VOLTAR

Pilhas

Apresentar o conceito de pilhas e mostrar alguns exemplos da utilização dessa estrutura de armazenamento de dados.

NESTE TÓPICO

Introdução

Uma das estruturas de dados mais simples é a pilha. Possivelmente, por essa razão, é a estrutura de dados mais utilizada em programação, sendo inclusive implementada diretamente pelo hardware da maioria das máquinas modernas.

Para se entender o funcionamento de uma estrutura de pilha, pode-se fazer uma analogia com uma pilha de pratos. Se quiser adicionar um prato na pilha, o mesmo deve ser colocado no topo. Para pegar um prato da pilha, deve-se retirar o prato do topo. Assim, tem-se que retirar o prato do topo para ter acesso ao próximo prato. A estrutura de pilha funciona de maneira análoga. Cada novo elemento é inserido no topo e só tem-se acesso ao elemento do topo da pilha.

Outro exemplo que ilustra o funcionamento de uma pilha é a analogia com o que acontece em uma rua sem saída, que, de tão estreita, apenas um carro passa por vez. Nesse caso, o primeiro carro a sair será o último a ter entrado. Observe, ainda, que não podemos retirar qualquer carro e não podemos inserir um carro de tal forma que ele não seja o último.

Definição

Uma pilha é uma lista linear em que apenas as operações de acesso, inserção e remoção são possíveis.

A ideia fundamental da pilha é que todo o acesso aos seus elementos é feito por meio do seu topo. Assim, quando um elemento novo é introduzido na pilha, passa a ser o elemento do topo, e o único elemento que pode ser removido da pilha é o do topo. Isto faz com que os elementos da pilha sejam retirados na ordem inversa à ordem em que foram introduzidos: o primeiro que sai é o último que entrou (a sigla LIFO – last in, first out – é usada para descrever esta estratégia).

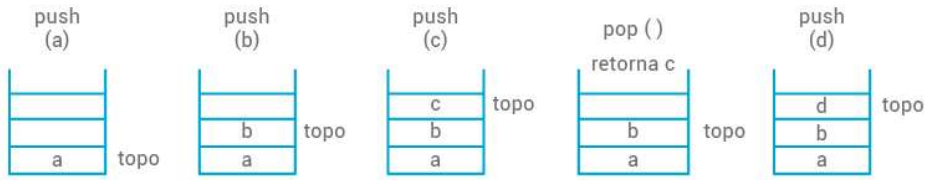
Operações Básicas de uma Pilha

Existem duas operações básicas que devem ser implementadas numa estrutura de pilha: a operação para empilhar um novo elemento, inserindo-o no topo, e a operação para desempilhar um elemento, removendo-o do topo. É comum se referir a essas duas operações pelos termos em inglês push (empilhar) e pop (desempilhar).

Seja p uma variável do tipo pilha e x um elemento qualquer. Então, temos:

- push (p, x): procedimento que insere (empilha) x no topo de p.
- pop (p): função que remove (desempilha) o elemento do topo de p, devolvendo o valor do topo.

A figura a seguir ilustra o funcionamento conceitual das operações push e pop em uma pilha.



Representação gráfica de uma pilha que armazena caracteres

Fonte: Do próprio autor

Veremos, a seguir, a implementação das seguintes operações para manipular e acessar as informações da pilha:

- Inserir um elemento no topo.
- Remover o elemento do topo.

- Mostrar os elementos da pilha.

Implementação de pilhas

Uma maneira muito comum de implementar a pilha é por meio da utilização de vetores. Para tanto, é necessário sabermos a quantidade máxima de elementos que podem ser armazenados na pilha, ou seja, neste caso, a estrutura pilha tem seu limite conhecido.

Dado um vetor conhecido de números inteiros para armazenar os elementos da pilha, os primeiros elementos ocupam as primeiras posições do vetor (n posições ocupadas). Temos, então, n elementos armazenados na pilha, logo o elemento n-1 representa o elemento do topo. O tamanho do vetor é determinado por meio da constante *MAX*, a qual deve ser definida no inicio do programa.

