Faculdade Estácio de Sá

**Aluno**: Euclides Rodrigues Junior

**N° da Matrícula**: 201902501691

**TRABALHO DE ESTRUTURA DE DADOS**

**PROFESSOR**: RICARDO EIJI KONDO

**TRABALHO AV2 – ESTRUTURA DE DADOS**

**Pilha=**

#include <iostream>

#include <tchar.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

using namespace std;

#define qtd 5

typedef struct DADOS\_ALUNO{

int number;

struct DADOS\_ALUNO \*link;

}dado;

int main(){

dado \*pLista;

int opcao=0;

void nulo(dado \*\*pR);

void inserir(dado \*\*pR);

void verificar(dado \*\*pR);

void remover(dado \*\*pR);

pLista = (dado \*)malloc(sizeof(struct DADOS\_ALUNO));

nulo(&pLista);

for(int y=0;y<qtd;y++){

cout<<"\n1 - Inserir: ";

cout<<"\n2 - Verificar: ";

cout<<"\n3 - Remover: ";

cout<<"\n5 - Esvaziar: ";

cout<<"\n5 - Sair: "<<endl;

cout<<"Selecione uma das opcoes: ";

cin>>opcao;

switch (opcao) {

case 1:

inserir(&pLista);

break;

case 2:

verificar(&pLista);

break;

case 3:

remover(&pLista);

break;

case 4:

break;

case 5:

exit(0);

break;

}

}

return 0;

}

int nulo(dado \*\*pR){

(\*pR)->link=NULL;

}

void inserir(dado \*\*pR){

dado \*temporario;

int valor;

cout<<"Valor: ";

scanf("%d", &valor);

temporario = (dado \*)malloc(sizeof(dado));

temporario->number=valor;

temporario->link=(\*pR)->link;

(\*pR)->link = temporario;

}

void verificar(dado \*\*pR){

dado \*temporario;

if((\*pR)->link==NULL){

cout<<"Lista Vazia!!!\n";

}

else {

temporario = (dado \*)malloc(sizeof(dado));

temporario = (\*pR)->link;

while(temporario!=NULL){

cout<<"Valor: "<<temporario->number<<endl;

temporario = temporario->link;

}

}

}

void remover(dado \*\*pR){

dado \*temporario;

if((\*pR)->link==NULL){

cout<<"Lista vazia!!!"<<endl;

}

else {

temporario = (

\*pR)->link;

(\*pR)->link = temporario->link;

free(temporario);

}

}

**Filas=**

#include <stdio.h>

#include <stdlib.h>

#define MALLOC(x) ((x \*) malloc (sizeof(x)))

void insere (void);

void exibir (void);

void exclui (void);

struct no{

int item;

struct no \*proximo,\*anterior;

};

typedef struct no no\_t;

no\_t \*novo;

struct cabec{

int tamanho;

no\_t \*first;

no\_t \*last;

};

typedef struct cabec cabec\_t;

cabec\_t \*t;

int main(void)

{

int op,k;

t=MALLOC(cabec\_t);

t->tamanho = 0;

novo = t->first = t->last = 0;

for (;;)

{

printf("\n1-Inserir");

printf("\n2-Exibir");

printf("\n3-Excluir");

printf("\n4-Sair");

printf("\nSua opï¿½ï¿½o: ");

scanf("%d",&op);

switch(op){

case 1 : insere();

break;

case 2 : exibir();

break;

case 3 : exclui();

break;

case 4 : exit(0);

default:

printf("Opï¿½ï¿½o Errada");

scanf("%d",&k);

}

}

}

void insere(void)

{

int j;

printf("Dado: ");

scanf("%d",&j);

novo = MALLOC(no\_t);

novo->proximo = 0;

novo->item = j;

if (t->first == 0)

{

novo->anterior = 0;

t->first = novo;

}

else

{

t->last->proximo = novo;

novo->anterior = t->last;

}

t->last = novo;

t->tamanho++;

}

void exibir(void)

{

no\_t \*p;

char k[80];

if (t->first==0)

{

printf("Lista Vazia");

return;

}

p = t->first;

do{

printf("\Dado %d",p->item);

p = p->proximo;

}while (p!=0);

printf("\n\nTem %d Dados",t->tamanho);

scanf("%s",&k);

}

void exclui(void)

{

no\_t \*anterior1, \*novo1, \*ajuda;

int j,h=0;

char i[80];

novo1 = t->first;

if (t->first==0)

{

printf("Lista Vazia");

scanf("%s",&i);

return;

}

printf("Digite o dado que deseja excluir: ");

scanf("%d",&j);

do{

if (novo1->item == j)

{

if (t->first->anterior == 0 && t->first->proximo == 0)

{

novo = t->first = t->last = 0;

t->tamanho = 0;

break;

}

else

if (novo1->anterior != 0 && novo1->proximo == 0)

{

anterior1->proximo=0;

t->last = anterior1;

free(novo1);

t->tamanho--;

break;

}

else

if (h == 0 && t->first->proximo != 0 && t->first->anterior == 0)

{

t->first = novo1->proximo;

t->first->anterior = 0;

novo1->proximo = 0;

novo1->anterior= 0;

free(novo1);

t->tamanho--;

break;

}

else

if (novo1->anterior != 0 && novo1->proximo != 0)

{

anterior1->proximo = novo1->proximo;

novo1->proximo = 0;

anterior1->proximo->anterior = novo1->anterior;

novo1->anterior = 0;

free(novo1);

t->tamanho--;

break;

}

}

anterior1 = novo1;

novo1 = novo1->proximo;

h++;

}while(novo1 != 0);

}