

Estruturas de Dados

Prof. Rodrigo Martins
rodrimartins2005@gmail.com

Cronograma da Aula

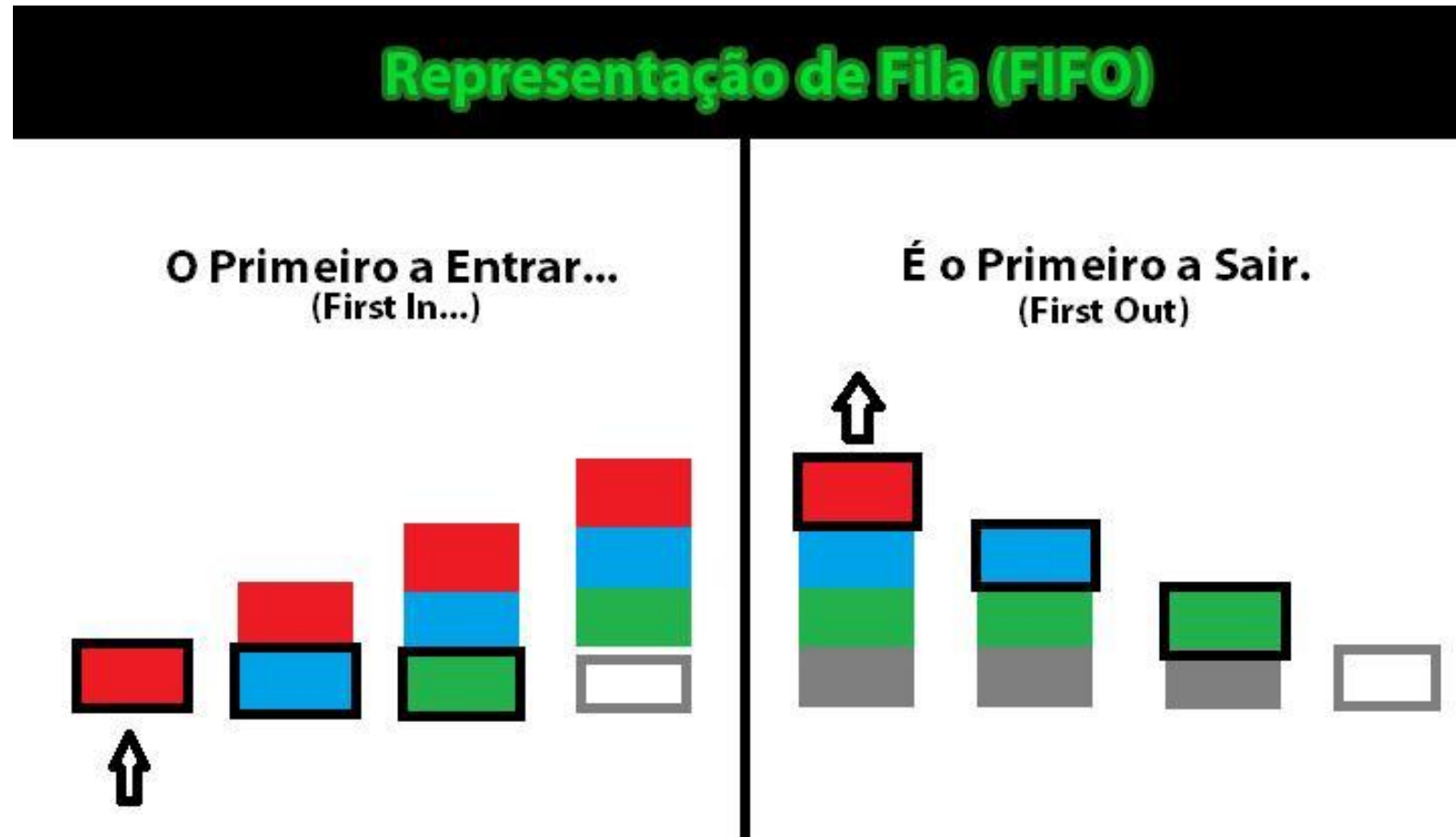
- Filas
- Exemplos
- Exercícios

Fila

- Uma fila é uma estrutura na qual os dados são inseridos em um extremo e retirados no outro extremo.
- São também chamadas de FIFO (“First In First Out” = “Primeiro a Entrar e o Primeiro a Sair”) ou LIFO (“Last In Last Out” = “Último a Entrar e o Último a Sair”).

Fila

- Sua representação pode ser vista abaixo:



Filas – Exemplos de aplicação

- **Filas de Impressão**

- Em uma situação em que várias impressoras compartilham um sistema, os documentos a serem impressos podem ser armazenados em uma fila.
- Cada trabalho de impressão é enfileirado e atendido na ordem em que foi recebido (FIFO), garantindo que os documentos sejam impressos na ordem correta.

Filas – Exemplos de aplicação

- **Processamento de Tarefas**

- Filas são utilizadas em sistemas que executam várias tarefas assíncronas.
- Por exemplo, um servidor web pode receber várias requisições dos clientes e enfileirá-las para processar cada requisição na ordem em que foi recebida.
- Ajudando a gerenciar o processamento e alocação de recursos para cada tarefa de forma eficiente.

Filas – Exemplos de aplicação

- **Sistemas de Mensagens**

- Em sistemas de mensagens assíncronos, as filas são usadas para armazenar mensagens enviadas de um serviço para outro.
- Por exemplo, em um sistema de microserviços, uma fila pode ser usada para que um serviço envie mensagens para outro, permitindo um fluxo de dados eficiente entre diferentes partes do sistema.