Abaixo é mostrado o procedimento "push" para inserir um elemento no topo da pilha. O procedimento recebe como parâmetros o vetor "pilha" e o novo elemento x que será inserido. Para a implementação desse procedimento, é preciso considerar que a variável *topo* é global e, inicialmente, seu valor é zero. Além disso, ela está sendo utilizada para controlar a posição no vetor onde será inserido o novo elemento na pilha.

```
1.  void push (int pilha [MAX], int x)
2.  {
3.      if (topo == MAX)
4.            printf ("\n Pilha Cheia");
5.      else{
6.            pilha [topo] = x;
7.            topo++;
8.      }
9.  }
```

Para remover (desempilhar) elementos do topo da pilha, deve ser utilizada a função "pop". Além de desempilhar o elemento, a função também retorna o valor que foi retirado da pilha. Vale lembrar que a remoção na pilha é feita por meio da variável *topo*, sendo que, a cada elemento retirado, é feito um decremento nesta variável. Abaixo é mostrada a implementação da função "pop".

```
1.  int pop (int pilha [MAX])
2.  {
3.       int x;
4.
5.       if (topo >= 1)
6.       {
7.            x = pilha [topo - 1];
8.            topo--;
9.       }
10.      else printf ("\n Pilha vazia");
11.
12.     return (x);
13. }
14.
15.
```

A exibição dos elementos da pilha deve ser feita por meio da utilização do laço *para*, onde o fim do laço é definido pela variável *topo*. Abaixo é mostrada a implementação do procedimento para exibir os elementos da pilha.

```
1.  void exibe (int pilha [MAX])
2.  {
3.       int i;
4.
5.       if (topo >= 1){
6.           for (i=0; i<topo; i++)
7.               printf ("\n Pilha [%d] = %d",i,pilha [i]);
8.       }
9.    else printf ("\n Pilha vazia");
10. }
```

Exemplo 1

A seguir, apresentamos um exemplo completo de um algoritmo que trabalha com uma pilha de números inteiros. É feita a inserção de 3 elementos na pilha e, na sequencia, 1 elemento é removido.

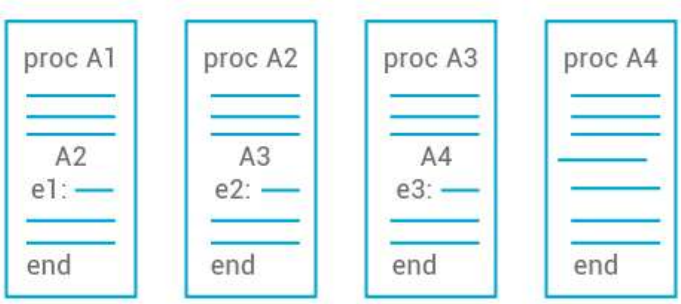
```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

//prototipos das funcoes
void push (int pilha [MAX], int x);
int pop (int pilha [MAX]);
void exibe (int pilha [MAX]);

int topo;

main(){
    int pilha [MAX];
    push (pilha, 22);
    push (pilha, 33);
    push (pilha, 44);
    pop (pilha);
    exibe (pilha);
    system ("PAUSE");
}

void exibe (int pilha [MAX])
{
     int i;

     if (topo >= 1){
         for (i=0; i<topo; i++)
             printf ("\n Pilha [%d] = %d",i,pilha [i]);
     }
  else printf ("\n Pilha vazia");
}

int pop (int pilha [MAX])
{
     int x;

     if (topo >= 1)
     {
          x = pilha [topo - 1];
          topo--;
     }
     else printf ("\n Pilha vazia");

    return (x);
}
void push (int pilha [MAX], int x)
{
    if (topo == MAX)
          printf ("\n Pilha Cheia");
    else{
          pilha [topo] = x;
          topo++;
    }
}
```

Aplicações de pilhas

As estruturas de pilha são comumente usadas em algoritmos de gerenciamento de memória (alocação de variáveis na memória e retorno de procedimentos), compiladores e sistemas operacionais.

A figura abaixo demonstra a situação do uso da pilha em Sistemas Operacionais para controle de chamadas de procedimentos. Quando o procedimento A1 é executado, ele efetua uma chamada a A2, que deve

carregar consigo o endereço de retorno e1. Ao término de A2, o processamento deve retornar ao A1, no devido endereço. Situação idêntica ocorre em A2 e A3. Assim, quando um procedimento termina, é o seu endereço de retorno que deve ser consultado. Portanto, há uma lista implícita de endereços (e0, e1, e2, e3) que deve ser manipulada como uma pilha pelo sistema, onde e0 é o endereço de retorno de A1.



