

# Pilhas – *Stack*

## Programação de computadores II

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC  
Campus Florianópolis  
`renan.starke@ifsc.edu.br`

22 de setembro de 2016



**INSTITUTO FEDERAL**  
**SANTA CATARINA**

Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
**INSTITUTO FEDERAL DE SANTA CATARINA**

# Tópicos da aula

1 Introdução

2 Pilhas

3 Exemplo

4 Exercícios

1 Introdução

2 Pilhas

3 Exemplo

4 Exercícios

- Entender o conceito de pilhas – *stack*
- Aprender a utilizar esta estrutura de dados
- Aprender a implementar uma pilha simples
- Utilizar o conceito de “dados abstratos e estruturados” para fornecer funções simples de manipulação de pilhas

1 Introdução

2 Pilhas

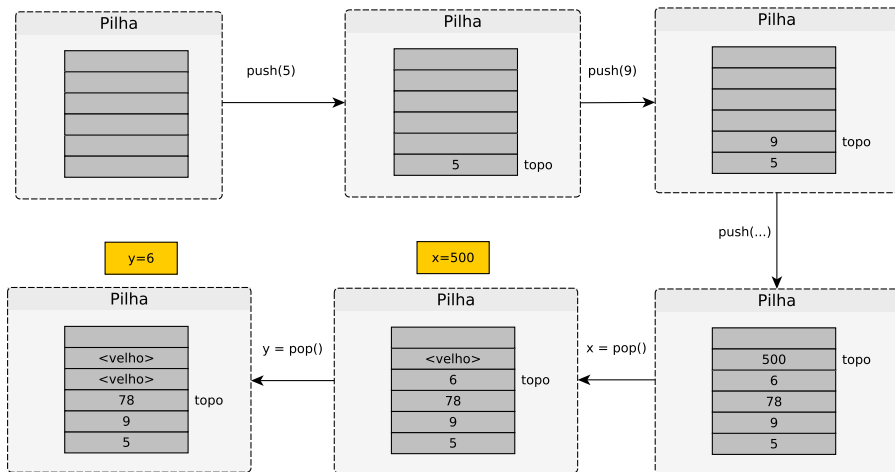
3 Exemplo

4 Exercícios

Pilha – *stack* – é uma estrutura de dados simples onde os dados são “empilhados”. Idealmente implementa o conceito de LIFO – *Last-in, First-out*: último a entrar é o primeiro a sair.

Funções básicas:

- *push*: adiciona um elemento
- *pop*: remove um elemento



## Funções adicionais:

- *topo*: retorna o valor do elemento do topo da pilha
- *imprimir*: imprimir, sem retirar, todos os elementos da pilha
- *contem*: procura um item
- *tamanho*: retorna tamanho da pilha
- *vazia*: retorna se pilha está vazia



1 Introdução

2 Pilhas

3 Exemplo

4 Exercícios

# Exemplo – pilha.h

```
#ifndef PILHA_H_INCLUDED
#define PILHA_H_INCLUDED

#define TAMANHO_DADOS_PILHA 100

typedef struct pilhas pilha_t;

pilha_t * cria_pilha (void);

void push(int dado, pilha_t *pilha);
int pop(pilha_t *pilha);

#endif // PILHA_H_INCLUDED
```

# Exemplo – pilha.c: struct pilha

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

#include "pilha.h"

struct pilhas {
    int topo;
    int data[TAMANHO_DADOS_PILHA];
};
```

# Exemplo – pilha.c: cria\_pilha

```
//cria uma pilha para inteiros
pilha_t * cria_pilha (void)
{
    pilha_t *pilha = (pilha_t*)malloc(sizeof(pilha_t));

    pilha->topo = 0;

    return pilha;
}
```

# Exemplo – pilha.c: push

```
//adiciona elemento
void push(int dado, pilha_t *pilha)
{
    int topo = pilha->topo;

    if (topo > TAMANHO_DADOS_PILHA) {
        fprintf(stderr, "Tamanho maximo da pilha atingido!\n");
        exit(EXIT_FAILURE);
    }

    pilha->data[topo] = dado;
    pilha->topo++;
}
```

# Exemplo – pilha.c: pop

```
//retira elemento do topo
int pop(pilha_t *pilha)
{
    int topo = pilha->topo;

    if (topo < 0 || topo > TAMANHO_DADOS_PILHA){
        fprintf(stderr, "Pilha corrompida!\n");
        exit(EXIT_FAILURE);
    }

    if (topo == 0) {
        fprintf(stderr, "pop() em pilha vazia!\n");
        return 0;
    }

    pilha->topo--;
    return pilha->data[topo - 1];
}
```

# Exemplo – pilha.c: main

```
int main()
{
    int x = 0;
    pilha_t *pilha;

    //cria uma pilha
    pilha = cria_pilha();

    //empilha dados
    push(5,pilha);
    push(10,pilha);
    push(33,pilha);
    push(60,pilha);

    //desempilha
    x = pop(pilha);
    printf("main: pop() ---> %d\n", x);

    x = pop(pilha);
    printf("main: pop() ---> %d\n", x);
    x = pop(pilha);
    printf("main: pop() ---> %d\n", x);

    (...)

    free(pilha);

    return 0;
}
```

1 Introdução

2 Pilhas

3 Exemplo

4 Exercícios



Baseando-se na implementação exemplo desta aula, implemente uma pilha através de uma lista simplesmente encadeada com as seguintes interfaces:

- *push*: empilha um elemento
- *pop*: desempilha um elemento
- *topo*: retorna o valor do elemento do topo da pilha
- *imprimir*: imprimir, sem retirar, todos os elementos da pilha
- *tamanho*: retorna tamanho da pilha
- *vazia*: retorna se pilha está vazia