MAUÁ ESCOLA DE ENGENHARIA

13

Pilhas

Pilha (*stack*)

Empilha elementos que deverão ser processados na ordem "primeiro que entra é o último que sai" (exemplo: pilha de pratos);

Em computação, uma pilha serve para:

- armazenar o estado do programa em chamadas recursivas (lembra do QuickSort?);
- representar processadores reais ou virtuais baseados em pilha (a calculadora HP é baseada em pilha);
- auxílio na avaliação de expressões algébricas;
- aplicações específicas em alguns algoritmos (busca em profundidade, etc.).

Pilha (*stack*)

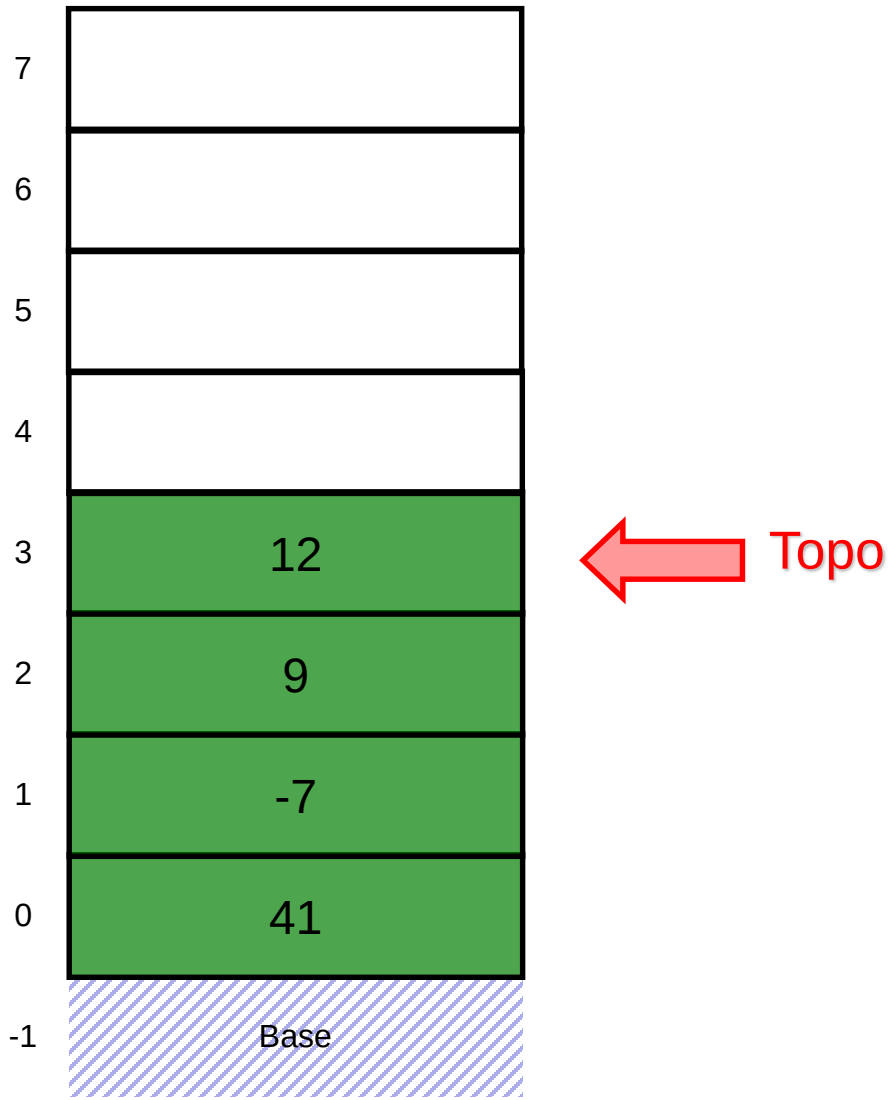
LIFO: *Last In First Out*;

a operação de inserção coloca um elemento no topo da pilha enquanto que a operação de retirada retira o elemento que estiver no topo;

Estudaremos uma pilha implementada com um vetor:

- itens são inseridos na posição “topo”;
- itens são retirados na posição “topo”;
- a pilha está vazia quando o topo é -1 (ou a base);
- a pilha está cheia quando o topo é o tamanho total da pilha.

Pilha (*stack*)



pilha.h

```
#ifndef PILHA_H_INCLUDED

#define PILHA_H_INCLUDED


#define MAX 20

#define MAX_PILHA 10

typedef struct {

    float valor;

    char texto[MAX];

} Dados;


typedef struct {

    Dados elemento[MAX_PILHA];

    int topo;

} TPilha;
```

pilha.h

```
//cria uma pilha nova
void CriarPilha(TPilha *p);
//insere
int InserirNaPilha(TPilha *p, Dados dados);
//retira
int RetirarDaPilha(TPilha *p, Dados *dados);
//indica se a pilha está vazia
int PilhaVazia(TPilha p);
//indica se a pilha está cheia
int PilhaCheia(TPilha p);
//retorna o tamanho da pilha
int QuantidadeNaPilha(TPilha p);
#endif // PILHA_H_INCLUDED
```

pilha.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "pilha.h"

void CriarPilha(TPilha *p){
 /* define valores iniciais das propriedades da pilha */
    p->topo = -1;
}
```

pilha.c

```
int InserirNaPilha(TPilha *p, Dados dados) {
    int pc = PilhaCheia(*p);
    if (!pc) {
        p->topo++;          /* incrementa posição do topo */
        /* insere elemento no topo da pilha */
        p->elemento[p->topo] = dados;
    }
    return !pc; /* retorna se conseguiu inserir o dado */
}
```

pilha.c

```
int RetirarDaPilha(TPilha *p, Dados *dados) {
    int pv;
    pv = PilhaVazia(*p);
    if (!pv) {
        /* recupera informação do topo da pilha */
        *dados = p->elemento[p->topo];
        p->topo--;              /* decrementa posição do topo */
    }
    return !pv; /* retorna se conseguiu remover o dado */
}
```

pilha.c

```
int PilhaVazia(TPilha p) {
    return p.topo == -1;
}


int PilhaCheia(TPilha p) {
    return p.topo == MAX_PILHA - 1;
}


int QuantidadeNaPilha(TPilha p) {
    return p.topo + 1;
}
```