Filas – Exemplos de aplicação

- **Gerenciamento de Prioridades**

- Filas podem ser usadas para implementar políticas de gerenciamento de prioridades em sistemas, como a execução de tarefas de alta prioridade antes das de baixa prioridade.

Filas – Exemplos de aplicação

- **Algoritmos de Busca e Inteligência Artificial**
 - Filas são amplamente usadas em algoritmos de busca, como busca em largura (breadth-first search).
 - Em problemas de inteligência artificial, filas podem ser usadas para gerenciar estados e caminhos a serem explorados durante a busca de soluções.

Filas – Exemplos de aplicação

- **Sistemas de Controle de Processos**

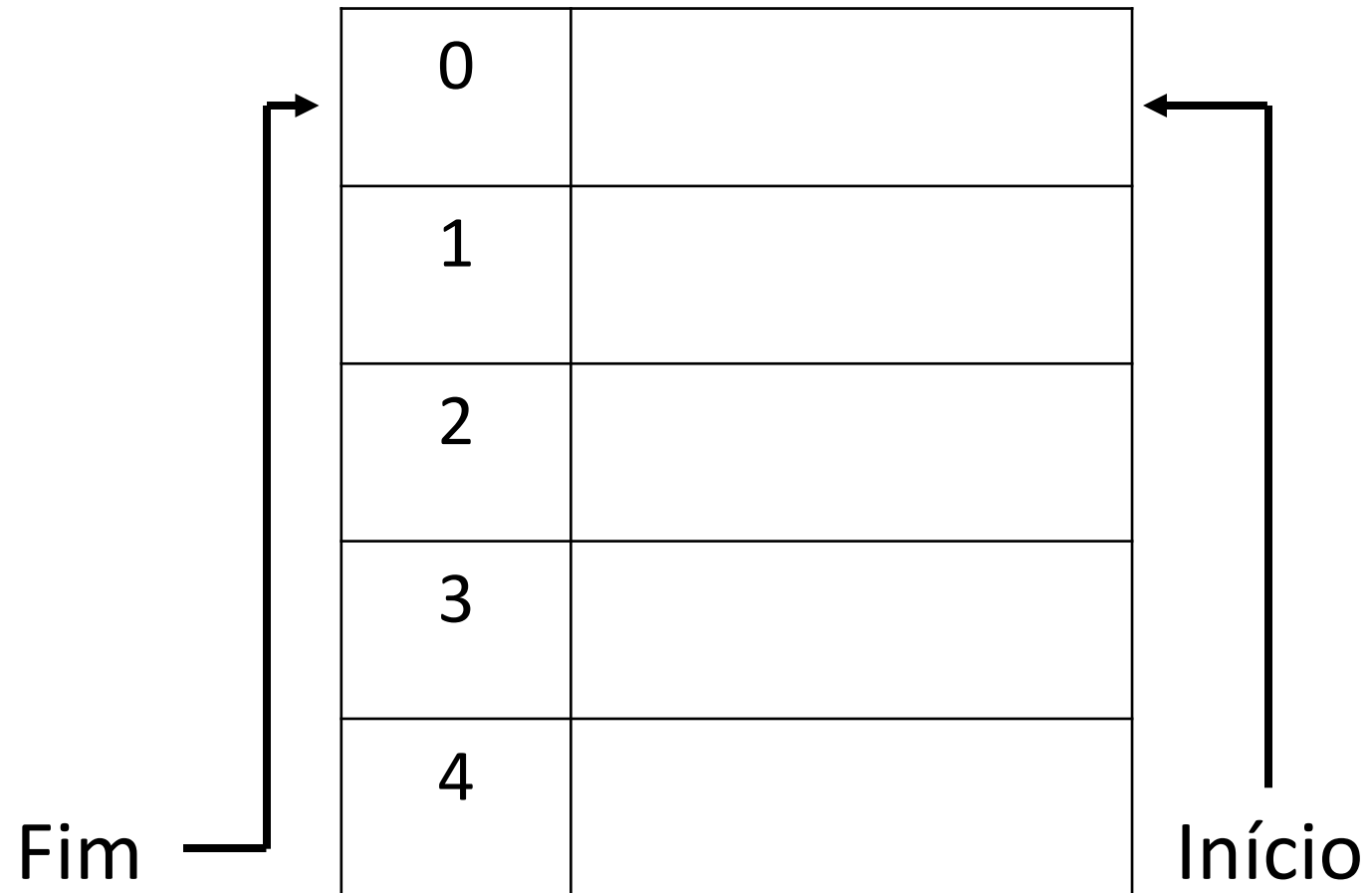
- Filas são utilizadas em sistemas operacionais para gerenciar processos em execução.
- Por exemplo, um sistema pode manter uma fila de processos prontos para execução ou uma fila de processos aguardando I/O.

Fila

- A implementação estática desta estrutura requer a utilização de um vetor, onde um elemento é inserido no final da fila (a direita) e retirado do início da fila (a esquerda).
- Esta implementação é dificultada devido ao fato da fila se locomover dentro do vetor. Se nada for feito para controlar esta locomoção, a fila atingirá o final do vetor (lado direito) e a próxima inserção não poderá ser feita, mesmo que haja espaço do lado esquerdo do vetor.
- Uma forma de resolver este problema é: sempre que um elemento for removido (lado esquerdo) deslocar os demais elementos da fila uma casa a esquerda.

