Estrutura de Dados

Aula 06 – Pilha e Fila

Prof. Dr. Daniel Vecchiato

Agenda

- Pilha
  - Introdução
  - Funcionamento
  - Aplicações
  - Implementação
  - Exercícios
- Fila
  - Introdução
  - Funcionamento
  - Aplicações
  - Implementação
  - Exercícios

Pilha - Introdução

- Pilha (*stack*)
  - É uma lista linear
  - Um tipo de lista especializada por ter uma característica própria
    - O primeiro elemento inserido será o último a ser removido

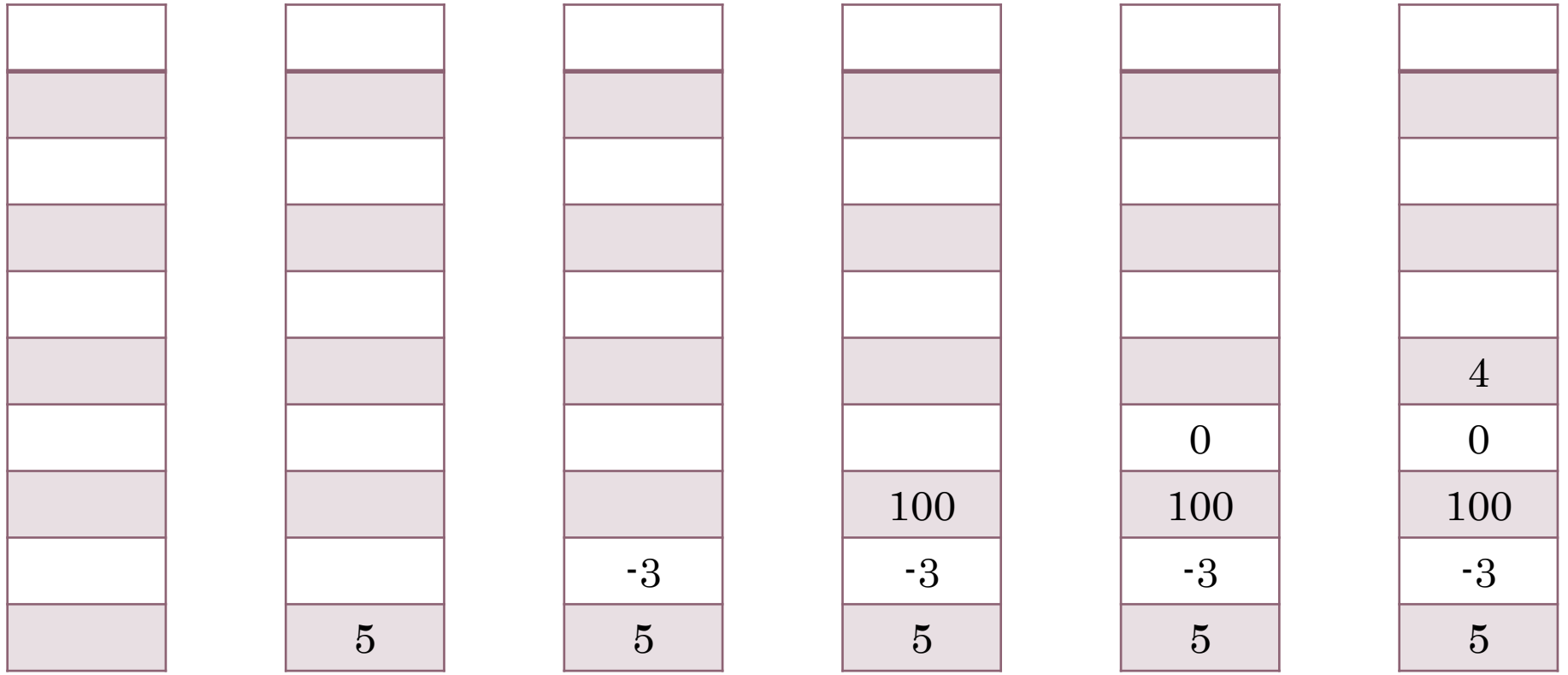
- LIFO: *last in, first out*
  - Um novo elemento sempre é inserido no topo da lista
  - O acesso aos elementos também sempre é feito pelo topo da lista
  - UEPS: Último a entrar, primeiro a sair

