Universidade Federal Rural de Pernambuco
(UFRPE)
- Unidade Acadêmica de Garanhuns-

Pilhas

Estruturas de Dados

Prof. Priscilla Kelly Machado Vieira

2019.1 – Aula 11

Apresentação do Capítulo

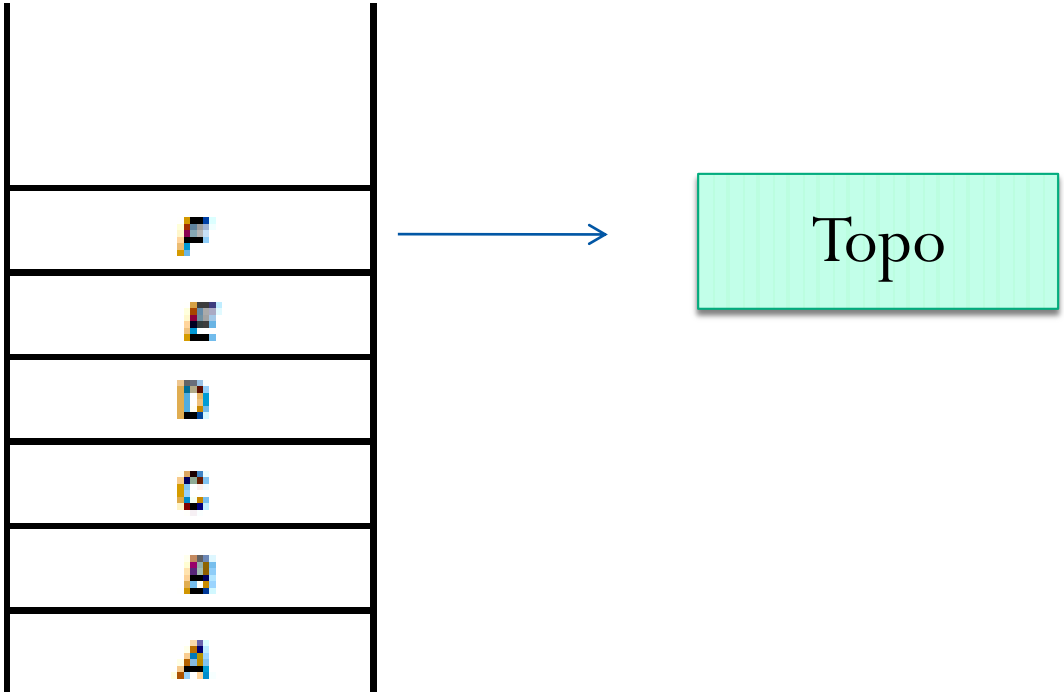
- Introdução
- Conceitos de Pilha
  - Operações
  - Exercícios

Introdução

- Uma pilha, assim como uma lista, é simplesmente uma lista linear de informações
- O que difere uma pilha de uma fila é o mecanismo responsável pelo armazenamento e recuperação dos seus elementos
- É uma coleção de objetos que são inseridos e retirados de acordo com o seguinte princípio: “O último que entra é o primeiro que sai” → (Last In First Out: LIFO)
- Todo acesso é realizado a partir do topo

Conceitos

- Empilhar (*push*) e Desempilhar (*pop*)



Operações

- **Push** (*e*): Insere o objeto e no topo da pilha.
- **Pop**: Remove o elemento do topo da pilha; ocorre um erro se a pilha estiver vazia.
- **Size**(): Retorna o número de elementos na pilha.
- **isEmpty**: Verifica se a pilha está vazia.
- **Top**(): Retorna o elemento do topo da pilha

Estrutura da Pilha

- É possível implementar uma pilha utilizando vetores ou listas
- Com vetores:
  - Se temos n elementos armazenados na pilha, o elemento vet[n-1] representa o do topo

```
struct pilha {
    int n;
    float vet[N];
};
typedef struct pilha Pilha;
```

#define N 50

Inicializa Pilha

- Inicializa a pilha como sendo vazia, ou seja, com o número de elementos igual a zero

```
Pilha* pilha_cria () {
    Pilha* p = (Pilha*)malloc(sizeof(Pilha));
    p -> n = 0; //inicializa com zero elementos
    printf("pilha criada\n");
    return p;
}
```

Inserção

- Sempre verificar se existe espaço para inserção de um novo elemento

```
void pilha_insere (Pilha* p, float v){
    if ((*p).n == N){  //verifica se a capacidade do vetor foi esgotada
        printf ("Capacidade da pilha estourou! \n");
        exit (1);
    }
    /*insere o novo elemento na próxima posição livre*/
    p -> vet[p->n] = v;
    p -> n++;
    printf("inserido. \n");
}
```

Remoção


```
float pilha_remove (Pilha* p) {
    float v;
    if (pilha_vazia(p) == 0){
        printf ("Pilha está vazia!\n");
        exit(1);
    }   //agora retira o elemento do topo da pilha
    v = p -> vet[p-> n-1];
    p -> n--;
    printf("removido.\n");
    return v;
}
```

Libera Memória


```
void pilha_libera (Pilha* p){
    printf("liberando memoria\n");
    free(p);
}
```

Laboratório 10

- Reutilize o código do exercício da aula anterior e implemente uma pilha utilizando ponteiros. Não esqueça de implementar as operações básicas.
  - Push - Inserção
  - Pop - Remoção