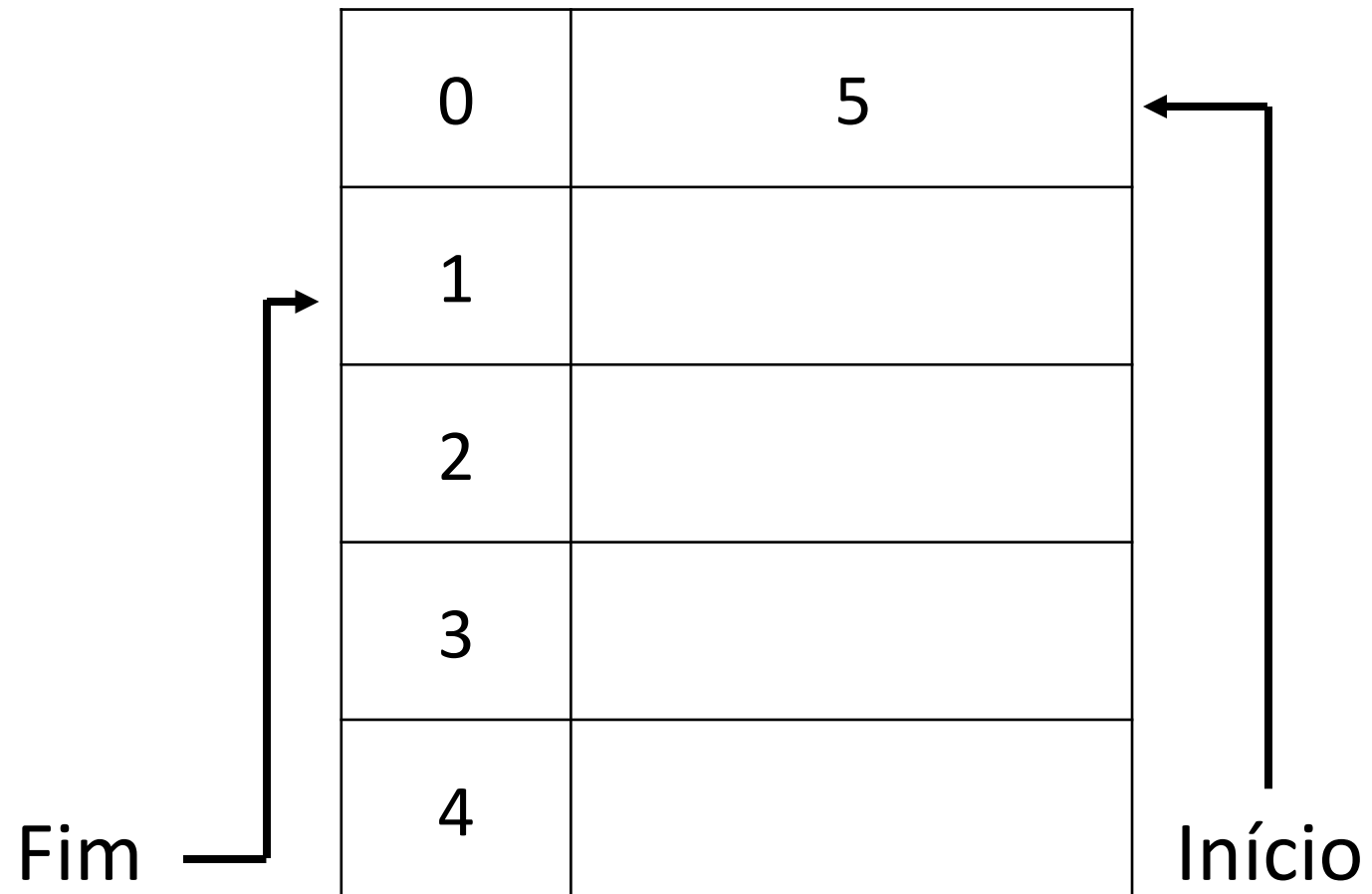
Fila Simples

- Novamente precisamos de um vetor de inteiros.
- Precisamos também de uma variável indicando o início e o fim da fila.
- Tanto o início quanto o final da fila apontam para o começo do vetor.