Pilha - Introdução

- Operações básicas
  - Push
    - Empilhar
    - Inserção de um novo elemento (sempre no topo)
  - Pop
    - Desempilhar
    - Remove um elemento (sempre do topo)
  - Peek (ou Top)
    - Consulta o elemento do topo da lista, mas não o remove

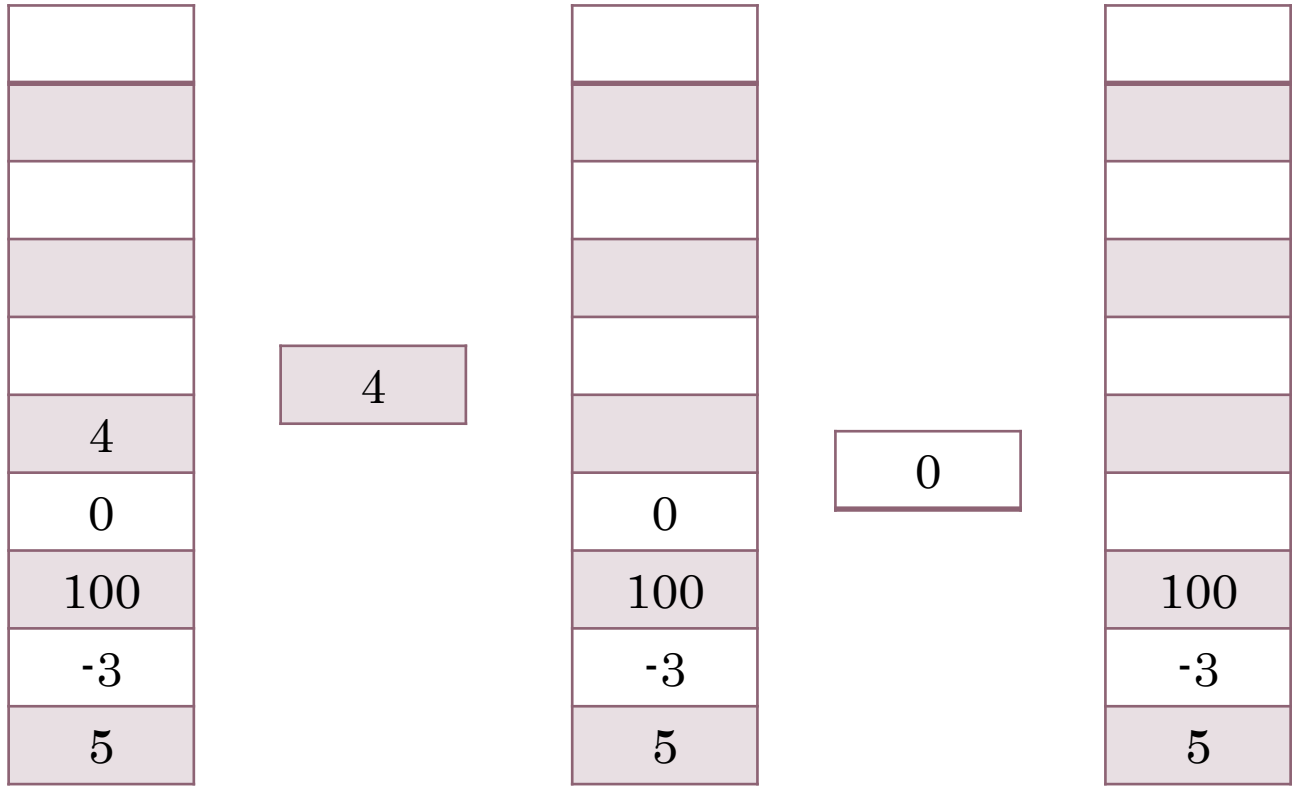
Pilha - Funcionamento

- Push
  - Realização de 5 push com os elementos 5, -3, 100, 0 e 4



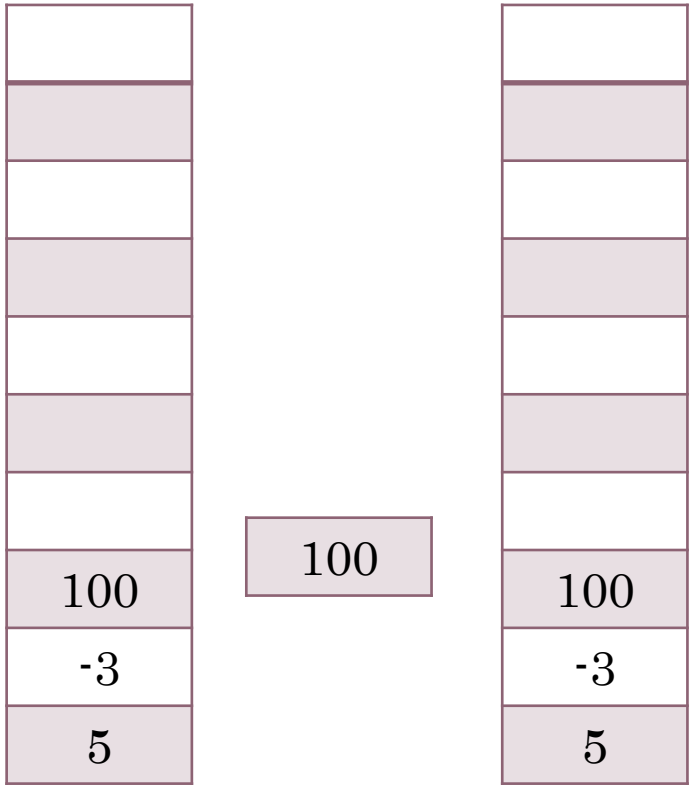
Pilha - Funcionamento

- Pop
  - Realização de 2 pop, obtendo os valores 4 e 0



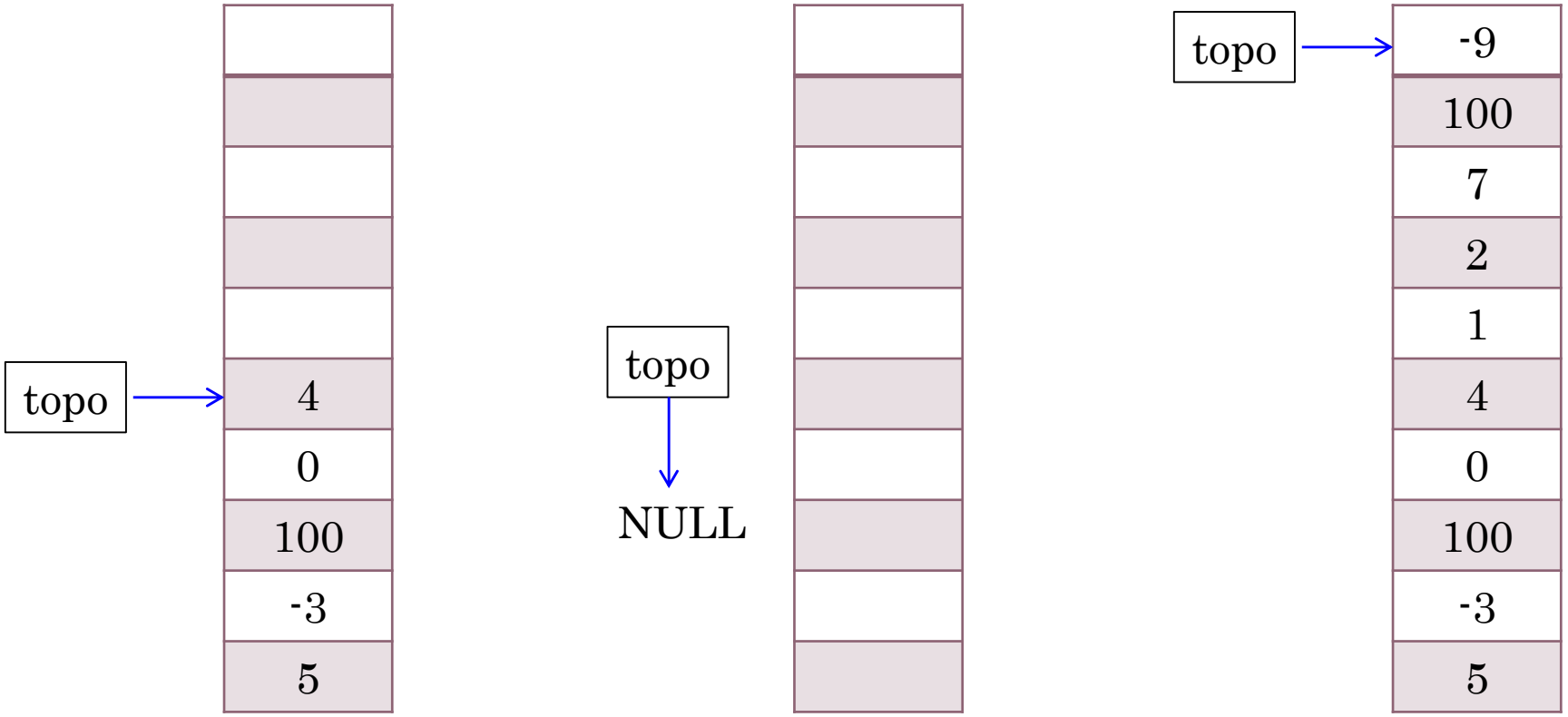
Pilha - Funcionamento

- Peek
  - Realização de 1 peek, obtendo o valor 100 (mantendo-o na lista)



Pilha - Funcionamento

- Para realizar as operações, é necessário um ponteiro para o topo da lista
  - Na pilha vazia, topo aponta para NULL
  - Dependendo da implementação, a pilha pode ter um tamanho limitado. Excedendo o tamanho ocorrerá um *stackoverflow*



