Universidade Federal Fluminense

Matéria Estrutura de Dados

Professor Dalessandro Soares

Alunos: Alessandro Sampaio e Tatiane Sousa

#include <stdlib.h>

#include <stdio.h>

typedef struct arvore{

int info;

struct arvore \*esq;

struct arvore \*dir;

}Arvore;

Arvore\* Ler\_arvore\_arq(FILE\* arq){

char c;

int x;

fscanf(arq,"%c",&c);

fscanf(arq,"%d",&x);

if(x==-1){

fscanf(arq,"%c",&c);

return NULL;

}

else{

Arvore \*no=(Arvore\*)malloc(sizeof(Arvore));

no->info=x;

no->esq=Ler\_arvore\_arq(arq);

no->dir=Ler\_arvore\_arq(arq);

fscanf(arq,"%c",&c);

return no;

}

}

void imprimir\_preordem (Arvore\* a){

if(a!=NULL)

{

printf("%d ",a->info);

imprimir\_preordem(a->esq);

imprimir\_preordem(a->dir);

}

}

void imprimir\_ordem (Arvore\* a){

if(a!=NULL)

{

imprimir\_ordem(a->esq);

printf("%d ",a->info);

imprimir\_ordem(a->dir);

}

}

void imprimir\_posordem (Arvore\* a){

if(a!=NULL)

{

imprimir\_posordem(a->esq);

imprimir\_posordem(a->dir);

printf("%d ",a->info);

}

}

int existe(Arvore \*a, int y){

if (a==NULL)

return 0;

else{

if (a->info==y)

return 1;

else{

if (y<=a->info)

return existe(a->esq,y);

else

return existe(a->dir,y);

}

}

}

Arvore\* remover\_no(Arvore\* a, int y){

if(y<a->info)

a->esq=remover\_no(a->esq,y);

else{

if(y>a->info)

a->dir=remover\_no(a->dir,y);

else{

if((a->esq==NULL) && (a->dir==NULL)){

free(a);

return NULL;

}

else{

if((a->esq==NULL)||(a->dir==NULL)){

Arvore\* aux;

if(a->esq==NULL)

aux=a->dir;

else

aux=a->esq;

free(a);

return aux;

}

else {

Arvore \*aux=a->esq;

while (aux->dir!=NULL)

aux=aux->dir;

a->info=aux->info;

a->esq=remover\_no(a->esq,aux->info);

return a;

}

}

}

}

return a;

}

Arvore\* inserir\_no(Arvore \*a, int y){

if(a==NULL){

Arvore \*no=(Arvore\*)malloc(sizeof(Arvore));

no->info=y;

no->esq=NULL;

no->dir=NULL;

return no;

}

else{

if (y<=a->info)

a->esq=inserir\_no(a->esq,y);

else

a->dir=inserir\_no(a->dir,y);

return a;

}

printf("\n");

}

void imprimir\_menorx(Arvore \*a, int y){

if (a!=NULL){

if(a->info<y){

imprimir\_menorx(a->esq,y);

printf ("%d ",a->info);

imprimir\_menorx(a->dir,y);

}

else{

imprimir\_menorx(a->esq,y);

if(a->info==y)

printf ("%d",a->info);

}

}

}

void imprimir\_maiorx(Arvore \*a, int y){

if (a!=NULL){

if(a->info>y){

imprimir\_maiorx(a->dir,y);

printf ("%d ",a->info);

imprimir\_maiorx(a->esq,y);

}

else{

imprimir\_maiorx(a->dir,y);

if(a->info==y)

printf ("%d",a->info);

}

}

}

void imprimir\_entrexy(Arvore \*a,int w, int y){

if(a!=NULL){

if(a->info> w && a->info < y){

imprimir\_entrexy(a->esq,w,y);

printf("%d ",a->info);

imprimir\_entrexy(a->dir,w,y);

}

else

if(a->info > w && a->info > y)

imprimir\_entrexy(a->esq,w,y);

else

if(a->info < w && a->info < y)

imprimir\_entrexy(a->dir,w,y);

}

}

Arvore\* remover\_arvore(Arvore\* a){

if (a!=NULL){

remover\_arvore(a->esq);

remover\_arvore(a->dir);

free(a);

}

return NULL;

}

int main (void){

FILE\* arq;

arq=fopen("arvore.txt","rt");

if (arq==NULL){

printf ("erro ao abrir o arquivo");

exit (1);

}

int e=1,y,w;

Arvore \*a=NULL;

while (e!=8){

system("cls");

printf ("Digite a opcao desejada\n\n");

printf ("1 - Ler a arvore de um arquivo\n");

printf ("2 - Imprimir\n");

printf ("3 - Inserir um noh\n");

printf ("4 - Remover um noh\n");

printf ("5 - Exite\n");

printf ("6 - Imprimir nohs maiores ou menor que x\n");

printf ("7 - Imprimir nohs entre x e y \n");

printf ("8 - Sair\n");

scanf ("%d",&e);

system("cls");

switch (e)

{

case 1:{

a=Ler\_arvore\_arq(arq);

printf ("A arvore foi lida com sucesso\n");

system("pause");

break;

}

case 2:{

printf ("Digite a opcao desejada\n\n");

printf ("1 - Para imprimir em Pre-ordem\n");

printf ("2 - Para imprimir em Ordem\n");

printf ("3 - Para imprimir em Pos-ordem\n");

scanf ("%d",&y);

system("cls");

switch (y){

case 1:{

printf ("Impressao em pre-ordem\n\n");

imprimir\_preordem(a);

printf (" \n");

system("pause");

break;

}

case 2:{

printf ("Impressao em ordem\n\n");

imprimir\_ordem(a);

printf (" \n");

system("pause");

break;

}

case 3:{

printf ("Impressao em pos-ordem\n\n");

imprimir\_posordem(a);

printf (" \n");

system("pause");

break;

}

}

break;

}

case 3:{

printf ("Digite o noh a ser inserido\n");

scanf ("%d",&y);

a=inserir\_no(a,y);

system("pause");

break;

}

case 4:{

printf ("Digite o noh a ser removido\n");

scanf ("%d",&y);

w=existe(a,y);

if (w==1)

a=remover\_no(a,y);

else

printf ("Elemento nao existe\n");

system("pause");

break;

}

case 5:{

printf ("Digite o elemento a ser verificado\n");

scanf ("%d",&y);

w=existe(a,y);

if (w==1)

printf ("Elemento existe\n");

else

printf ("Elemento nao existe\n");

system("pause");

break;

}

case 6:{

printf ("Imprimir nohs maiores ou menores x\n\n");

printf("1 - Para maior que x\n");

printf("2 - Para menor que x\n");

scanf ("%d",&y);

if (y==1){

printf ("Digite x: \n");

scanf ("%d",&y);

imprimir\_maiorx(a,y);

}

else{

printf ("Digite x: \n");

scanf ("%d",&y);

imprimir\_menorx(a,y);

}

system("pause");

break;

}

case 7:{

printf ("Imprimir nohs entre x e y\n\n");

printf ("Digite x:\n");

scanf ("%d",&w);

printf ("Digite y:\n");

scanf ("%d",&y);

imprimir\_entrexy(a,w,y);

system ("pause");

break;

}

case 8:{

a=remover\_arvore(a);

break;

}

}

}

fclose(arq);

return 0;

}