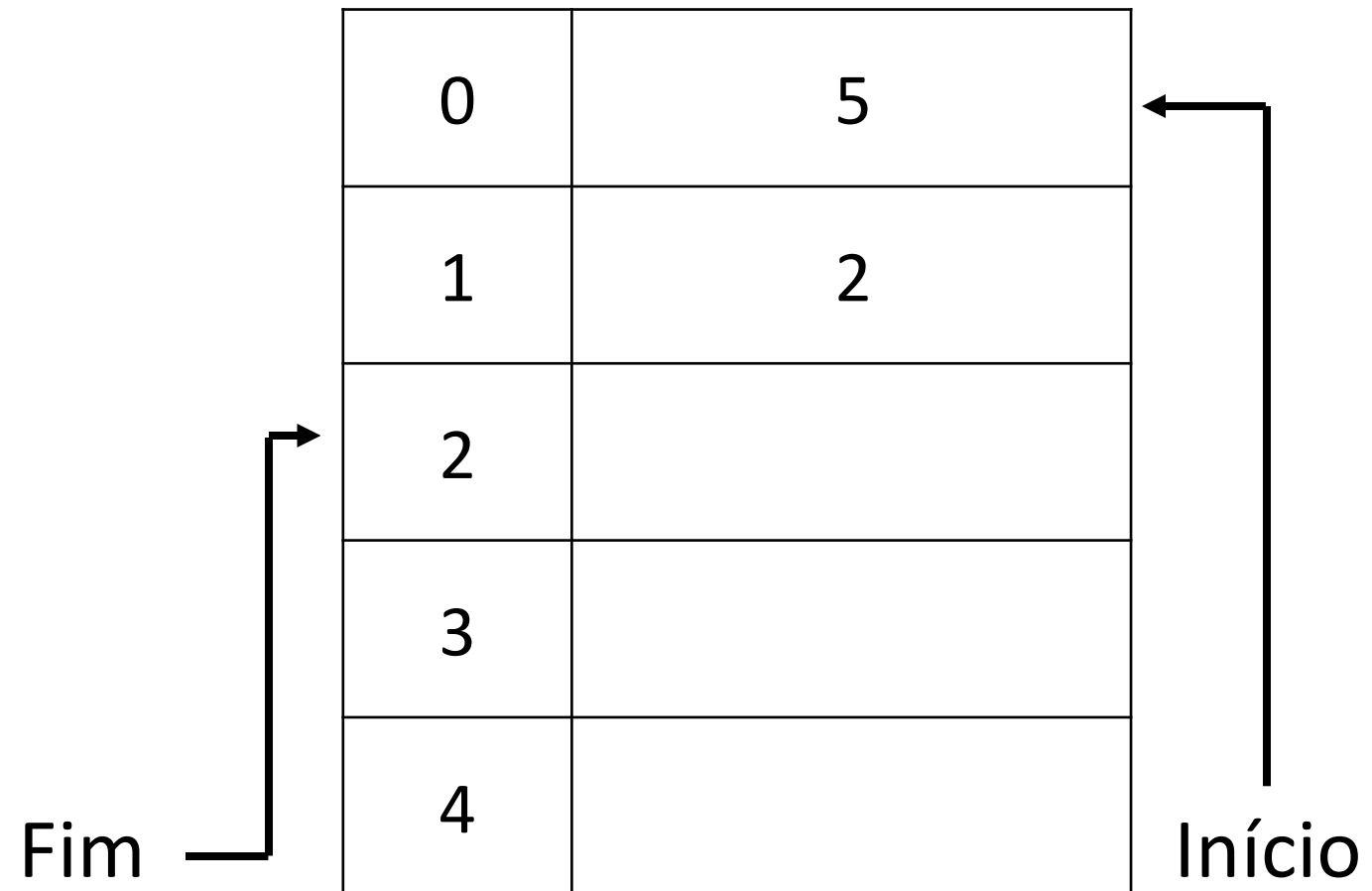
Fila Simples

- Para adicionarmos um elemento na fila devemos colocá-lo na posição apontada por fim e depois incrementarmos o fim.



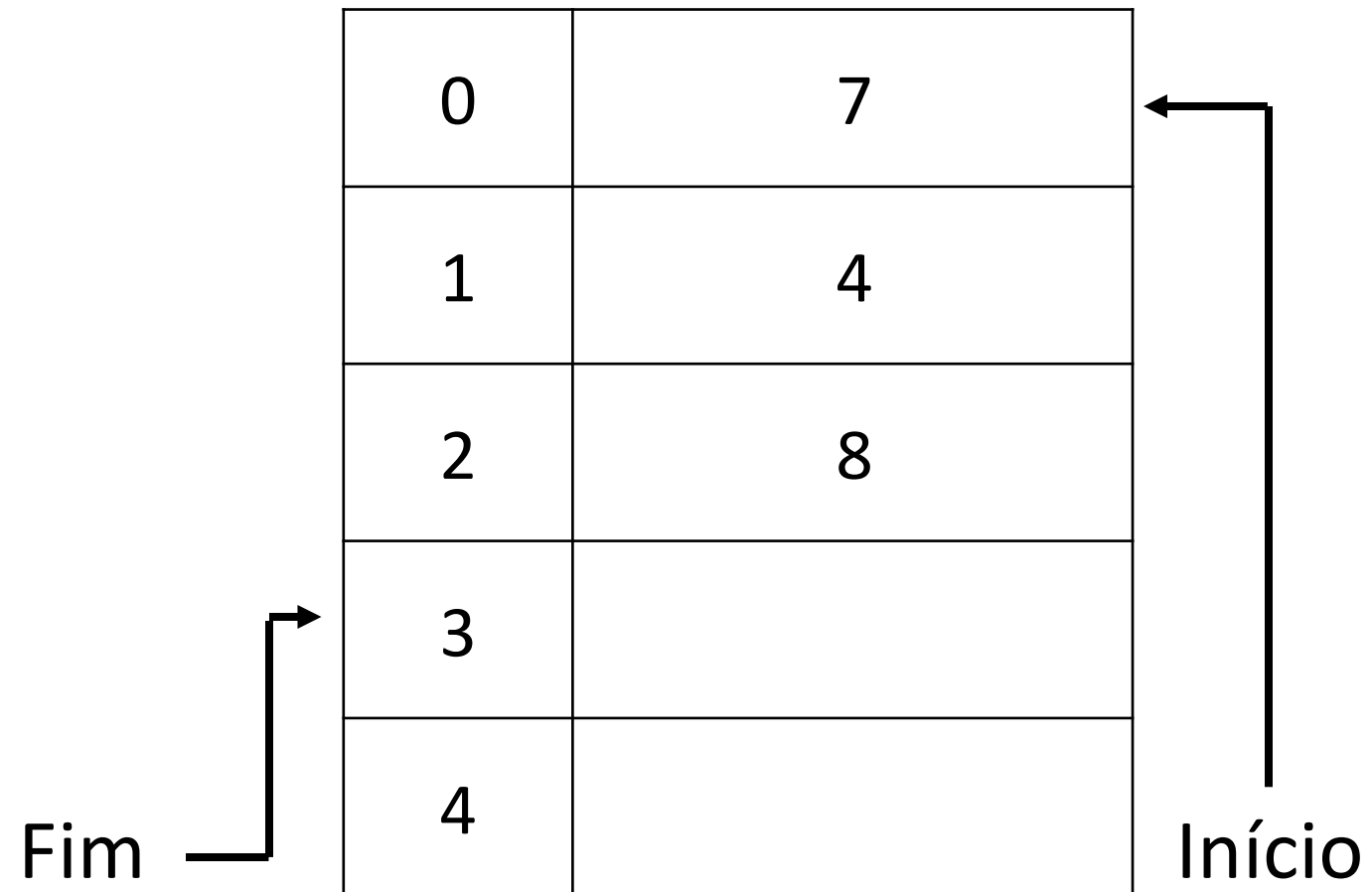
Fila Simples

- Completando a fila....



