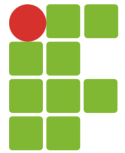
Pilhas

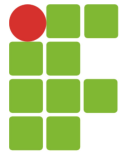
Estruturas de Dados e Algoritmos

Prof. Daniel Saad Nogueira Nunes

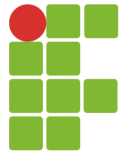
IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário



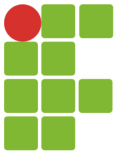
Sumário



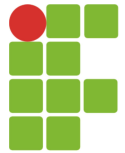
Introdução

Pilhas

- Pilhas são um TAD em qual os elementos são mantidos em uma ordem específica. Esta ordem é a ordem LIFO (Last-in-First-Out)
- A ordem LIFO se caracteriza pelo fato dos últimos elementos a fazerem parte da estrutura, também serão os primeiros elementos a deixarem a estrutura.
- Seu uso é interessante quando é necessário acessar elementos na ordem inversa a de inserção.

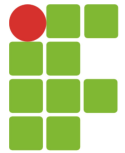


Pilhas

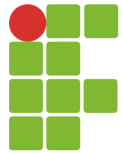


Operações sobre Pilhas

- Algumas das operações suportadas por uma pilha devem ser:
  - Empilhar elementos;
  - Desempilhar elementos;
  - Acessar o topo da pilha;
  - Verificar o tamanho da pilha.
  - Verificar se a pilha está vazia.

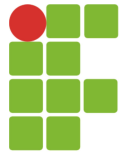


Sumário

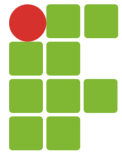


Implementação de pilhas

- Pilhas podem ser implementadas através de vetores dinâmicos ou de listas.

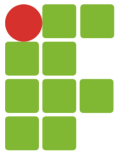


Sumário

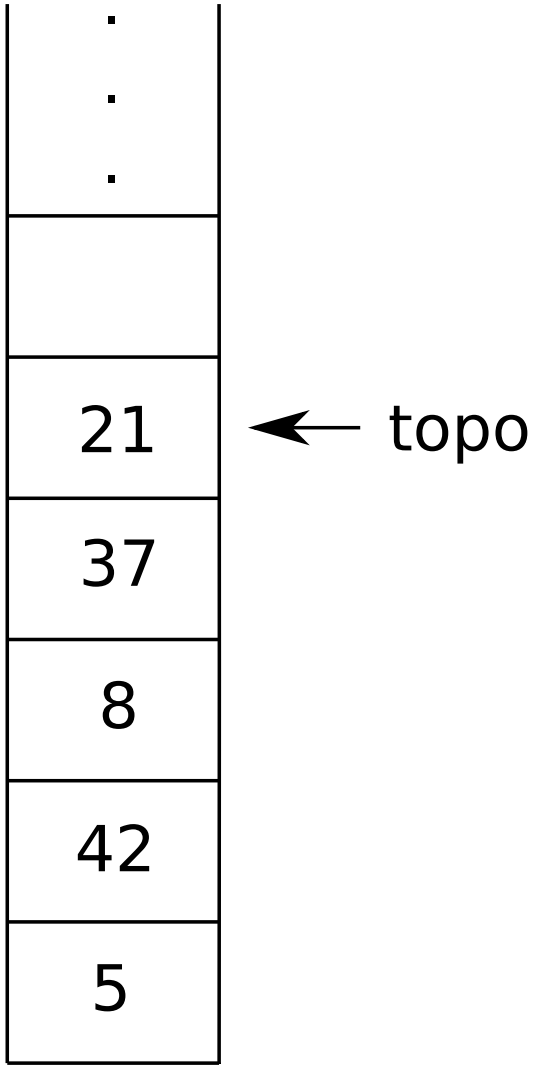


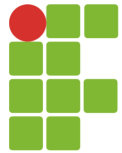
Implementação de Pilhas

- Pilhas podem ser implementadas através de vetores dinâmicos.



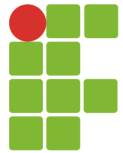
Implementação de Pilhas sobre Vetores





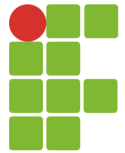
Implementação de Pilhas sobre Vetores

- Empilhar é equivalente a inserir ao final do vetor dinâmico.
- Desempilhar é equivalente a remover o último elemento do vetor dinâmico.
- Acessar o topo é equivalente a acessar o último elemento do vetor dinâmico.
- Verificar o tamanho da pilha é equivalente a verificar o tamanho do vetor dinãmico.
- Verificar se a pilha é vazia é equivalente a verificar se o tamanho do vetor dinâmico é zero.