Pilha - Funcionamento

- Operações
  - iniciarPilha (pilha)
    - Inicia uma pilha
  - isVazia (pilha)
    - Verifica se a pilha está vazia
  - push (pilha, x)
    - Insere um elemento X na pilha
  - pop (pilha, x)
    - Remove o elemento do topo da pilha, copiando-o em X
  - peek (pilha, x)
    - Consulta o elemento do topo da pilha, copiando-o em X
  - tamanho (pilha)
    - Retorna quantos elementos há na lista

Pilha - Aplicações

- Inverso de uma string
- Desfazer & Refazer
- Avaliação de
  - expressões regulares
  - sintaxe em linguagem de programação
  - expressões aritméticas
- Chamadas de funções (inclusive recursivas)
- Gerenciamento de memória

Pilha - Implementação

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

typedef struct {
    int chave;
} TItem;

typedef struct celula {
    struct celula *pProx;
    TItem item;
} TCelula;

typedef struct {
    TCelula *pTopo;
    int tamanho;
} TPilha;
```

Pilha - Implementação

```
void iniciarPilha (TPilha *pPilha);
int isVazia (TPilha *pPilha);
int push (TPilha *pPilha, TItem x);
int pop (TPilha *pPilha, TItem *pX);
int peek (TPilha *pPilha, TItem *pX);
int tamanho (TPilha *pPilha);
```

Pilha - Implementação

```
void iniciarPilha (TPilha *pPilha) {
    pPilha->pTopo = NULL;
    pPilha->tamanho = 0;
}

int isVazia (TPilha *pPilha) {
    return pPilha->pTopo == NULL;
}
```

Pilha - Implementação

```
int push (TPilha *pPilha, TItem x) {
    // A mesma ideia da inserção no início
    // da lista simplesmente encadeada
}

int pop (TPilha *pPilha, TItem *pX) {
    // A mesma ideia da remoção no início
    // da lista simplesmente encadeada
}

int peek (TPilha *pPilha, TItem *pX) {
    // Semelhante ao pop, mas sem remover o item da lista
}

int tamanho (TPilha *pPilha) {
    return pPilha->tamanho;
}
```

Pilha - Implementação