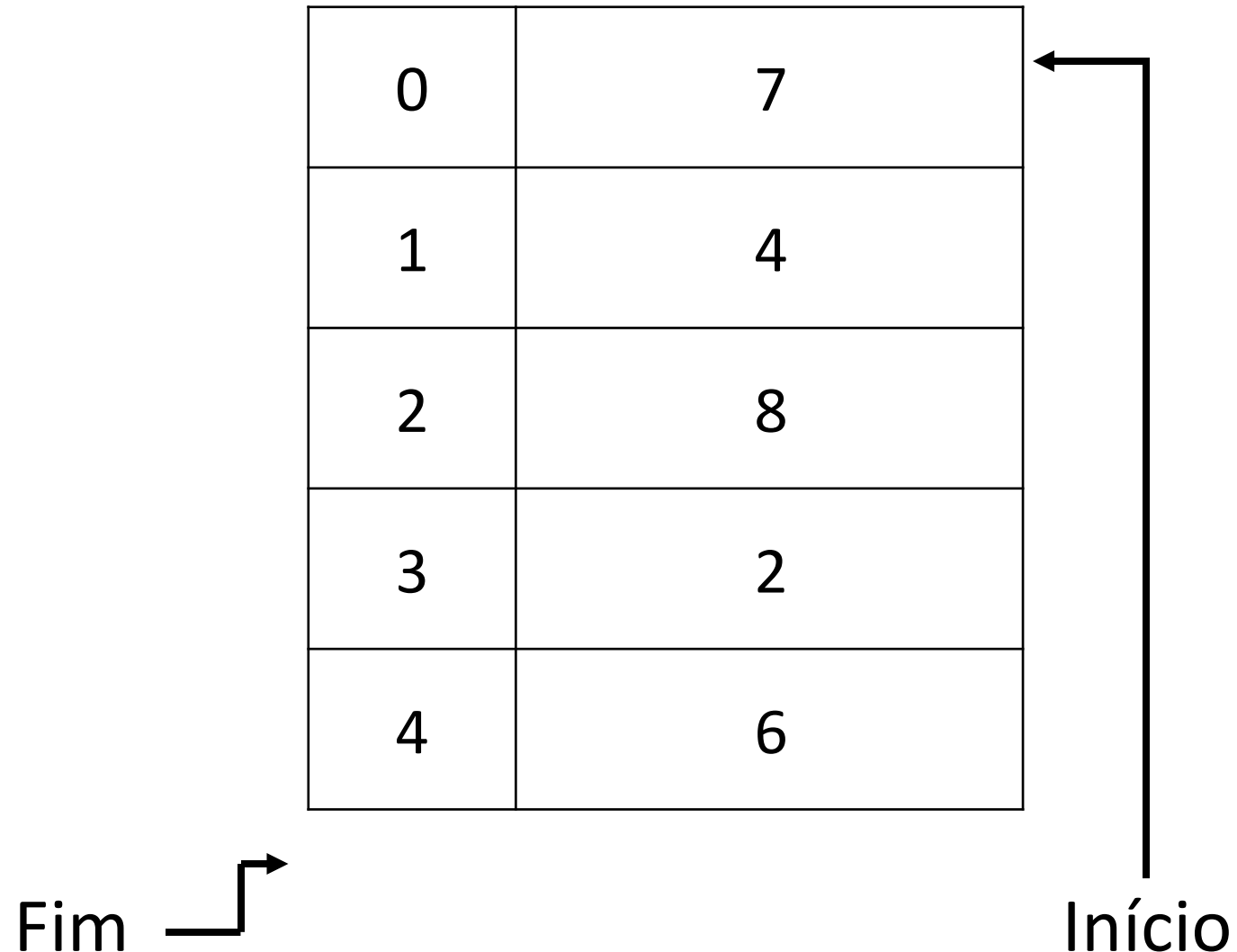
Fila Simples

- Completando a fila....



Fila Simples

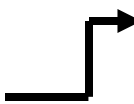
- Ao chegar no fim a fila está cheia.
- Não é possível mais entrar outros elementos.
- Na fila simples, nem mesmo se alguém sair da fila, outros elementos não podem mais entrar.



Fila Simples

- A retirada dos elementos na fila se faz copiando o elemento para uma outra variável.
- Incrementamos o início e devolvemos o conteúdo da variável.

