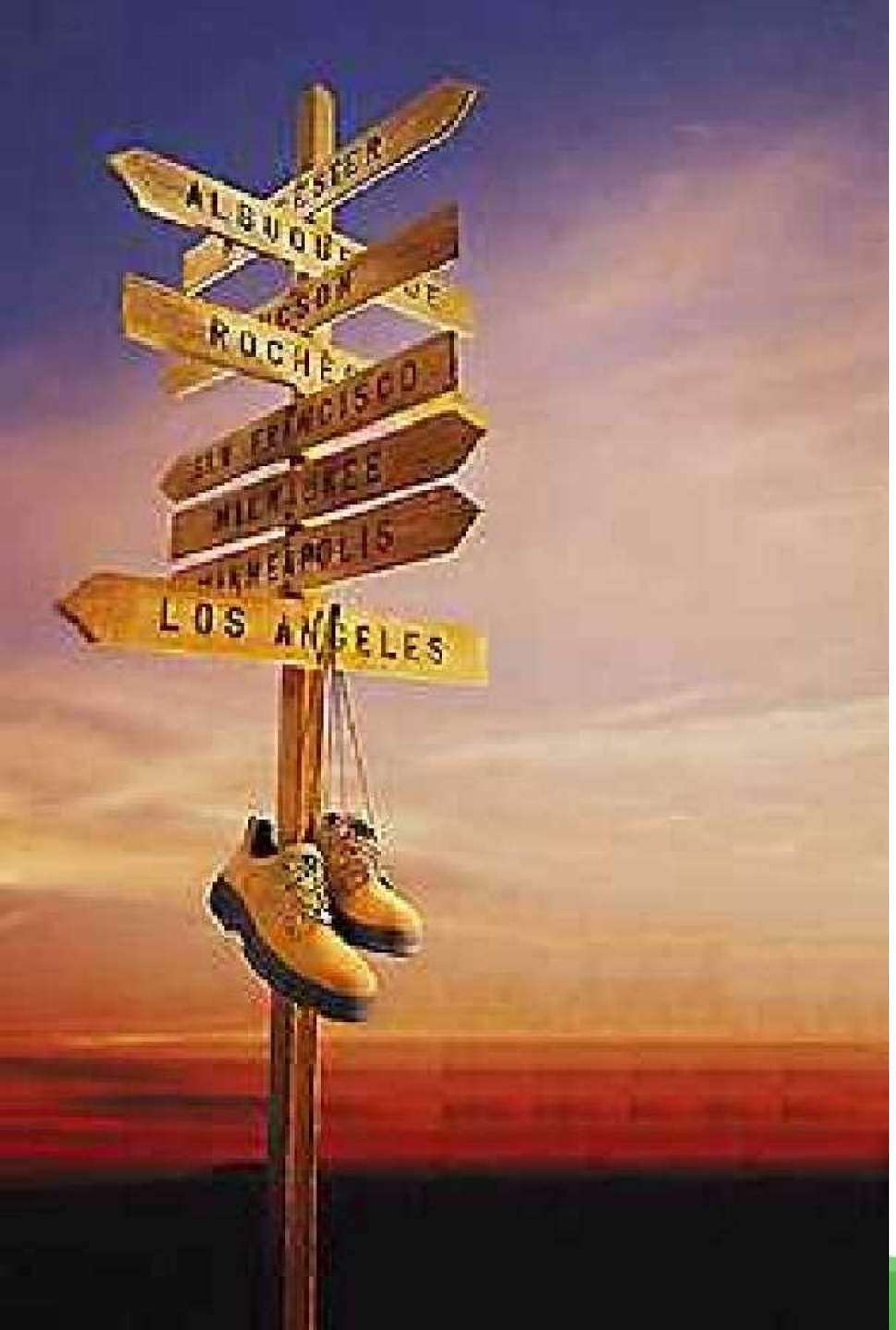
Aula Pilha

Alessandro Calin

ESTRUTURA DE DADOS

alessandrocalin@gmail.com
981524482



"Se o homem não sabe a que porto se dirige, nenhum vento lhe será favorável."

Sêneca

PILHA → É uma estrutura de dados que inicialmente foi feita de forma sequencial (vetor). Hoje temos pilha dinâmica (usando encadeamento).

Segue o critério LIFO (last in First out) (Último a entrar é o primeiro a sair).

Exemplo: Lista de processos em execução no Windows.

Uma **pilha** é uma estrutura de dados com uma coleção linear de elementos que segue a política de acesso **LIFO** (Last In, First Out), ou seja, o último elemento inserido na pilha é o primeiro a ser removido. Imagine uma pilha de pratos: você só pode pegar ou adicionar o prato que está no topo.

- **Operações básicas em uma pilha:**
- **Push**: Adiciona um elemento ao topo da pilha.
- **Pop**: Remove o elemento do topo da pilha.
- **Peek (ou Top)**: Retorna o elemento no topo da pilha sem removê-lo.
- **isEmpty**: Verifica se a pilha está vazia.
- **isFull**: Verifica se a pilha está cheia (caso de pilhas com tamanho limitado).

