

# **Estrutura de Dados**

**Prof. Orlando Saraiva Júnior**  
**[orlando.saraiva@unesp.br](mailto:orlando.saraiva@unesp.br)**

"Blame the implementation,  
not the technique."

Tim Kadlec

# **Estrutura de Dados**

# Objetivo da aula

---

Conhecer a estrutura de dados Fila

Fila em C++

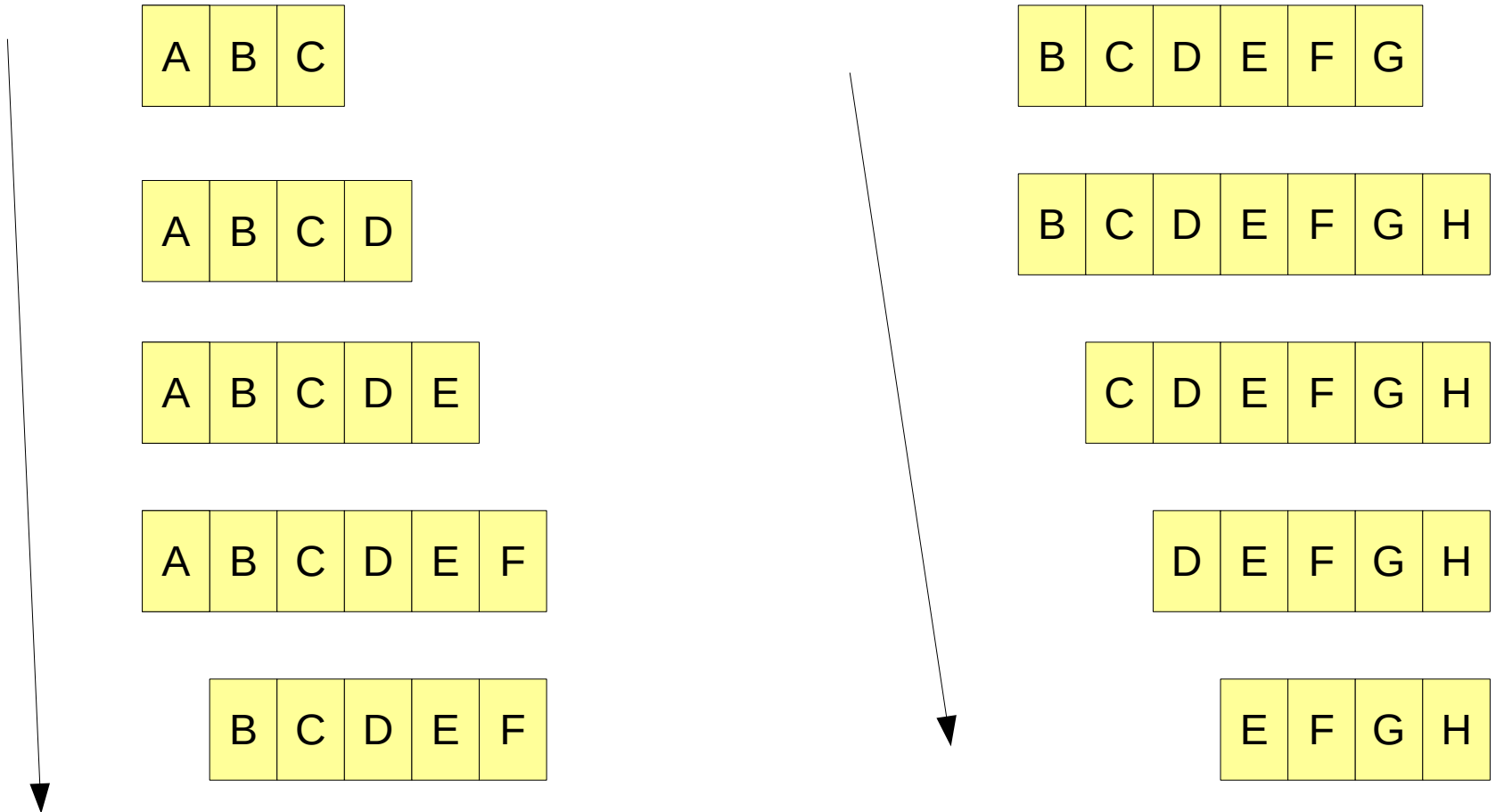
Fila em Python

Uma **fila** é um conjunto ordenado de itens a partir do qual podem-se eliminar itens numa extremidade (chamado de **início** da fila) e no qual podem-se inserir itens na outra extremidade (chamada **final** da fila)

Possuem duas funções básicas: ENQUEUE, que adiciona um elemento ao final da fila, e DEQUEUE, que remove o elemento no início da fila.

O primeiro item a ser inserido na pilha é o primeiro a ser removido. Esta política é conhecida pela sigla FIFO ( **F**irst **I**n **F**irst **O**ut ), ao contrário da pilha.

# Fila



## Exemplo 1

O escalonamento dos processos em um sistema operacional é feito por filas. O escalonador do sistema operacional utiliza uma fila de processos, dando um tempo  $t$  de CPU para cada um.