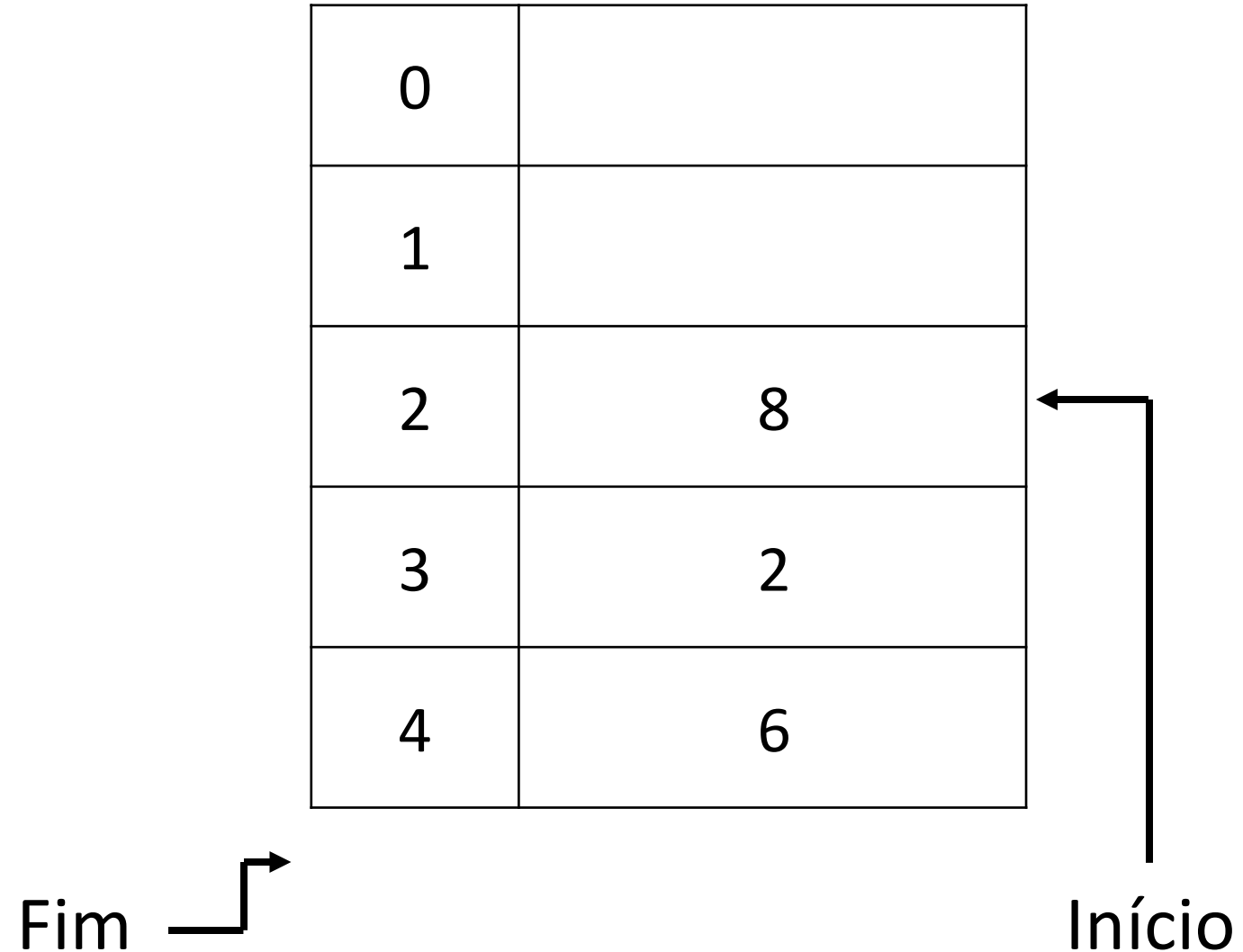
0	7
1	4
2	8
3	2
4	6

Fim 

 Início

Fila Simples

- Assim retiramos todos...



Fila Simples

- Assim retiramos todos...

