ED1 – Pilha

Prof. Andres J. Porfirio

UTFPR

Sumário

- Conceito
- Implementação Estática
- Detalhes para a Implementação
- Exercícios

Pilha

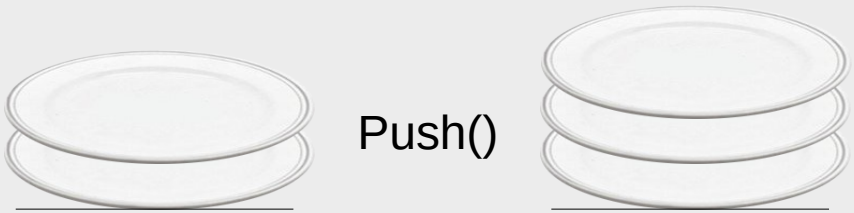
- Conceito:
  - Uma pilha é uma estrutura de dados que possibilita a inserção de remoção de dados;
  - Operações só podem ser realizadas no topo;
    - Não é possível fazer acesso a elementos que não estejam no topo.
  - Conceito similar à uma pilha de pratos;

Pilha

- Conceito:

Pilha

- Conceito:
  - Inserção de elementos (empilhar):
    - O elemento é inserido sempre o do topo da pilha;
    - Operação: Push



A operação push insere
um elemento no topo da pilha.
Ex: em uma pilha de caracteres:
`void push(pilha,'c');`

Pilha

- Conceito:
  - Remoção de elementos (desempilhar):
    - O elemento a ser removido é sempre o do topo da pilha;
    - É o elemento mais novo na estrutura;
    - Operação: Pop

Pop()

A operação pop remove o elemento do topo da pilha. O elemento removido pode ser retornado: Ex: em uma pilha de caracteres a operação pop pode retornar um caractere:

