TIPO ABSTRATO DE DADOS – PILHA

Pilha é o nome que se dá a uma lista linear com a seguinte restrição ao acesso: a inserção e a remoção de um elemento da lista só pode ser feita em uma das extremidades da lista, denominada topo. Desta forma, ocorre que o primeiro elemento inserido é o último elemento a ser removido (LIFO = *Last In First Out*).

As únicas operações permitidas em uma pilha são: criar a pilha vazia, esvaziar a pilha, colocar um novo elemento, retirar o elemento do topo da pilha, acessar o topo da pilha e verificar o estado da pilha (vazia ou cheia).

A seguir, a estrutura de armazenamento de dados de uma pilha de números inteiros, com um campo identificado por topo, que indica a posição em que se encontra o último elemento colocado na pilha. A constante MaxPilha dimensiona o array com 10 posições indexadas de 0 a 9. Com a escolha que fizemos, a pilha tem capacidade para armazenar 10 elementos e se topo= Max-1, identificamos a pilha cheia.

#define MaxPilha 10

typedef struct {

int topo;

int tabela[MaxPilha];

} Pilha;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | tabela |  |  | topo |
| 9 |  |  |  | 3 |
| 8 |  |  |  |  |
| 7 |  |  |  |  |
| 6 |  |  |  |  |
| 5 |  |  |  |  |
| 4 |  |  |  |  |
| 3 | 8 |  |  |  |
| 2 | 15 |  |  |  |
| 1 | 27 |  |  |  |
| 0 | 13 |  |  |  |

A ilustração mostra uma pilha com os números 13, 27, 15 e 8 e o topo indicando a posição em que se encontra o último elemento colocado na pilha.

Operações:

1. criar Pilha vazia – inicializa o valor do topo
2. esvaziar Pilha - inicializa o valor do topo
3. verificar Pilha vazia – devolve TRUE se a pilha é vazia
4. verificar Pilha cheia – devolve TRUE se não é possível acrescentar mais elementos à pilha
5. acessar topo da Pilha – recupera o último elemento colocado na pilha
6. colocar na Pilha – acrescenta um novo elemento à pilha
7. retirar da Pilha – remove o último elemento colocado na pilha

Exemplo de uma interface para uma pilha de inteiros:

/\* construtores: ambos criam uma pilha vazia, devemos escolher um deles \*/

Pilha criarPilha();

void criarPilhaVazia(Pilha \*);

/\* acesso \*/

int acessarTopo(Pilha);

bool verificarPilhaVazia(Pilha);

bool verificarPilhaCheia(Pilha);

/\* manipulação - a função push coloca um número na pilha e a função pop retira o número que está no topo da pilha, mas não devolve o que foi retirado \*/

void push(Pilha \*, int);

void pop(Pilha \*);

void esvaziarPilha(Pilha \*);

EXERCÍCIOS

1) Implementar o tipo de dados Pilha, com as especificações dadas anteriormente, gravando um arquivo header.

/\* Pilha de inteiros \*/

/\*

Arquivo: <suas iniciais>\_PilhaInt.h

Autor:

Date:

Descrição: Implementa o tipo Pilha - TÓPICO 3

\*/

2) Implemente a função

void estragar(Pilha \*,int);

que deve provocar a seguinte mudança em uma pilha:

* se a pilha não for vazia, a função *estragar* deve substituir o número que está no topo da pilha pelo número x dado como parâmetro;
* caso contrário, a função *estragar* deve encher a pilha com o valor x dado como parâmetro.

3) Qual a saída produzida pelas chamadas das funções?

push(&p1, 1);

push(&p1, 3);

push(&p1, 5);

push(&p1, 7);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

4) Qual a saída produzida pelas chamadas das funções?

push(&p1, 1);

push(&p1, 3);

push(&p1, 5);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

push(&p1, 1);

push(&p1, 3);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

5) Qual a saída produzida pelas chamadas das funções?

push(&p1, 1);

push(&p1, 3);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

push(&p1, 5);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

push(&p1, 7);

y = acessarTopo(p1); printf(“ x = %d \n”, y); pop(&p1);

6) Escreva uma seqüência de chamadas com push e pop, que coloque na pilha p1 os inteiros 1, 2, 3 e 4 (cada valor é colocado uma única vez, nesta ordem), retire da pilha quantas vezes quiser e produza como saída a seqüência 3,2,4,1.

7) Uma lista de caracteres está armazenada em uma estrutura do tipo ListaCHAR

#define Max 100;

typedef struct{

int tamanho;

char elemento[Max];

} ListaCHAR;

Escreva uma função para verificar se os pares de parênteses “(“ e “)” de uma expressão estão bem colocados, usando uma pilha de caracteres. Utilize a sugestão: ao encontrar um parênteses “(“ , coloque na pilha, ao encontrar um parênteses “)” , retire da pilha. Se houver uma tentativa de retirada em uma pilha vazia ou se ao terminar a lista de caracteres, a pilha não estiver vazia então a expressão não está bem construída. A função deve devolver um valor booleano.