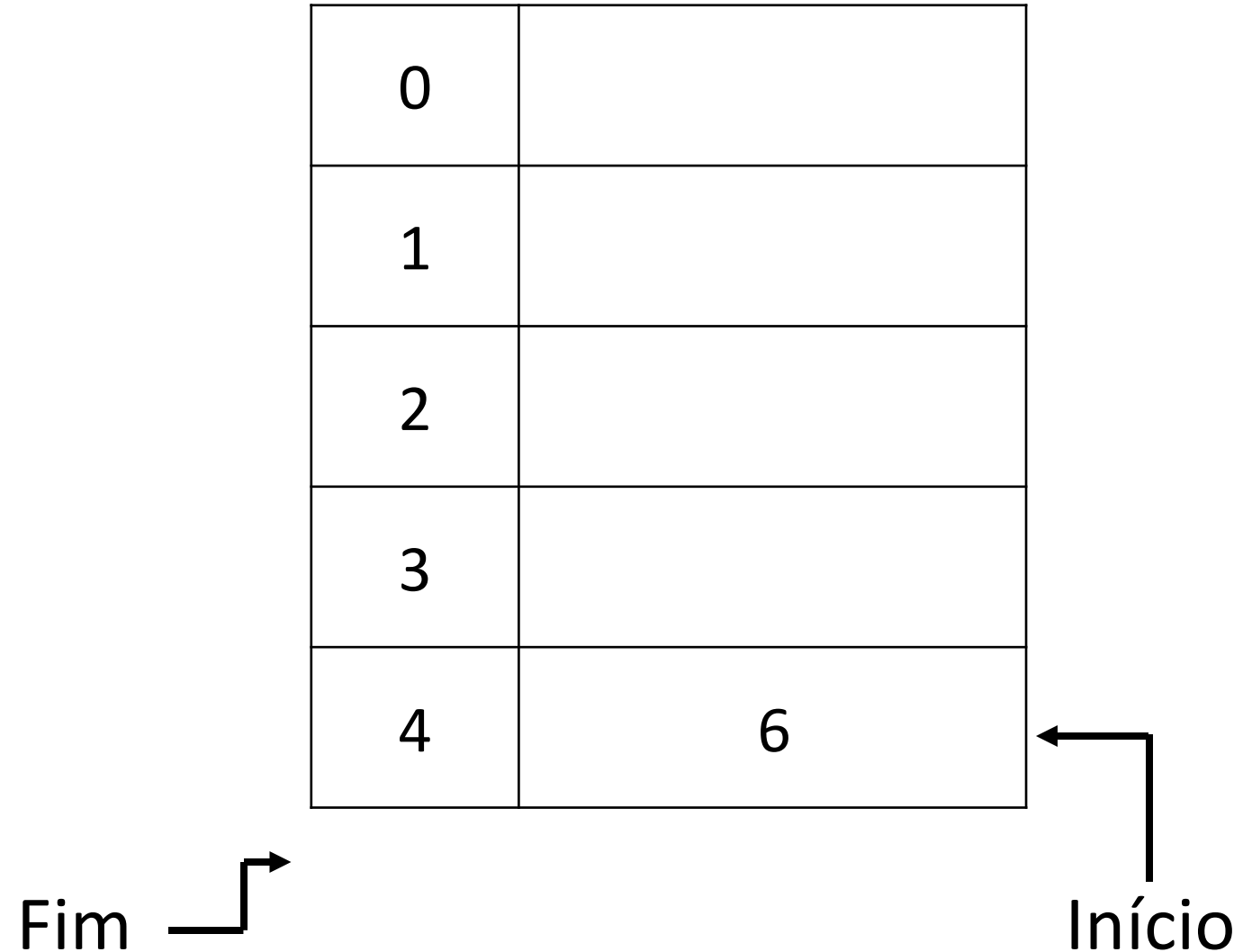
0	
1	
2	
3	2
4	6

Fim ↗

←
Início

Fila Simples

- Assim retiramos todos...



Fila Simples

- Assim retiramos todos...

0	
1	
2	
3	
4	

Fim ↗

↖ Início

Exemplo de Fila Estática

```
1  #include <iostream>
2
3  #define TAM 10
4
5  using namespace std;
6
7  void iniciaPosFila(int *inicio, int *fim)
8  {
9      *inicio = 0;
10     *fim = -1;
11 }
12
13
14 void imprimeFila(int vetor[TAM], int tamanho)
15 {
16
17     cout << endl;
18
19     for (int cont = 0; cont < TAM; cont++)
20     {
21         cout << vetor[cont] << endl;
22     }
23 }
24
25 int retornaTamanho(int fim, int inicio)
26 {
27     return (fim - inicio) + 1;
28 }
29
```

Exemplo de Fila Estática

```
30 bool filaCheia(int fim)
31 {
32     if (fim == TAM - 1)
33     {
34         return true;
35     }
36     else
37     {
38         return false;
39     }
40 }
41
42 bool filaVazia(int inicio, int fim)
43 {
44     if (inicio > fim)
45     {
46         return true;
47     }
48     else
49     {
50         return false;
51     }
52 }
53
```

Exemplo de Fila Estática

```
54 void enfileirar(int fila[TAM], int valor, int *fim)
55 {
56     if (filaCheia(*fim))
57     {
58         cout << "FILA CHEIA";
59     }
60     else
61     {
62         *fim = *fim + 1;
63         fila[*fim] = valor;
64     }
65 }
66
67 void desenfileirar(int fila[TAM], int *inicio, int fim)
68 {
69
70     if (filaVazia(*inicio, fim))
71     {
72         cout << "IMPOSSIVEL DESINFELEIRAR FILA VAZIA";
73     }
74     else
75     {
76         cout << "O valor " << fila[*inicio] <<
77         " foi removido" << endl;;
78         fila[*inicio] = 0;
79         *inicio = *inicio + 1;
80     }
81 }
82
```


Exemplo de Fila Estática

```
83  int main()  
84  {  
85      int fila[TAM] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  
86      int inicio, fim;  
87  
88      iniciaPosFila(&inicio, &fim);  
89  
90      enfileirar(fila, 10, &fim);  
91      enfileirar(fila, 9, &fim);  
92      enfileirar(fila, 8, &fim);  
93      enfileirar(fila, 7, &fim);  
94      enfileirar(fila, 6, &fim);  
95      enfileirar(fila, 5, &fim);  
96      enfileirar(fila, 4, &fim);  
97      enfileirar(fila, 3, &fim);  
98      enfileirar(fila, 2, &fim);  
99      enfileirar(fila, 1, &fim);  
100  
101      imprimeFila(fila, retornaTamanho(fim, inicio));  
102  
103      desenfileirar(fila, &inicio, fim);  
104      desenfileirar(fila, &inicio, fim);  
105      desenfileirar(fila, &inicio, fim);  
106  
107      imprimeFila(fila, retornaTamanho(fim, inicio));  
108  
109      return 0;  
110 }
```

A biblioteca/classe <queue>

- A biblioteca <queue> é uma parte do padrão C++ que fornece implementações de estruturas de dados de fila (queue) e fila de prioridade (priority_queue).
- Ela faz parte da Standard Template Library (STL), que é uma coleção de classes e funções genéricas que fornecem diversas estruturas de dados e algoritmos eficientes.

