1°)

a) E ,I , J e K

b)

* Nível 0: A
* Nível 1: B, C e D
* Nível 2: E, F, G e H
* Nível 3: I, J e K

c) 4

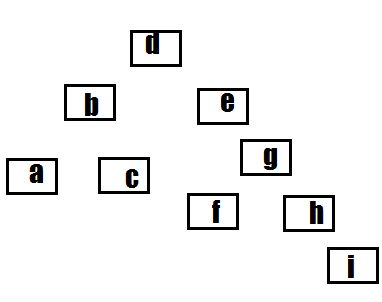
d) 3

e) 2

f) 1

2°)

3°)



4°)

Types.h

#ifndef true

#define true 1

#endif

#ifndef false

#define false 0

#endif

#ifndef estruturas

#define estruturas

typedef struct No{

int cod,preco,qtdEsto;

char descri[100];

struct No \*esq,\*dir;

}Tno;

typedef struct NoQueue{

struct NoQueue \*prox;

Tno \*ende;

}TNoQueue;

typedef struct NoDescritor{

TNoQueue \*inicio,\*fim;

}TNoDescritor;

typedef TNoDescritor \*Queue;

#endif

Queue.h

#include <stdio.h>

#include "Types.h"

Queue initialize (void);

void enqueue (Queue q, Tno \*num);

Tno \*dequeue (Queue q);

int head (Queue q);

int isEmpty (Queue q);

int isFull (Queue q);

void destroy (Queue q);

Queue.c

#include <stdlib.h>

#include "Types.h"

Queue initialize(void){

TNoDescritor \*aux;

aux = (TNoDescritor \*) malloc(sizeof(TNoDescritor));

aux->inicio = NULL;

aux->fim = NULL;

return aux;

}

void enqueue (Queue q, Tno \*endere){

NoQueue \*novo;

novo = (NoQueue \*) malloc (sizeof(NoQueue));

novo->ende = endere;

novo->prox = NULL;

q->fim = novo;

if(q->inicio == NULL)

q->fim = novo;

}

Tno \*dequeue (Queue q){

NoQueue \*aux;

Tno \*n;

aux = q->inicio;

q->inicio = q->inicio->prox;

n = aux->ende;

free(aux);

return n;

}

Tno \*head (Queue q){

return q->inicio->ende;

}

int isEmpty (Queue q){

if (q->inicio == NULL)

return true;

else

return false;

}

int isFull (Queue q){

return false;

}

void destroy (Queue q){

NoQueue \*aux;

while(q->inicio != NULL){

aux = q->inicio;

q->inicio = q->inicio->prox;

free(aux);

}

free(q);

}

Principal.c

#include <stdio.h>

#include <stdlib.h>

#include "Types.h"

#include "Queue.h"

void cadastrar(Tno \*\*L);

Tno \*criarNo(Tno \*\*L);

void exibirTodos(Tno \*L);

Tno \*buscar(Tno \*L,int codig);

void alterarPreco(Tno\*L,int codig);

void alterarEstoque(Tno \*L,int codig);

void exibirProduto(Tno \*L,int codig);

void remover(Tno \*\*L, int codig);

void remover\_no(Tno \*\*L);

Tno \*maior(Tno \*\*L);

int main(){

Tno \*Arvore=0,\*endAux;

int codAux;

char op;

do{

system("cls");

fflush(stdin);

printf("Menu\n1-Cadastrar\n2-Exibir Tudo\n3-Alterar preco do produto\n4-Alterar qtd. de estoque do produto\n"

"5-exibir produto\n6-Remover Produto\n7-Remover Tudo\n8-Sair\nDigite Sua Opcao: ");

scanf("%c",&op);fflush(stdin);

switch(op){

case '1':

printf("digite o codigo do produto: ");

scanf("%i",&codAux);

cadastrar(&Arvore,codAux);

break;

case '2':

exibirTodos(Arvore);

break;

case '3':

printf("Digite o codigo do produto: ");

scanf("%i",&codAux);

alterarPreco(Arvore,codAux);

break;

case '4':

printf("Digite o codigo do produto: ");

scanf("%i",&codAux);

alterarEstoque(Arvore,codAux);

break;

case '5':

printf("Digite o codigo do produto: ");

scanf("%i",&codAux);

break;

case '6':

printf("Digite o codigo do produto: ");

scanf("%i",&codAux);

remover(&Arvore,codAux);

break;

case '7':

break;

case '8':

break;

default:

printf("opcao invalida!");

break;

}

system("pause");