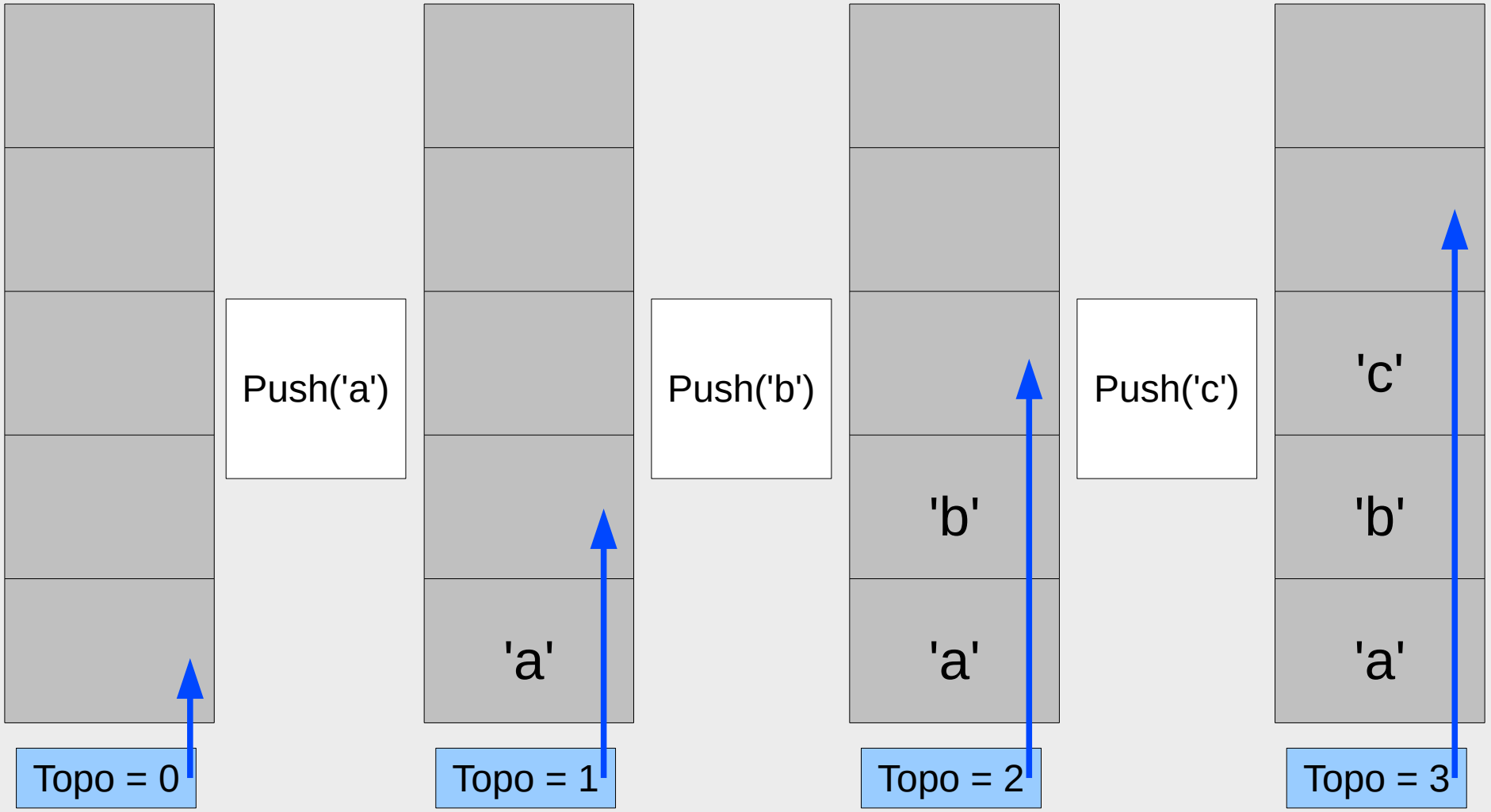
```
char pop(pilha);
```

Implementação Estática

- A implementação de uma pilha estática pode ser feita com o uso de um vetor;
- O tamanho da pilha é fixo;
  - Preferencialmente com o uso de #define.
- O topo da pilha é marcado por uma variável auxiliar que indica primeira posição vazia da pilha;