## Exemplo 2

O controle de estoque serve para a empresa avaliar a entrada e saída de mercadorias e auxilia uma companhia a reduzir custos e administrar a cadeia de produção e distribuição com mais eficiência.

Formalmente, uma fila é um tipo de dado abstrato (ADT) tal que uma instância  $Q$  ( de *queue*) suporta as seguintes funcionalidades:

$Q.enqueue(elem)$  → Adiciona o elemento  $elem$  ao fim da fila  $Q$

$Q.dequeue()$  → Remove e retorna o primeiro elemento da fila  $Q$ .  
Espera-se um erro caso a fila esteja vazia.



# Entendendo Filas

---

No exemplo, teremos outros três funcionalidades

- `Q.first()` → Retorna a referência ao elemento no início da fila.
- `Q.is_empty()` → Retorna verdadeiro, caso a fila esteja vazia.
- `Q.tamanho()` → Retorna o número de elementos da fila `Q`.

# Entendendo Filas

Operação	Valor de retorno	Conteúdo da fila
Q.enqueue(5)	-	[ 5 ]
Q.enqueue(3)	-	[ 5, 3 ]
Q.tamanho()	2	[ 5, 3 ]
Q.dequeue()	5	[ 3 ]
Q.is_empty()	False	[ 3 ]
Q.pop()	5	[ ]
Q.is_empty()	True	[ ]
Q.dequeue()	"erro"	[ ]
Q.enqueue(7)	-	[ 7 ]
Q.enqueue(9)	-	[ 7, 9 ]
Q.enqueue(4)	-	[ 7, 9 , 4 ]
Q.tamanho()	3	[ 7, 9 , 4 ]
Q.enqueue(8)	-	[ 7, 9 , 4 , 8 ]

# Entendendo Filas

## (continuação)

Operação	Valor de retorno	Conteúdo da fila
Q.enqueue(6)	-	[ 7 , 9 , 4 , 8 , 6 ]
Q.enqueue(1)	? ? ?	? ? ?
Q.tamanho()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.is_empty()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.enqueue(1)	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?

# Entendendo Filas

## (continuação)

Operação	Valor de retorno	Conteúdo da fila
Q.enqueue(6)	-	[ 7 , 9 , 4 , 8 , 6 ]
Q.enqueue(1)	-	[ 7 , 9 , 4 , 8 , 6 , 1]
Q.tamanho()	6	[ 7 , 9 , 4 , 8 , 6 , 1]
Q.dequeue()	7	[ 9 , 4 , 8 , 6 , 1]
Q.is_empty()	False	[ 9 , 4 , 8 , 6 , 1]
Q.dequeue()	9	[ 4 , 8 , 6 , 1]
Q.dequeue()	4	[ 8 , 6 , 1]
Q.dequeue()	8	[ 6 , 1]
Q.dequeue()	6	[ 1 ]
Q.enqueue(1)	-	[ 1, 1 ]
Q.dequeue()	1	[ 1 ]
Q.dequeue()	1	[ ]
Q.dequeue()	"erro"	[ ]

# **Implementando pilha com C++ Primeira versão**

# Fila em C++: Primeira versão

---

Nesta primeira versão, declara-se um ponteiro da estrutura **Item** (linha 16) e inteiros **N**.

A função **inicializar** aloca **N** elementos de tamanho **Item**. A função **inserir** insere um elemento no fim da fila, e a função **remover** imprime o último elemento na fila. O controle de início e fim da fila ocorre com uso das variáveis estáticas **inicio** e **fim**.

# **Implementando fila com C++ Segunda versão**

# Pilha em C++: Segunda versão

---

Nesta segunda versão, **não há um limite** para o número de elementos que podem ser empilhados.

Uma estrutura (NO) serve para armazenar o valor do elemento e um ponteiro de NO para o próximo elemento da fila.

Tanto o ELEMENTO e a FILA são definidos com typedef.



# Implementando fila com Python

Assim como pilhas, podemos implementar uma fila facilmente armazenando seus elementos em uma lista interna. Python possui tipos *built-in* para tipos sequencia.

O tipo *list* já suporta a adição de um elemento ao final com o método *insert*, e removendo o último elemento com o método *remove*.

Observe a implementação proposta, comparando-o com a linguagem C++.

# Dúvidas

**Prof. Orlando Saraiva Júnior**  
**[orlando.saraiva@unesp.br](mailto:orlando.saraiva@unesp.br)**

# Fechamento

## Melhorar a segunda implementação

---

### Fila de itens

Implementar as funções **buscaFila**, que deve receber o elemento procurado e o final da fila.

Exemplo de uso:

[2] -> [4] -> [5] -> [3] -> [1] -> [8] ->

O valor 8 encontra-se na posicao 1 da fila

### Destrutor de alocações

Ao encerrar o programa fila2, ocorre vazamento de memória ( leaks de memória). Trabalhe na função **destruir** de forma que, ao passar o final da fila,