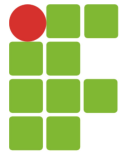


Representação de Pilhas sobre Vetores

- Através da nossa implementação de vetores dinâmicos, implementar uma pilha é trivial.

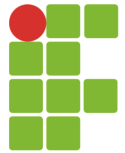


Sumário

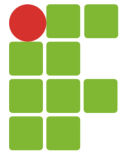


Pilhas: definição

```
6  typedef struct stack_t {
7      dynamic_array *stack_array;
8  } stack_t;
```

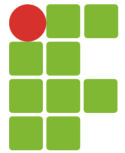


Sumário

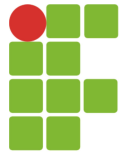


Pilhas: Inicialização

```
4  void stack_initialize(stack_t **stack) {
5      *stack = mallocx(sizeof(stack_t));
6      dynamic_array_initialize(&(*stack)->stack_array);
7  }
```

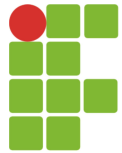


Sumário



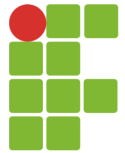
Pilhas: verificar o tamanho

```
31 size_t stack_size(stack_t *stack) {
32     return dynamic_array_size(stack->stack_array);
33 }
```

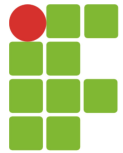


Pilhas: verificar se é vazia

```
27 bool stack_empty(stack_t *stack) {
28     return stack_size(stack) == 0;
29 }
```

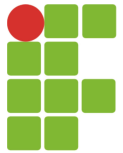


Sumário

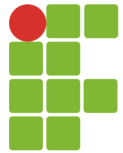


Pilhas: empilhar

```
15 void stack_push(stack_t *stack, int data) {
16     dynamic_array_push_back(stack->stack_array, data);
17 }
```

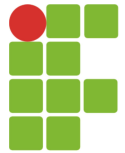


Sumário

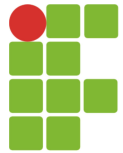


Pilhas: desempilhar

```
23 void stack_pop(stack_t *stack) {
24     dynamic_array_pop_back(stack->stack_array);
25 }
```

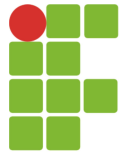


Sumário

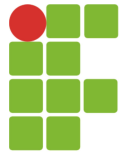


Pilhas: acessar o topo

```
19 int stack_top(stack_t *stack) {
20     return dynamic_array_back(stack->stack_array);
21 }
```

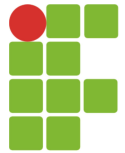


Sumário

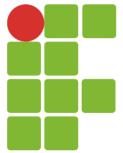


Pilhas: limpeza

```
9  void stack_delete(stack_t **stack) {
10     dynamic_array_delete(&(*stack)->stack_array);
11     free(*stack);
12     *stack = NULL;
13 }
```



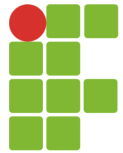
Sumário



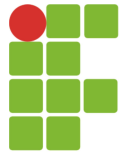
Pilhas: análise

Complexidade das Operações

Operação	Complexidade
Empilhar	$\Theta(1)$
Desempilhar	$\Theta(1)$
Verificar topo	$\Theta(1)$

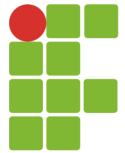


Sumário



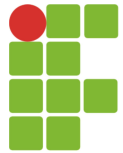
Implementação de Pilhas

- Pilhas podem ser implementadas por meio de estruturas auto-referenciadas.
- Uma das estruturas que podem prover as funcionalidades de uma pilha é uma lista ligada.
- Basta utilizar a lista de uma maneira muito específica para simular uma pilha.

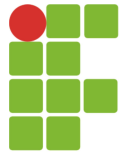


Implementação de Pilhas sobre Listas

- Utilizando listas, a operação de verificar se a pilha está vazia equivale à verificar se a lista está vazia.
- Para empilhar um elemento, insere-se um elemento na cabeça.
- Para desempilhar um elemento, retira-se da cabeça.
- Para acessar o topo da pilha, a cabeça deve ser acessada.



Sumário



Exemplo

```c
#include "stack.h"
#include <stdio.h>
#include <string.h>

int main(void) {
    stack_t *stack;
    stack_initialize(&stack);
    for (int i = 0; i < 1000000; i++) {
        stack_push(stack, i);
    }
    while (!stack_empty(stack)) {
        printf("%d\n", stack_top(stack));
        stack_pop(stack);
    }
    stack_delete(&stack);
    return 0;
}
```