```
int main() {
    TPilha pilha;
    iniciarPilha (&pilha);
    printf("Vazia: %s\n", isVazia(&pilha) == 1 ? "SIM":"NAO");

    TItem item1, item2, item3;
    item1.chave = 10;
    item2.chave = -5;
    item3.chave = 20;
    push (&pilha, item1);
    push (&pilha, item2);
    push (&pilha, item3);
    printf("Vazia: %s\n", isVazia(&pilha) == 1 ? "SIM":"NAO");

    TItem itemTeste;
    pop (&pilha, &itemTeste);
    printf("Elemento removido: %d\n", itemTeste.chave);
    peek (&pilha, &itemTeste);
    printf("Elemento no topo: %d\n", itemTeste.chave);
    printf("Elementos existentes: %d\n", tamanho(&pilha));
}
```

Pilha - Implementação

```
Vazia: SIM
Vazia: NAO
Elemento removido: 20
Elementos existentes: 2
Elemento no topo: -5
```

Pilha - Exercícios

- Implemente as funções:
  - push
  - pop
  - peek

Fila - Introdução

- Fila (*queue*)
  - Outra variação da lista encadeada
  - Característica principal
    - o primeiro elemento inserido será o primeiro a ser removido
- FIFO: *first in, first out*
  - Um novo elemento sempre é inserido no fim da lista
  - O acesso aos elementos sempre é feito pelo início da lista

Fila - Introdução

- Operações básicas
  - Enqueue (enfileira)
    - Inserção de um novo elemento (no fim da lista)
  - Dequeue ("desenfileira”)
    - Remove um elemento (sempre do início)
  - Peek
    - Consulta o elemento do início da lista, mas não o remove

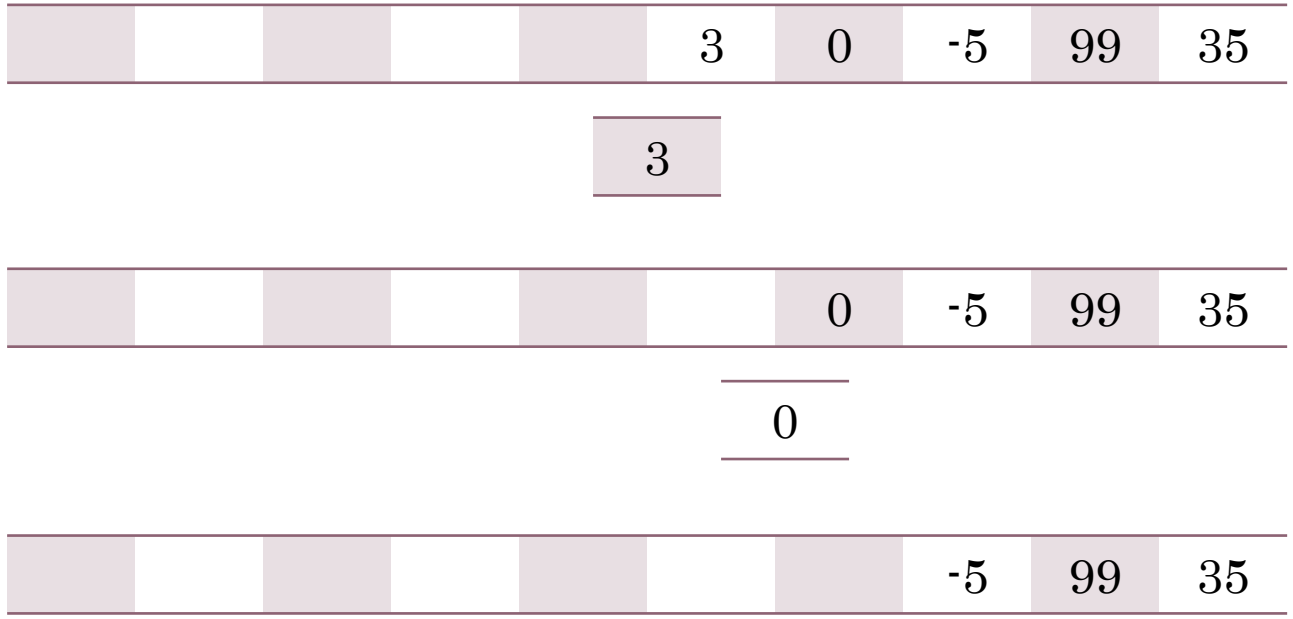
Fila - Funcionamento