Operações básicas

- **push**

- A operação de enfileirar consiste em adicionar um elemento ao final da fila.
- Ao enfileirar um elemento, ele é inserido na parte posterior da fila (fim), mantendo a ordem dos elementos já existentes na fila.

```
1  #include <iostream>
2  #include <queue>
3
4  using namespace std;
5
6  int main() {
7      queue<int> fila;
8
9      fila.push(2);
10     fila.push(20);
11     fila.push(200);
12
13     cout << "Elementos na fila: ";
14     while (!fila.empty()) {
15         cout << fila.front() << " "; // exibe o elemento no inicio da fila
16         fila.pop(); // remove o elemento no inicio da fila
17     }
18     cout << endl;
19
20 }
```

Operações básicas

- **pop**

- A operação de desenfileirar remove o elemento na frente da fila.
- O elemento na frente (início) da fila é retirado, seguindo o princípio FIFO (First In, First Out), que garante que o primeiro elemento a entrar na fila seja o primeiro a sair.

```
#include <iostream>
#include <queue>

using namespace std;

int main() {
    queue<int> fila;

    fila.push(10);
    fila.push(20);

    cout << "Remove o elemento na frente da fila: " << fila.front() << endl;
    fila.pop();

    cout << "Elemento na frente da fila: " << fila.front() << endl;

}
```

Operações básicas

- **front**

- A operação de verificar o início da fila permite acessar o elemento que está no início (frente) da fila sem removê-lo.
- O método retorna uma referência ao elemento na frente da fila, que é o próximo a ser removido pela operação de desenfileirar.

```
#include <iostream>
#include <queue>

using namespace std;

int main() {
    queue<int> fila;

    fila.push(1000);
    fila.push(2000);
    fila.push(3000);

    cout << "O elemento na frente da fila é: " << fila.front() << endl;

    return 0;
}
```

Operações básicas

- **back**

- O método **back()** fornece uma referência ao elemento no final da fila.
- Retorna uma referência ao elemento que está no final da fila. Se a fila estiver vazia, o comportamento é indefinido (não é seguro chamar esse método em uma fila vazia).

```
#include <iostream>
#include <queue>

using namespace std;

int main() {

    queue<int> fila;

    fila.push(1);
    fila.push(10);
    fila.push(100);

    cout << "O elemento no final da fila é: " << fila.back() << endl;

    return 0;
}
```

Operações básicas

- **empty**

- A operação de verificar se a fila está vazia verifica se a fila contém algum elemento ou não.
- Se não houver elementos na fila, ela está vazia. Caso contrário, há pelo menos um elemento presente.

```
#include <iostream>
#include <queue>

using namespace std;

int main() {
    queue<int> fila;

    cout << "A fila está vazia? " << (fila.empty() ? "Sim" : "Não") << endl;

    fila.push(10);

    cout << "A fila está vazia? " << (fila.empty() ? "Sim" : "Não") << endl;

    return 0;
}
```

Operações básicas

- **size**

- A operação de obter o tamanho da fila retorna a quantidade de elementos atualmente presentes na fila.
- O método retorna um valor inteiro que indica quantos elementos existem na fila.

```
#include <iostream>
#include <queue>

using namespace std;

int main() {
    queue<int> fila;

    fila.push(1000);
    fila.push(2000);
    fila.push(3000);

    cout << "Número de elementos na fila: " << fila.size() << endl;

    return 0;
}
```


Exemplo de Fila Dinâmica

*filaDinamica.cpp X

```
4     using namespace std;
5
6     int main(int argc, char** argv)
7     {
8         /*
9         empty = vazio
10        size = tamanho
11        front =
12        back =
13        push =
14        pop =
15        */
16
17        queue <string> pessoas;
18
19        pessoas.push("RODRIGO"); //push adiciona elementos na fila
20        pessoas.push("ANDRE");
21        pessoas.push("MARIA");
22        pessoas.push("ANA");
23        pessoas.push("JOSE");
24        pessoas.push("RITA");
25
```

Exemplo de Fila Dinâmica

```
26      // size mostra o tamanho da fila
27      cout << "Tamanho da Fila: " << pessoas.size() << endl;
28      cout << "Primeira pessoa " << pessoas.front() << endl;
29      cout << "Ultima pessoa " << pessoas.back() << endl << endl;
30
31
32      while(!pessoas.empty()){
33          cout << "Primeira pessoa " << pessoas.front() << endl;
34          pessoas.pop(); // retira elementos da pilha
35      }
36
37
38      return 0;
39  }
```

Exercício

1. Escreva um programa em C++ que simule um sistema de atendimento em uma loja. O programa deve utilizar uma fila para gerenciar a ordem de atendimento dos clientes.

- O programa deve oferecer as seguintes funcionalidades:
- Inserir cliente na fila:
 - O programa deve solicitar ao usuário o nome do cliente que deseja ser atendido e inserir o cliente na fila de atendimento.
- Atender próximo cliente:
 - O programa deve remover o próximo cliente da fila e exibir uma mensagem com o nome do cliente que está sendo atendido.
- Exibir fila de espera:
 - O programa deve exibir a lista de clientes que estão aguardando na fila, na ordem em que foram inseridos.
- Encerrar o programa:
 - O programa deve permitir ao usuário encerrar o programa.

Exercício

```
Bem-vindo ao sistema de atendimento da loja!

Opções disponíveis:
1. Inserir cliente na fila
2. Atender próximo cliente
3. Exibir fila de espera
4. Encerrar o programa

Digite a opção desejada: 1
Digite o nome do cliente: Rodrigo Martins
Cliente Rodrigo Martins foi adicionado à fila de espera

Opções disponíveis:
1. Inserir cliente na fila
2. Atender próximo cliente
3. Exibir fila de espera
4. Encerrar o programa

Digite a opção desejada: 3
Fila de espera: Rodrigo Martins

Opções disponíveis:
1. Inserir cliente na fila
2. Atender próximo cliente
3. Exibir fila de espera
4. Encerrar o programa

Digite a opção desejada:
```

Referência desta aula

- Notas de Aula do Prof. Prof. Armando Luiz N. Delgado baseado em revisão sobre material de Prof.a Carmem Hara e Prof. Wagner Zola
- <http://www.cplusplus.com/reference/>

Obrigado

Rodrigo