Utilização da pilha em Sistemas Operacionais para controle de chamadas de procedimentos

Fonte: Do próprio autor

Outro exemplo de aplicação da pilha é o funcionamento das calculadoras da HP (Hewlett-Packard). Elas trabalham com expressões pós-fixadas, então para se avaliar uma expressão como (1-2)*(4+5), a mesma pode ser digitada 1 2 – 4 5 + *.

O funcionamento dessas calculadoras é muito simples. Cada operando é empilhado numa pilha de valores. Quando se encontra um operador, desempilha-se o número apropriado de operandos (dois para operadores binários e um para operadores unários), realiza-se a operação devida e empilha-se o resultado. Deste modo, na expressão acima, são empilhados os valores 1 e 2. Quando aparece o operador -, 1 e 2 são desempilhados e o resultado da operação, no caso -1 (= 1 - 2), é colocado no topo da pilha. A seguir, 4 e 5 são empilhados. O operador seguinte, +, desempilha o 4 e o 5 e empilha o resultado da soma, 9. Nessa hora, estão na pilha os dois resultados parciais, -1 na base e 9 no topo. O operador *, então, desempilha os dois e coloca -9 (= -1 * 9) no topo da pilha. A Figura abaixo ilustra este tipo de aplicação.

Passo 1	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Passo 7
1 2 - 4 5 + * ^	1 2 - 4 5 + * ^	1 2 - 4 5 + * ^	1 2 - 4 5 + * ^	1 2 - 4 5 + * ^	1 2 - 4 5 + * ^	1 2 - 4 5 + * ^
Push(1)	Push(2)	n1 = pop( ) n2 = pop( ) push(n1-n2)	Push(4)	Push(5)	n1 = pop( ) n2 = pop( ) push(n1+n2)	n1 = pop( ) n2 = pop( ) push(n1*n2)
1 ⟵ topo	2 ⟵ topo 1	-1 ⟵ topo	4 ⟵ topo -1	5 ⟵ topo 4 -1	9 ⟵ topo -1	-9 ⟵ topo

Utilização da pilha na resolução de operações matemáticas pós-fixadas (calculadora HP)

Fonte: Do próprio autor

A pilha também pode ser utilizada como uma estrutura para a conversão de um número na base decimal para binário. Na Figura abaixo é possível observar que, para se converter um número decimal para binário, é necessário fazer divisões sucessivas por 2 até que o quociente da divisão seja igual a zero. O número binário é, então, definido pelos restos das divisões sucessivas, do último resto até o primeiro. Dessa forma, empilhando cada resto em uma pilha, é possível obter o número binário desempilhando os elementos, já que os números serão exibidos na ordem inversa. Abaixo, no exemplo 2, segue a implementação dessa aplicação.

Utilização da pilha para a conversão de um número na base decimal para binário

Fonte: Do próprio autor

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 4

//prototipos das funcoes
void push (int pilha [MAX], int x);
int pop (int pilha [MAX]);
void exibe (int pilha [MAX]);

int topo;

main(){

    int pilha [MAX], n, r;
    topo = 0;

    printf("\n Digite o numero na base decimal=");
    scanf("%d", &n);
    while (n!=0){
       r = n%2;
       push (pilha, r);
       n = n/2;
    }
       printf ("\n Numero correspondente ao binario=");
       while (topo!=0){
             r = pop (pilha);
             printf("%d", r);
       }
    system ("PAUSE");
}

void exibe (int pilha [MAX])
{
     int i;

     if (topo >= 1){
         for (i=0; i<topo; i++)
             printf ("\n Pilha [%d] = %d",i,pilha [i]);
     }
  else printf ("\n Pilha vazia");
}

int pop (int pilha [MAX])
{
     int x;

     if (topo >= 1)
     {
          x = pilha [topo - 1];
          topo--;
     }
     else printf ("\n Pilha vazia");

    return (x);
}
void push (int pilha [MAX], int x)
{
    if (topo == MAX)
          printf ("\n Pilha Cheia");
    else{
          pilha [topo] = x;
          topo++;
    }
}
```

Quiz

Exercício

Pilhas

INICIAR

Quiz

Exercício Final

Pilhas

INICIAR

Referências

MIZRAHI, V. V. *Treinamento em linguagem C*, São Paulo: Pearson, 2008.

SCHILDT, H. C – Completo e Total, São Paulo: Pearson, 2006.

Avalie este tópico

ANTERIOR
Estruturas

Índice

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.unidCurso=)