- Enqueue
  - Inserção de 5 elementos na fila: 3, 0, -5, 99, 35

									3
								3	0
							3	0	-5
						3	0	-5	99
					3	0	-5	99	35

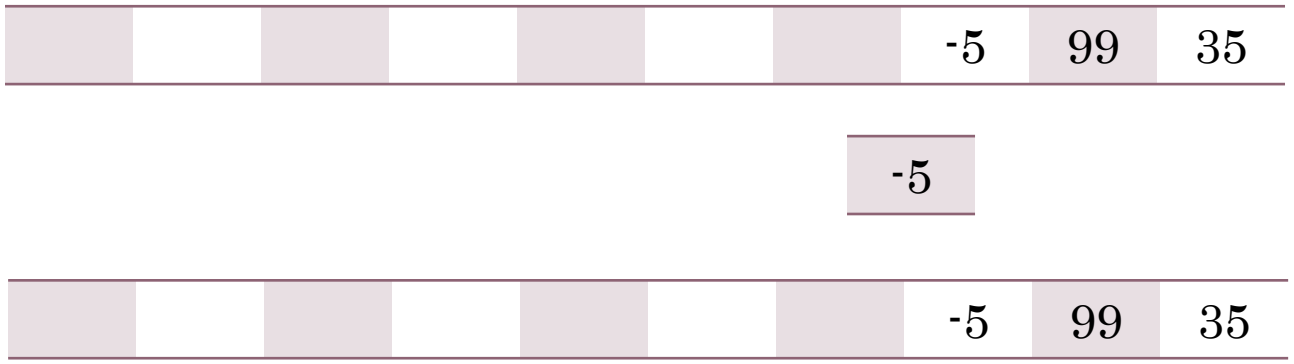
Fila - Funcionamento

- Dequeue
  - Remoção de 2 elementos da fila, obtendo os valores 3 e 0



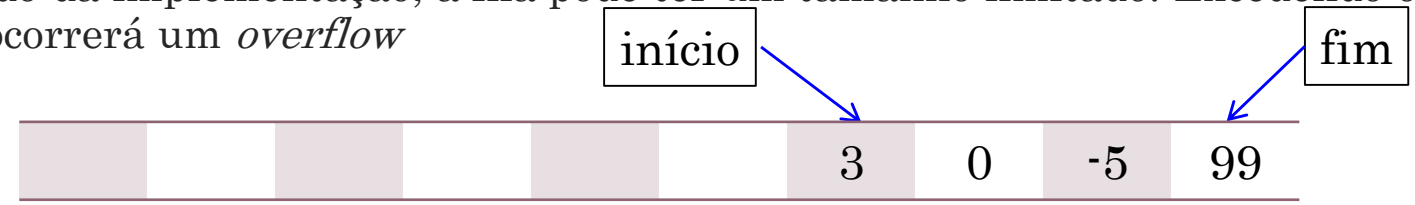
Fila - Funcionamento

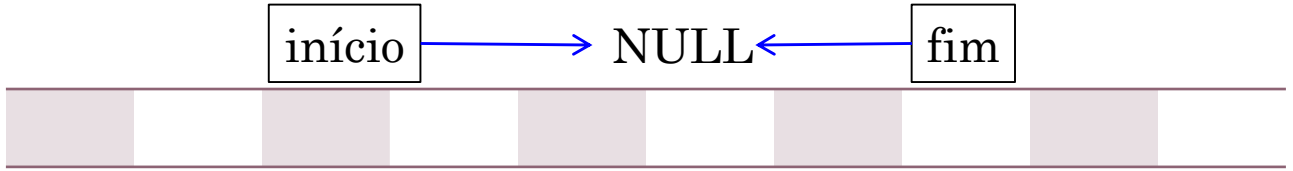
- Peek
  - Realização de 1 peek, obtendo o valor -5 (mantendo-o na lista)

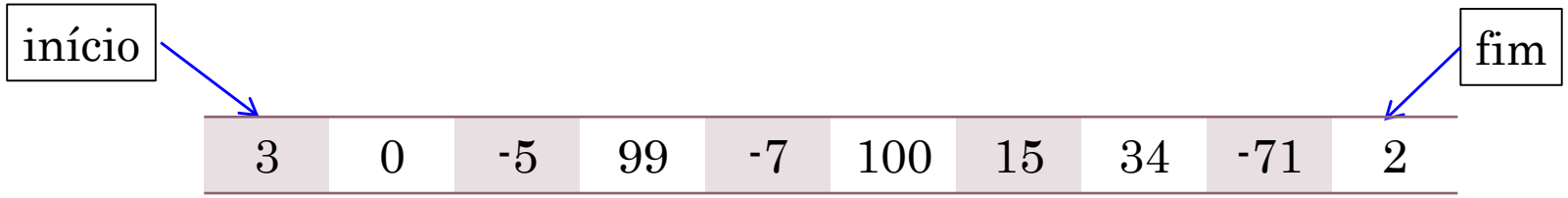


Fila - Funcionamento

- Para realizar as operações, é necessário um ponteiro para o início e outro para o final da fila
  - Na lista vazia, o ponteiro para o início e fim da fila aponta para NULL
  - Dependendo da implementação, a fila pode ter um tamanho limitado. Excedendo o tamanho ocorrerá um *overflow*







Fila - Funcionamento

- Operações
  - iniciarFila (fila)
    - Inicia uma fila
  - isVazia (fila)
    - Verifica se a fila está vazia
  - enqueue (fila, x)
    - Insere um elemento X no fim da fila
  - dequeue (fila, x)
    - Remove o elemento do início da fila, copiando-o em X
  - peek (fila, x)
    - Consulta o elemento do início da fila, copiando-o em X
  - tamanho (fila)
    - Retorna quantos elementos há na lista

Fila - Aplicações

- No sistema operacional
  - Fila de impressão
  - Fila de processamento
- Transferência de dados
- Lista de espera
- Controle de requisições

Fila - Implementação

- Possui a mesma estrutura de uma lista simplesmente encadeada
