1º)

1. E,i,j,g,k,h
2. Nível 0 – A

NIvel 1 – B,C e D

Nivel 2 – E , F , G e H

Nivel 3 – I , J e K

1. 4
2. 3
3. 3
4. F, G , I e J

2º)

6

5

6

9

7

4

1

2

6

3º)

I

H

F

G

E

C

A

B

D

4º)

#Types.h

#ifndef ESTRUTURAS

#define ESTRUTURAS

typedef struct Produtos{

int codigo,estoque;

float preco;

char decri[100];

}Tproduto;

typedef struct No{

Tproduto info ;

struct No \*dir,\*esq;

}Tno;

typedef struct NoQueue{

Tno \*info;

struct NoQueue \*prox;

}NoQ;

typedef struct DescirtorQueue{

NoQ \*inicio,\*fim;

}Descritor;

typedef Descritor \*fila;

#endif

#ifndef FALSE

#define FALSE 0

#endif

#ifndef TRUE

#define TRUE 1

#endif

#Queue.h

#include "types.h"

Descritor \*createQueue();

void enqueue(fila \*L,Tno \*aux);

Tno \*dequeue(fila \*L);

Tproduto head(fila L);

int isempty(fila L);

int isfull(fila L);

void Destroy(fila \*L);

#Queue.c

#include "Queue.h"

#include <stdlib.h>

Descritor \*createQueue(){

Descritor \*novo;

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

novo ->fim = NULL;

novo ->inicio = NULL;

return novo;

}

void enqueue(fila \*L,Tno \*aux){

NoQ \*novo;

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

novo ->prox = NULL;

novo ->info = aux;

if(isempty(\*L)==TRUE){

(\*L)->inicio = novo;

}

(\*L)->fim->prox = novo;

(\*L)->fim = novo;

}

Tno \*dequeue(fila \*L){

Tproduto del;

NoQ \*aux;

aux = (\*L)->inicio;

del = aux->info;

(\*L)->inicio=(\*L)->inicio->prox;

free(aux);

if(isempty(\*L)==TRUE)

(\*L)->fim = NULL;

return del;

}

Tproduto head(fila L){

return L->inicio->info;

}

int isempty(fila L){

if(L->inicio == NULL)

return TRUE;

else

return FALSE;

}

int isfull(fila L){

return FALSE;

}

void Destroy(fila \*L){

NoQ \* aux;

while((\*L)->inicio != NULL){

aux = (\*L)->inicio;

(\*L)->inicio = (\*L)->inicio->prox;

free(aux);

}

free((\*L)->fim);

free(\*L);

\*L = NULL;

}

#Principal.c

#include <stdio.h>

#include <stdlib.h>

#include "Queue.h"

void InserirSemRepitidos(Tno \*\*Arvore, int nun);

Tno \*CriarNo(int codig);

void exibirPorNivel(Tno \*Arvore);

void alterarPreco(Tno \*Arvore,int nun);

void alterarQuantidade(Tno \*Arvore,int nun);

void exibirProduto(Tno \*Arvore,int nun);

void limparCadastro(Tno \*Arvore);

void Inserir(Tno \*\*Arvore,int nun){

Tno \*novo,\*aux;

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

novo -> info = nun;

novo->esq = NULL;

novo->dir = NULL;

if(\*Arvore == NULL){

\*Arvore = novo;

}else{

aux = \*Arvore;

while(aux != NULL){

if(nun < aux ->info){

if(aux->esq != NULL)

aux = aux->esq;

else{

aux ->esq = novo;

return;

}

}

else{

if(aux->esq != NULL)

aux = aux->dir;

else{

aux ->esq = novo;

return;

}

}

}

}

}

Tno \*CriarNo(int codig){

Tno \*novo;

Tproduto aux;

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

aux.codigo = codig;

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

gets(aux.decri);

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

scanf("%i",&aux.estoque);

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

scanf("%f",&aux.preco);

novo->info = aux;

novo->esq = NULL;

novo->dir = NULL;

return novo;

}

void InserirSemRepitidos(Tno \*\*Arvore, int nun){

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

if(\*Arvore == NULL){

novo = CriarNo(nun);

\*Arvore = novo;

}

else{

aux = (\*Arvore);

while(aux != NULL){

if(nun == aux->info.codigo){

printf("Valor repetido!\n");

return;

}

else if(nun < aux->info.codigo){

if(aux->esq != NULL)

aux = aux->esq;

else{// -- Quando for Menor que aux->info (Esquerda) --

novo = CriarNo(nun);

aux -> esq = novo;

}

}

else{// -- Quando for Maior que aux->info (Direita) --

if(aux->dir != NULL)

aux = aux->dir;

else{

novo = CriarNo(nun);

aux -> dir = novo;

}

}

}

}

}

void exibirPorNivel(Tno \*Arvore){

Tno \*resp;

fila Fila;

if(Arvore == NULL)

printf("Arvore vazia!\n");

else{

Fila = createQueue();

enqueue(&Fila,Arvore);

while(isempty(Fila) != TRUE){

resp = dequeue(&Fila);

if(resp->esq != NULL)

enqueue(&Fila,resp->esq);

if(resp->dir != NULL)

enqueue(&Fila,resp->dir);

printf("Codigo do produto : %i\n",resp->info.codigo);

printf("Descricao do produto : %s\n",resp->info.decri);

printf("Valor do produto : %.2f\n",resp->info.preco);

printf("Qtd. no estoque : %i\n\n",resp->info.estoque);

}

Destroy(&Fila);

}

}

Tno \*Buscar(Tno \*Arvore, int nun){ // -- Retorna o Nó se encontrar | Retorna NULL se ñ encontrar --

if(Arvore == NULL){

printf("Arvore Vazia!\n");

}

else{

while(Arvore != NULL){

if(Arvore->info.codigo == nun)

return Arvore;

else if(nun < Arvore ->info.codigo)

Arvore = Arvore->esq;

else

Arvore = Arvore->dir;

}

return NULL;

}

}

void alterarPreco(Tno \*Arvore, int nun){

Tno \* resp;

if(Arvore == NULL){

printf("Arvore vazia!\n");

}

else{

resp = Buscar(Arvore,nun);

if(resp == NULL){

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

}

else{

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

scanf("%f",&resp->info.preco);

}

}

}

void alterarQuantidade(Tno \*Arvore,int nun){

Tno \* resp;

if(Arvore == NULL){

printf("Arvore vazia!\n");

}

else{

resp = Buscar(Arvore,nun);

if(resp == NULL){

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

}

else{

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

scanf("%f",&resp->info.preco);

}

}

}

void exibirProduto(Tno \*Arvore,int nun){

Tno \* resp;

if(Arvore == NULL){

printf("Arvore vazia!\n");

}

else{

resp = Buscar(Arvore,nun);

if(resp == NULL){

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

}

else{

printf("Codigo do produto : %i\n",resp->info.codigo);

printf("Descricao do produto : %s\n",resp->info.decri);

printf("Valor do produto : %.2f\n",resp->info.preco);

printf("Qtd. no estoque : %i\n\n",resp->info.estoque);

}

}

}

void limparCadastro(Tno \*Arvore){

if(Arvore!=NULL){

limparCadastro(Arvore->esq);

limparCadastro(Arvore->dir);

free(Arvore);

}

}

5º)

c)

int contarNo(Tno \*Arvore){

int cont=0;

if(Arvore == NULL)

return cont;

else{

cont = 1;

if(Arvore->esq != NULL){

contarNo(Arvore->esq);

cont ++;

}

if(Arvore->dir != NULL){

contarNo(Arvore->dir);

cont++;

}

return cont;

}

}