- Pilhas são estruturas de dados versáteis e têm várias aplicações práticas em computação e programação. Aqui estão alguns exemplos:

1. **Recursão**

- Quando uma função recursiva é chamada, o estado atual da função (incluindo variáveis locais e ponto de retorno) é armazenado em uma pilha de chamadas. Isso permite que, ao finalizar uma chamada recursiva, o programa possa voltar ao estado anterior e continuar a execução.

2. **Navegadores da Web**

- Navegadores utilizam pilhas para gerenciar o histórico de páginas visitadas. Quando você visita uma nova página, ela é empilhada no topo da pilha. Ao pressionar o botão "voltar", o navegador "desempilha" a página atual para retornar à anterior.

3. Desfazer/Refazer em editores de texto

- Muitos editores de texto e software de edição utilizam pilhas para implementar a funcionalidade de "Desfazer" (undo) e "Refazer" (redo). Cada alteração feita é empilhada; ao acionar "Desfazer", a última modificação é removida da pilha e revertida.

4. Avaliação de Expressões

- Pilhas são usadas para avaliar expressões matemáticas, especialmente na conversão de notação infixa (comum) para notação pós-fixa (ou polonesa reversa). Isso é útil em compiladores e calculadoras.

5. Algoritmo de Backtracking

- Em problemas que exigem exploração de múltiplos caminhos, como em labirintos ou resolução de quebra-cabeças, pilhas são usadas para armazenar estados anteriores e retornar a eles se um caminho se mostrar incorreto.

6. Conversão de Números

- A conversão de um número decimal para outra base (como binária ou hexadecimal) pode ser feita usando uma pilha. Os restos da divisão são empilhados e, em seguida, desempilhados na ordem inversa para formar o número na nova base.

7. Análise Sintática em Compiladores

- Compiladores usam pilhas para gerenciar a análise sintática de expressões e sentenças em linguagens de programação, especialmente ao lidar com parênteses, colchetes e chaves para garantir que estão bem formados.

8. Algoritmo de Busca em Profundidade (DFS)

- Em grafos, o algoritmo de busca em profundidade usa uma pilha (implícita, via recursão, ou explícita) para explorar todos os vértices e arestas.

Como implementar uma pilha?

Pelo menos 3 funções:

- Função de empilhar
- Função de desempilhar
- Função principal (main)

Passo a passo para implementação da Pilha:

"Foi colocado no caderno para os alunos copiarem"

Passo a passo para implementar uma pilha.

LIFO – ÚLTIMO A ENTRAR, PRIMEIRO A SAIR.

EXEMPLOS:

Gerenciamento de processos do Sistema Operacional

Processo de binarização

Passo a passo para implementar uma pilha.

Toda pilha é formada por duas variáveis:

Um vetor

Uma variável como nome de topo. A variável topo é um ponteiro que aloca o conteúdo na pilha e verifica se ela está cheia e vazia. Quando a pilha está cheia a variável topo retirar o elemento da pilha.

Passo a passo para implementar uma pilha.

Todo Pilha é uma Struct, pois possui duas variáveis que podem ser de tipos diferentes.

Passo a passo para implementar uma pilha.

1) Importar as Bibliotecas;

2) Definir a Struct;

3) Criar a função empilhar;

3.1) Definir a função empilhar;

3.2) Na função empilhar temos que passar como parâmetro o valor a ser empilhado e um ponteiro para manipular a Struct;

3.3) Verificar se a pilha esta cheia. A pilha está cheia quando o TOPO chegar ao limite (tamanho do vetor)

3.4) Incrementar a variável topo

3.5) Armazenar o valor dentro da pilha.

Passo a passo para implementar uma desempilha.

4) Criar a função desempilha

4.1) E retornável e possui como parâmetro apenas um ponteiro;

4.2) Declarar uma variável local para retirar o valor da pilha e retornar pra main;

4.3) Verificar se a pilha está vazia. A pilha está vazia quando o topo for -1

4.4) Desempilhar o valor do vetor

4.5) Decrementar o topo

4.6) Retornar o valor pra main.

Passo a passo para implementar a main.

5) Criar a função main

5.1) Declarar a Struct na main

5.2) Declarar uma variável para receber o valor digitado pelo usuário (a ser empilhado)

5.3) Iniciar o TOPO com -1

5.4) Chamar a função empilha o número de vezes do tamanho do vetor (terá que ficar dentro de uma estrutura de repetição)

5.5) Fazer uma estrutura de repetição para desempilhar o valor

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

#define TAM 10

typedef struct{
        int vet[TAM];
        int topo;
}pilhavet;
```

```
void empilha(int valor, pilhavet *p){
        if(p -> topo == TAM-1){
                printf("\nPilha Cheia");
                exit(0);
        }
        p -> topo++;
        p -> vet[p -> topo]=valor;
}
```

```
int desempilha(pilhavet *p){
        int aux;
        if(p -> topo <0){
                printf("\npilha vazia");
                exit(1);
        }
        aux = p ->vet [p -> topo];
        p -> topo--;
        return aux;
}
```

```
int main(void){
        pilhavet pilha;
        int valor;
        pilha.topo = -1;
        for (int i=0; i<TAM; i++){
                printf("\nDigite o valor ser empilhado: ");
                scanf("%d",&valor);
                empilha(valor, &pilha);
        }
        for (int i=0; i<TAM; i++){
                printf("\n%d", desempilha(&pilha));
        }
        printf("\n");
        return(0);
}
```