- A mesma lógica de inserção e remoção deve ser implementada, considerando
  - Inserção sempre no fim da fila
  - Remoção sempre no início da fila

Fila - Exercícios

- Implemente uma lista do tipo fila
- Faça uma função para excluir todos os itens da lista
- Deve conter as operações:

```
void iniciarFila (TFila *fila);
int isVazia (TFila *fila);
int enqueue (TFila *fila, TItem x);
int dequeue (TFila *fila, TItem *pX);
int peek (TFila *fila, TItem *pX);
int tamanho (TFila *fila);
int limpar (TFila *fila);
```

Fila - Exercícios

- Teste a fila com o código abaixo:

```
int main() {
        TFila fila;
        iniciarFila (&fila);
        TItem item1, item2, item3;
        item1.chave = 10;
        item2.chave = -5;
        item3.chave = 20;
        enqueue (&fila, item1);
        enqueue (&fila, item2);
        enqueue (&fila, item3);
        printf("Vazia: %s\n", isVazia(&fila) == 1 ? "SIM":"NAO");
        TItem itemTeste;
        dequeue (&fila, &itemTeste);
        printf("Elemento removido: %d\n", itemTeste.chave);
        printf("Elementos existentes: %d\n", tamanho(&fila));
        peek (&fila, &itemTeste);
        printf("Elemento no início: %d\n", itemTeste.chave);
}
```

- Deve ser impresso:

```
Vazia: NAO
Elemento removido: 10
Elementos existentes: 2
Elemento no início: -5
```

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura
Daniel Avila Vecchiato

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)