}while(op != '8');

}

void cadastrar(Tno \*\*L, int cod){

Tno \*novo,\*aux=\*L,\*aux2;

if((\*L)==NULL){

novo = criarNo(L);

(\*L)=novo;

novo->cod=cod;

}

else{

do{

if(cod == aux->cod){

printf("Codigo de produto repetido!\nOperacao cancelada!\n");

return;

}

else{

aux2 = aux;

if(cod < aux->cod)

aux = aux->esq;

else

aux = aux->dir;

}

}while(aux != NULL);

novo = criarNo(L);

novo->cod= cod;

if(cod < aux2->cod)

aux2 -> esq = novo;

else

aux2 -> dir = novo;

}

}

Tno \*criarNo(Tno \*\*L){

Tno \*novo;

novo = (Tno\*)malloc(sizeof(Tno));

novo->esq=NULL;

novo->dir=NULL;

fflush(stdin);

printf("Digite a descricao do produto: ");

gets(novo ->descri);fflush(stdin);

printf("Digite o preco do produto: ");

scanf("%i",&(novo->preco));

printf("Digite a qtd. no estoque do produto: ");

scanf("%i",&(novo->qtdEsto));

fflush(stdin);

return novo;

}

void exibirTodos(Tno \*L){

Queue fila;

Tno \*aux;

if (L != NULL)

{

initialize(&fila);

enqueue (fila,L);

while (isEmpty(fila) == false)

{

aux = dequeue(fila);

if (aux->esq != NULL)

enqueue(fila,aux->esq);

if (aux->dir != NULL)

enqueue(fila,aux->dir);

printf("Codigo : %i\n", aux->cod);

printf("Preco : %i\n", aux->preco);

printf("Descricao : %s\n", aux->descri);

printf("Qtd. no Estoque : %i\n", aux->qtdEsto);

}

}

}

Tno \*buscar(Tno \*L,int codig){

Tno \*aux = L;

if(L != NULL){

do{

if(codig == aux->cod)

return aux;

else{

if(codig < aux->cod)

aux = aux->esq;

else

aux = aux ->dir;

}

}while(aux!=NULL);

}

return NULL;

}

void alterarPreco(Tno\*L,int codig){

Tno \*endAux;

endAux=buscar(L, codig);

if(endAux == NULL){

printf("Produto nao encontrado!\n");

return;

}

else{

printf("Digite o novo preço do produto : ");

scanf("%i",&endAux->preco);

}

}

void alterarEstoque(Tno\*L,int codig){

Tno \*endAux;

endAux=buscar(L, codig);

if(endAux == NULL){

printf("Produto nao encontrado!\n");

return;

}

else{

printf("Digite qtd. de estoque atual do produto : ");

scanf("%i",&endAux->qtdEsto);

}

}

void exibirProduto(Tno \*L,int codig){

Tno \*endAux;

endAux=buscar(L, codig);

if(endAux == NULL){

printf("Produto nao encontrado!\n");

return;

}

else{

printf("Codigo: %i",endAux->cod);

printf("Preco: %i",endAux->preco);

printf("Descricao: %s",endAux->descri);

printf("Qtd. no Estoque: %i",endAux->qtdEsto);

}

}

void remover(Tno \*\*L,int codig){// recursivo

Tno \*aux = \*L;

if(\*L==NULL){

printf("Codigo nao encontrado! ");

}

else if( codig == (\*L)->cod ){

remover\_no(L);

}

else{

if(codig < (\*L)->cod)

remover(&((\*L)->esq),codig);

else

remover(&((\*L)->dir),codig);

}

}

void remover\_no(Tno \*\*L){

Tno \*pos;

pos = \*L;

if((\*L)->dir == NULL && (\*L)->esq == NULL){

\*L == NULL;

}

else if((\*L)->dir==NULL){

\*L = (\*L)->esq;

}

else if((\*L)->esq==NULL){

\*L = (\*L)->dir;

}

else{

pos = maior(&((\*L)->esq));

(\*L)->cod = pos->cod;

}

free(pos);

}

Tno \*maior(Tno\*\*L){

Tno \*aux= \*L,\*pos;

if(aux==NULL)

return aux;

else

pos = maior(&((aux)->dir));

}

5°)

a)

void inserir(Tno \*\*L,int n){

Tno \*aux,\*aux2,\*novo;

novo =(Tno\*) malloc(sizeof(Tno));

novo->dir = NULL;

novo->esq = NULL;

novo->nun = n;

if(\*L==NULL){

\*L = novo;

