

PILHA: é uma poderosa estrutura de dodos que pode ser implementada em vetor on en lista ligada Quando por baseada em vetor ela tera um tamanio máxing mas quando 200 baseada em lista ligada sau tamanho será limitado apenas pela memoria disponível

A filosofia (on politica) de acesso da PeL HA

é chamada de LAST IN/FIRST Out (LIFO) súltimo a entrar/primeiro a sair. A ocesso à uma pilla del sempre começar pels Topo da pilha.

Todo novo elemento é sempre incerido no topo da pilha. E sempre deve ser & elemento de tops do pilha a ser remando. Principais funcionalidades.

consultaritopo(): denly o elemento do topo do pilha.

(→ PUSH(): insere um elemento no topo da pilha POP(): remove o elemento do topo da pilha

mostrartilho (): mostra todos os elementos de pilka, a partes do topo

#include <stdio.h>

#include <stdlib.h>

struct tpNo {

int valor;

struct tpNo \*prox;

};

void push(struct tpNo \*\* pTopo, int parm);

int pop(struct tpNo\*\*pTopo);

int consultarTopo(struct tpNo\*pTopo);

void mostrarPilha(struct tpNo\*pTopo);

int main(void){

struct tpNo \*topo=NULL;

int qtd, pos;

qtd=rand()%51;

for(pos=0;pos<qtd;pos++){

push(&topo, rand()%101);

}

mostrarPilha(topo);

while(topo!=NULL){

printf("Topo=%d\n",pop(&topo));

}

return 0;

}

void mostrarPilha(struct tpNo\*pTopo){

while(pTopo !=NULL){

printf("Valor=%d", pTopo->valor);

pTopo=pTopo->prox;

}

}

int consultarTopo(struct tpNo\*pTopo){

return pTopo->valor;

}

void push (struct tpNo\*\*pTopo, int parm){

struct tpNo \*aux;

if(!(aux=malloc(sizeof(struct tpNo)))){

printf("Faltou memoria!\n");

exit(1);

}

aux->valor=parm;

aux->prox=\*pTopo;

\*pTopo=aux;

}

int pop(struct tpNo\*\*pTopo){

int num;

struct tpNo\*aux;

if(\*pTopo==NULL){

printf("Pilha vazia!\n");

exit(2);

}

aux=\*pTopo;

num=aux->valor;

\*pTopo=aux->prox;

free(aux);

return num;

}

Ex.1) Escreva um programa para carregar uma pilha com 10 valores aleatórios. Mostre a pilha, crie outra pilha com os elementos da primeira, mas ordenados e mostre a nova pilha.

#include <stdio.h>

#include <stdlib.h>

struct tpNo {

int valor;

struct tpNo \*prox;

};

void push(struct tpNo \*\* pTopo, int parm);

int pop(struct tpNo\*\*pTopo);

int consultarTopo(struct tpNo\*pTopo);

void mostrarPilha(struct tpNo\*pTopo);

int main(void){

struct tpNo \*topo=NULL;

int qtd, pos;

qtd=rand()%51;

for(pos=0;pos<qtd;pos++){

push(&topo, rand()%101);

}

mostrarPilha(topo);

while(topo!=NULL){

printf("Topo = %d\n",pop(&topo));

}

return 0;

}

void mostrarPilha(struct tpNo \*pTopo){

while(pTopo !=NULL){

printf("Valor=%d ", pTopo->valor);

pTopo=pTopo->prox;

}

}

int consultarTopo(struct tpNo \*pTopo){

return pTopo->valor;

}

void push (struct tpNo\*\*pTopo, int parm){

struct tpNo \*aux;

if(!(aux=malloc(sizeof(struct tpNo)))){

printf("Faltou memoria!\n");

exit(1);

}

aux->valor=parm;

aux->prox=\*pTopo;

\*pTopo=aux;

}

int pop(struct tpNo\*\*pTopo){

int num;

struct tpNo \*aux;

if(\*pTopo==NULL){

printf("Pilha vazia!\n");

exit(2);

}

aux=\*pTopo;

num=aux->valor;

\*pTopo=aux->prox;

free(aux);

return num;

}

void ordena(int pVet[], int tam);

void ordena(int pVet[], int tam){

int inte,exte,aux;

for(exte=0;exte<tam-1;exte++){

for(inte=exte+1;inte<tam;inte++){

if(pVet[exte]>pVet[inte]){

aux=pVet[exte];

pVet[exte]=pVet[inte];

pVet[inte]=aux;

}

}

}

}

Ex.2) Escreva um programa para ler uma expressão digitada pelo usuário que contenha apenas as 4 operações (+,-,x,/) e mostre o resultado.