ESTRUTURAS DE DADOS - FUNDAMENTOS

Parte II

Pilhas (stack)

Definição:

Uma pilha é uma lista linear restrita pelo fato das operações de acesso (top), inserção (push) e remoção (pop) serem realizadas somente em uma das extremidades chamada topo.
Em outras palavras, o ultimo elemento inserido na pilha será o primeiro a ser removido.
Essa característica é conhecida pela sigla LIFO *(Last-In-First-Out).*

Representação em alocação sequencial

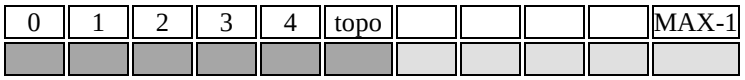
Considerando uma pilha representada em um vetor de nome elementos de [0..MAX-1] elementos. Vamos supor que os elementos da pilha são inteiros. A parte do vetor ocupada pela pilha será

elementos[0..topo]

elementos

O índice topo indica o ***topo*** da pilha. Esta é a primeira posição vaga da pilha. A pilha está ***vazia*** se topo vale -1 e ***cheia*** se topo vale MAX-1.
Para ***consultar*** (acesso) a pilha sem desempilhar faça x = elementos[topo].
Para ***inserir,*** ou seja, para empilhar *(push)* um objeto x na pilha faremos um *deslocamento* do topo utilizando topo = topo+1 e em seguida a, elementos[topo] = x;
Antes de empilhar, verifique se a pilha já está cheia para evitar que ela *transborde* (ou seja, para evitar um *stack overflow*). Em geral, a tentativa de inserir em uma pilha cheia é uma situação excepcional, que indica um mau planejamento lógico do seu programa.
Para remover um elemento da pilha, operação conhecida como desempilhar *(pop)*, devemos, caso necessário, *guardar* o valor em x = elementos[topo] e, seguida, retirarmos a referência de topo para topo = topo-1. Desta maneira, a nova situação do vetor será:

elementos



É claro que você só deve desempilhar se tiver certeza de que a pilha não está vazia, evitando o que chamamos de *stack underflow*.

Implementação utilizando C/C++

Arquivo Pilha.h

```
#ifndef PILHA_H_INCLUDED
#define PILHA_H_INCLUDED
template <typename Tipo>
struct Pilha{
  private:
   Tipo *v;
   int topo;
   int tamanho;
  public:
   Pilha(int tam){
      tamanho = tam;
      v = new Tipo[tamanho];
      topo=-1;
   }
   ~Pilha(){
      delete v;
   }
   void empilha(Tipo x){
      topo++;
      v[topo]=x;
   }
   Tipo desempilha(){
      Tipo temp=v[topo];
      topo--;
      return temp;
   }
   Tipo elementoDoTopo(){
      return v[topo];
   }
   bool pilhaCheia(){
      return topo==tamanho-1;
   }
   bool pilhaVazia(){
      return topo==-1;
   }
   int getTopo(){
      return topo;
   }
   int getTamanho(){
      return tamanho;
   }
   Tipo getValor(int pos){
      return v[pos];
   }
};

#endif // PILHA_H_INCLUDED
```

Exercício:

Considere os protótipos, definidos abaixo, para uma estrutura de dados do tipo Pilha:

a) Procedimento Empilha(x:inteiro); - empilha um novo elemento na pilha
b) Função Desempilha:inteiro; - função que desempilha um elemento retornando o elemento desempilhado
c) Função Topo:inteiro; - retorna o elemento do topo da pilha

Mostre a situação de uma pilha P, inicialmente vazia, após a execução de cada umas das operações:

1. Empilha(1); 2. Empilha(4); 3.Empilha(5); 4.Empilha(Topo());
5. Empilha(Desempilha()); 6. Desempilha(); 7.Empilha(9); 8. Desempilha();
9. Empilha(3); 10. Empilha(2); 11. Empilha(Desempilha()); 12. Desempilha();

Aplicação: parênteses e colchetes

Suponha que queremos decidir se uma dada seqüência de parênteses e colchetes está bem formada. Por exemplo, a primeira das seqüências abaixo está bem formada enquanto a segunda não está.

{(()[()]}	{[(])}

Suponha que a seqüência de parênteses e colchetes está armazenada em uma cadeia de caracteres *(string)*. Construa uma aplicação, utilizando uma pilha, para verificar se uma dada seqüência está bem formada. Uma solução é apresentada em *Estruturas de Dados Usando C e C++ (Tenembaum)*:

```
algoritmo analizaExpressao;
var Expressao:string;
   i:inteiro;
   Simbolo:char;
   Valido:boolean;
Inicio
  Obter(Expressao);
  i := 1;
  valido := true;
  Enquanto  i < Tamnaho(Exp)+1 faca
  Inicio
      Simbolo := Expressao[i];
      se simbolo pertence a { '{','[','(' } entao
        Empilha(MinhaPilha,simbolo);
      senao
        Inicio
         Se Simbolo pertence a { '}',']',')' } entao
           se PilhaVazia(MinhaPilha) entao
             valido:=false
           senao
             se (Simbolo = '}')
                          e (ElementoDoTopo(MinhaPilha) = '{' ) entao
                Desempilha(MinhaPilha)
            senao
              se (Simbolo = ']')
                          e (ElementoDoTopo(MinhaPilha) ='[' ) then
                Desempilha(MinhaPilha)
              senao
                se (Simbolo = ')')
                          e (ElementoDoTopo(MinhaPilha) = '(' ) entao
                   Desempilha(MinhaPilha);
        end;
      i := i + 1;
   fim enquanto;
   Se PilhaVazia(MinhaPilha) e valido entao
      Escrevere('Expressão Correta')
   senao
      Escrever('Expressão Incorreta');
fim algoritimo.
```