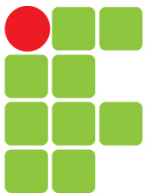
Pilhas – *Stack*

Programação de computadores II

Prof. Renan Augusto Starke

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

19 de março de 2018



INSTITUTO FEDERAL
SANTA CATARINA

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

Tópicos da aula

Tópico

Objetivos

- 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

Tópico

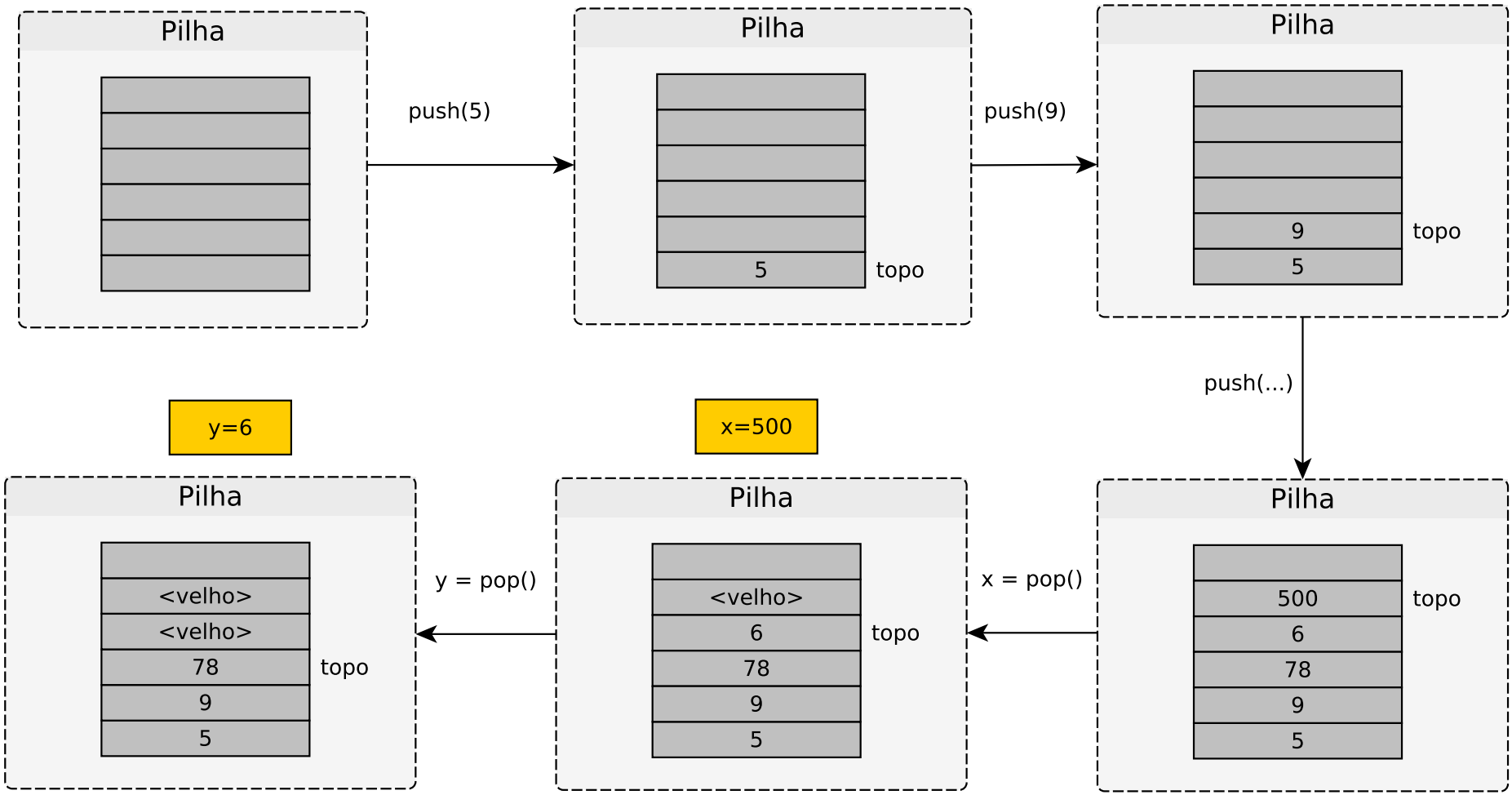
Pilhas

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

Pilhas


Pilha
push(5)
Pilha
5
topo
push(9)
Pilha
9
5
topo
push(...)
Pilha
500
6
78
9
5
topo
x=500
x = pop()
Pilha
<velho>
6
78
9
5
topo
y=6
y = pop()
Pilha
<velho>
<velho>
78
9
5
topo

Pilhas

Funções adicionais:

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

Tópico

Exemplo – pilha.h

```c
#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

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

#include "pilha.h"

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

Exemplo – pilha.c: cria_pilha

```c
//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

```c
//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

```c
//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

```c
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;
}
```

Tópico

Exercício

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

- *pilha_t * cria_pilha()*: cria uma pilha genérica (aceita qualquer ponteiro de dados) vazia
- *push (void *dado)*: empilha um elemento
- *void * pop()*: desempilha um elemento
- *void * topo*:retorna o valor do elemento do topo da pilha
- *int tamanho()*: retorna tamanho da pilha
- *int vazia()*: retorna se pilha está vazia
- Você deve construir sua pilha utilizando o conceito de dados abstratos:
  - typedef struct pilhas pilha_t
  - pilha.c
  - pilha.h

Exercício

Após a implementação da pilha genérica através de uma lista encadeada:

- Modifique o exemplo dos dados, implementado com lista encadeada, para usar sua pilha.
  - A função de leitura de arquivo deve agora retornar uma pilha