}

else{

aux = (\*L);

do{

aux2 = aux;

if(aux->nun < n)

aux = aux->esq;

else

aux = aux->dir;

}while(aux!=NULL);

if(aux2->nun < n)

aux2->esq = novo;

else

aux2->dir = novo;

}

}

b)

void remover(Tno \*\*L,int n){

Tno \*aux = \*L,\*aux2;

while(aux != NULL){

if(aux==NULL){

printf("Codigo nao encontrado! ");

break;

}

else if( n == aux->nun ){

remover\_no(&aux);

break;

}

else{

if(n < (\*L)->nun)

aux = aux->esq;

else

aux= aux->dir;

}

}

}

void remover\_no(Tno \*\*L){

Tno \*pos;

pos = \*L;

if((\*L)->dir == NULL && (\*L)->esq == NULL){

\*L == NULL;

}

else if((\*L)->dir==NULL){

\*L = (\*L)->esq;

}

else if((\*L)->esq==NULL){

\*L = (\*L)->dir;

}

else{

pos = maior(&((\*L)->esq));

(\*L)->nun = pos->nun;

}

free(pos);

}

Tno \*maior(Tno\*\*L){

Tno \*aux= \*L,\*pos;

while(1){

if(aux==NULL)

return pos;

else{

pos = aux;

aux = aux->dir;

}

}

}

c)

int contar(Tno \*L){

Tno \*aux;

int cont=0;

if(L != NULL){

if(L->dir != NULL && L->esq != NULL){

cont = contar(L->esq) + 1;

cont = contar(L->dir) + cont;

}

else if(L->dir == NULL)

cont = contar(L->esq) + 1;

else if(L->esq == NULL)

cont = contar(L->dir) + 1;

else

return 0;

}

else

return 0;

}

d)

int contar(Tno \*L){

Tno \*aux,\*aux2;

Queue fila;

int cont=0;

if(L == NULL){

printf("Arvore vazia!");

}

else{

if (L != NULL)

{

initialize(&fila);

enqueue (fila,L);

while (isEmpty(fila) == false)

{

aux = dequeue(fila);

if (aux->esq != NULL)

enqueue(fila,aux->esq);

if (aux->dir != NULL)

enqueue(fila,aux->dir);

cont++;

}

}

printf("Quantidade de nos na arvore : %i",cont);

}

}

e)

int contarFolhas(Tno \*L){

Tno \*aux;

int cont=0;

if(L != NULL){

if(L->dir != NULL && L->esq != NULL){

cont = contar(L->esq);

cont = contar(L->dir) + cont;

}

else if(L->dir == NULL)

cont = contar(L->esq);

else if(L->esq == NULL)

cont = contar(L->dir);

else

return 1;

}

else

return 1;

}

f)

int contarFolhas(Tno \*L){

Tno \*aux,\*aux2;

Queue fila;

int cont=0;

if(L == NULL){

printf("Arvore vazia!");

}

else{

if (L != NULL)

{

initialize(&fila);

enqueue (fila,L);

while (isEmpty(fila) == false)

{

aux = dequeue(fila);

if (aux->esq != NULL)

enqueue(fila,aux->esq);

if (aux->dir != NULL)

enqueue(fila,aux->dir);

if(aux->esq == NULL && aux->dir==NULL)

cont++;

}

}

printf("Quantidade de folhas na arvore sao : %i",cont);

}

}

g)

int contarNTerminais(Tno \*L){

Tno \*aux;

int cont=1;

if(L != NULL){

if(L->dir != NULL && L->esq != NULL){

cont = contar(L->esq) + cont;

cont = contar(L->dir) + cont;

}

else if(L->dir == NULL)

cont = contar(L->esq) + cont;

else if(L->esq == NULL)

cont = contar(L->dir) + cont;

else

return -1;

}

else

return -1;

}

h)

int contarNTerminais (Tno \*L){

Tno \*aux,\*aux2;

Queue fila;

int cont=0;

if(L == NULL){

printf("Arvore vazia!");

}

else{

if (L != NULL)

{

initialize(&fila);

enqueue (fila,L);

while (isEmpty(fila) == false)

{

aux = dequeue(fila);

if (aux->esq != NULL)

enqueue(fila,aux->esq);

if (aux->dir != NULL)

enqueue(fila,aux->dir);

if(aux->esq != NULL || aux->dir!=NULL)

cont++;

}

}

printf("Quantidade de nos nao terminais na arvore sao : %i",cont);

}

}