULL)

imprimeArvoreSeNaoVazia(&(\*T)->direita);

}else

printf("Árvore Vazia! \n");

}

}

/\*

\* Início do Programa Principal

\*/

int main(){

struct Valor x;

int c;

Cod \*T;

criaNovoCodigo(&T);

setlocale(LC\_ALL, "portuguese");

int opcao;

do{

system("CLS");

printf("\* \* \* MENU DE OPÇÕES \* \* \*\n\n");

printf("1- Incluir nó: \n");

printf("2- Remover nó: \n");

printf("3- Buscar pré-ordem: \n");

printf("4- Buscar em ordem: \n");

printf("5- Buscar pós-ordem: \n");

printf("6- Imprime Árvore \n");

scanf("%d", &opcao);

switch(opcao){

case 1:

system("CLS");

printf("\nprefixo: ");

scanf("%s", x.prefixo);

printf("\nDigite um Número (Referência na Árvore): ");

scanf("%d",&c);

insereCodigo(&T,c);

system("PAUSE");

break;

case 2:

system("CLS");

printf("\nDigite um Número: ");

scanf("%d",&c);

removeValor(&T,c);

system("PAUSE");

break;

case 3:

system("CLS");

imprimePreOrdem(&T);

system("PAUSE");

break;

case 4:

system("CLS");

imprimeEmOrdem(&T);

system("PAUSE");

break;

case 5:

system("CLS");

imprimePosOrdem(&T);

system("PAUSE");

break;

case 6:

system("CLS");

imprimeArvoreSeNaoVazia(&T);

printf("\n");

system("PAUSE");

break;

case 0:

break;

default:

printf("\n\n Opa, parece que você digitou uma opção inválida \n");

system("PAUSE");

break;

}

}

while(opcao!=0);

return 0;

}