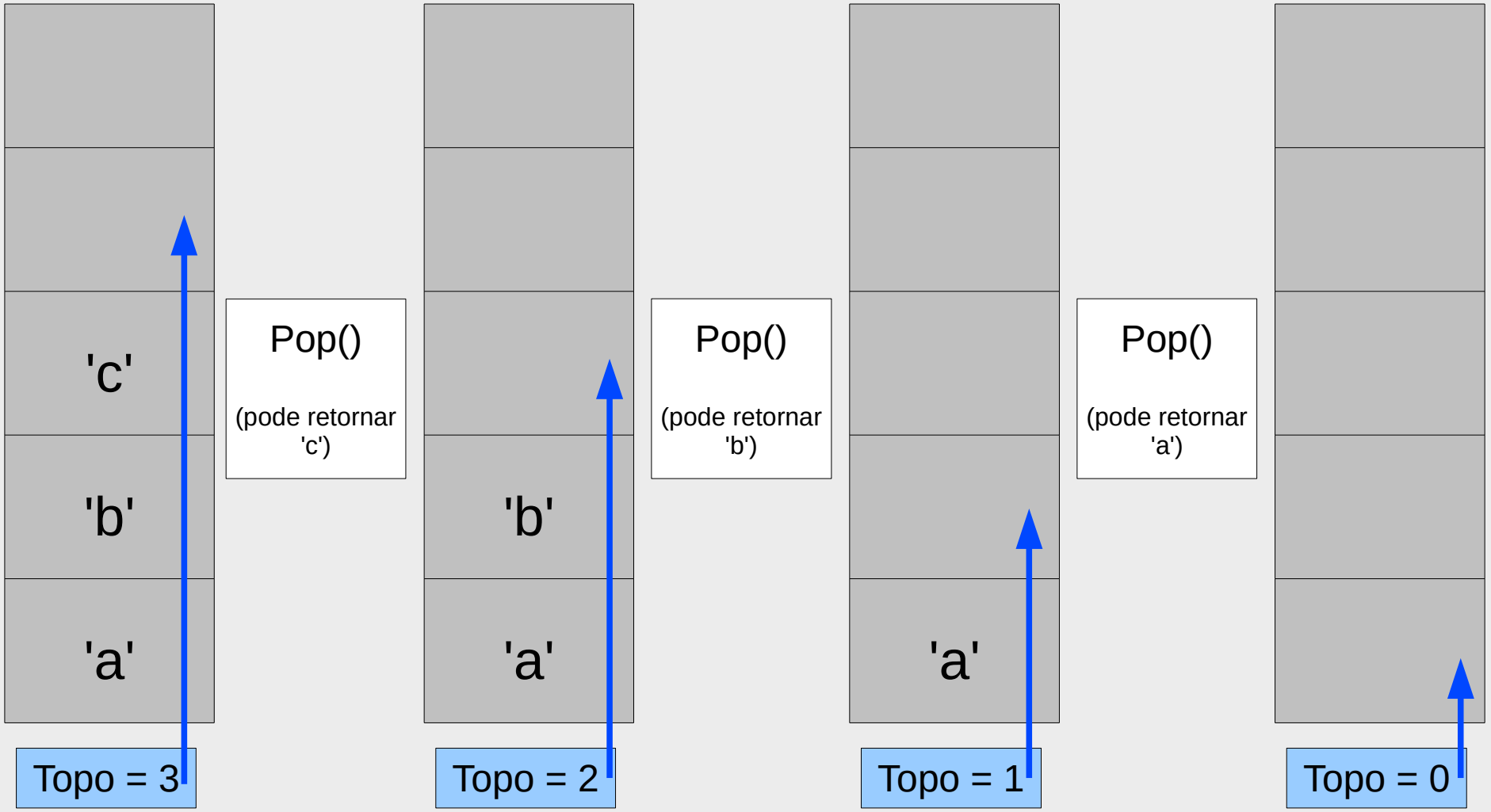
Implementação Estática

- Exemplo, uma pilha de caracteres:



Implementação Estática

- Exemplo, uma pilha de caracteres:



Implementação Estática

- Operações:
  - static_stack_create
    - Aloca memória, atribui 0 para o topo, retorna a Stack recém criada;
  - static_stack_push
    - Verifica se não está cheia, insere um elemento, atualiza o topo;
  - static_stack_pop
    - Verifica se não está vazia, remove um elemento, atualiza o topo;
  - static_stack_is_empty
    - Retorna 1 se a pilha está vazia, 0 caso contrário;
  - static_stack_is_full
    - Retorna 1 se a pilha está cheia, 0 caso contrário;
  - static_stack_peek
    - Retorna o elemento do topo da pilha;
  - static_stack_free
    - Libera a memória alocada em static_stack_create.

Detalhes da Implementação

- Cuidado ao **desempilhar** um dado: sempre verifique se a pilha não está vazia;
  - **`Topo == ?`**
- Cuidado ao **empilhar** um dado: sempre verifique se a pilha não está cheia;
  - **`Topo == ?`**

Exercícios

De acordo com os conceitos abordados nesta aula, implemente uma pilha de caracteres estática. Utilize a seguinte base:

```
#define SIZE 3

typedef struct stack {
    char svet[SIZE];
    int  top;
}Stack;
```

```
void main()
{
    Stack *S = static_stack_create();

    static_stack_push(S, 'a');
    static_stack_push(S, 'b');
    static_stack_push(S, 'c');
    static_stack_push(S, 'd');

    printf("Top: %c\n", static_stack_peek(S));
    static_stack_pop(S);
    printf("Top: %c\n", static_stack_peek(S));
    static_stack_pop(S);
    printf("Top: %c\n", static_stack_peek(S));

    static_stack_pop(S);
    static_stack_pop(S);
    static_stack_pop(S);
}
```

Saída esperada:

Info:inserted:a
Info:inserted:b
Info:inserted:c
Error: stack full
Top: c
Info:removed:c
Top: b
Info:removed:b
Top: a
Info:removed:a
Error: stack empty
Error: stack empty

Referências

- Aula baseada no material de Paulo Feofiloff, IME-USP, 2013.