Aula 09 - Estrutura de dados 1 – Uniube

Prof. Marcos Lopes

34 9 9878 0925

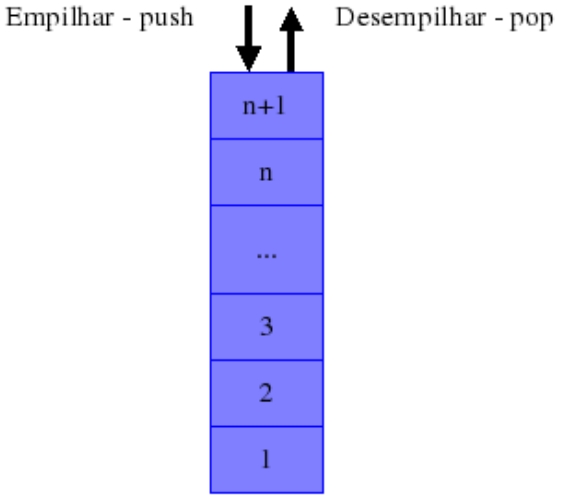
Estrutura de dados do tipo pilha

Uma pilha é uma estrutura de dados que admite remoção de elementos e inserção de novos objetos.

Mais especificamente, uma pilha (stack) é uma estrutura sujeita à seguinte regra de operação:

** sempre que houver uma remoção, o elemento removido é o que está na estrutura há menos tempo.

Em outras palavras, o primeiro objeto a ser inserido na pilha é o último a ser removido. Essa política é conhecida pela sigla LIFO (= Last-In-First-Out).



Estrutura de dados do tipo pilha

De modo geral a API (Application Programming Interface) da estrura de dados Pilha tem quatro funções:

push: insere um elemento no topo da Pilha
void push(int conteudo, Pilha* pilha)

pop: pega e retira um elemento do topo da Pilha
int pop(Pilha* pilha)

isEmpty: verifica se a Pilha está vazia
bool isEmpty(Pilha* pilha)

peek: pega um elemento do topo, sem retirar ele da Pilha
int peek(Pilha* pilha)

Da mesma forma que na implementação da estrutura de dados Lista, podemos implementar a Pilha usando vetores (arrays) ou células encadeadas.

Vamos trabalhar o uso de células encadeada e implementar as 4 funções básicas da Pilha.

Estrutura de dados do tipo pilha

Exercício 1) Implementar um programa em C com as 4 operações de uma Pilha, com base na implementação com células encadeadas. O “main” deverá ter demonstrações das chamadas das funções da Pilha.

Observação: usar as operações definidas para Lista encadeada e ajustar pra definição da Pillha com célula encadeada.

Sugestão de função “main”, no próximo slide:

Estrutura de dados do tipo pilha

```
int main(void) {

    celula *pilha = NULL;
    pilha = (celula*) malloc(sizeof(celula));   //head
    pilha->prox = NULL;

    push(10, pilha);
    push(20, pilha);
    push(30, pilha);

    int topo = peek(pilha);
    printf("\ntopo peek: %d", topo);
    topo = pop(pilha);
    printf("\ntopo pop: %d", topo);
    topo = pop(pilha);
    printf("\ntopo pop: %d", topo);
    topo = pop(pilha);
    printf("\ntopo pop: %d\n\n", topo);

    return EXIT_SUCCESS;
}
```

Implementação da estrutura de pilha com célula encadeada (com cabeça)

Vamos sugerir uma implementação nos slides subsequentes:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct celula celula;

struct celula {
    int conteudo;
    celula *prox;
};
```

Implementação da estrutura de pilha com célula encadeada (com cabeça)

(continuação)

```
void push(int x, celula *pilha){
    celula *nova;
    nova = (celula*) malloc (sizeof (celula));
    nova->conteudo = x;
    nova->prox = pilha->prox;
    pilha->prox = nova;
}

int pop(celula *pilha) {
    if(pilha->prox == NULL) {
        printf("Não tem elemento pra retirar!");
        return -1;
    }
    celula *topo = pilha->prox;
    int topoConteudo = topo->conteudo;
    pilha->prox = topo->prox;
    free(topo);
    return topoConteudo;
}
```

Implementação da estrutura de pilha com célula encadeada (com cabeça)

(continuação)

```
bool isEmpty(celula *pilha) {
    if(pilha->prox == NULL) {
        return true;
    }
    return false;
}

int peek(celula *pilha) {
    if(pilha->prox == NULL) {
        printf("Não tem elemento pra inspecionar!");
        return -1;
    }

    return (pilha->prox)->conteudo;
}
```

Exercício 2) Considere o problema de decidir se uma dada sequência de parênteses e colchetes está bem-formada (ou seja, parênteses e colchetes são fechados na ordem inversa àquela em que foram abertos). Por exemplo, a sequência

(( ) [( )])

está bem-formada, enquanto ([ )] está malformada. Suponha que a sequência de parênteses e colchetes está armazenada em uma string ASCII s. (Como é hábito em C, o último caractere da string é \0.)

Usaremos uma pilha para resolver o problema. O algoritmo é simples: examine a string da esquerda para a direita e empilhe os parênteses e colchetes esquerdos à espera de que apareçam os correspondentes parênteses e colchetes direitos.

O balanceamento estará correto se após percorrer toda a string de entrada, a pilha esteja vazia e nenhum problema de balanceamento tenha ocorrido.