

# MC202 - Estruturas de Dados

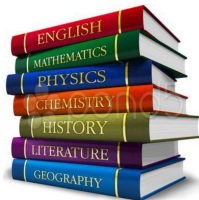
Alexandre Xavier Falcão

Instituto de Computação - UNICAMP

[afalcao@ic.unicamp.br](mailto:afalcao@ic.unicamp.br)

# Pilhas e Filas

- Uma **pilha** é uma estrutura de dados linear, na qual operações de inserção e remoção de um elemento ocorrem sempre no mesmo extremo (**topo da pilha**).
- O último elemento empilhado é sempre o primeiro a sair da pilha.
- Uma pilha pode ser implementada usando lista ligada ou vetor.



# Pilhas e Filas

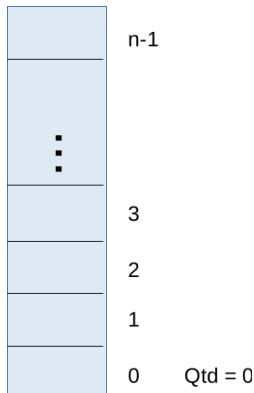
- Uma **fila** é uma estrutura de dados linear, na qual operações de inserção ocorrem em um extremo (**final da fila**) e remoção ocorrem no outro extremo (**início da fila**).
- O primeiro elemento na fila é sempre o primeiro a sair dela.
- Uma fila também pode ser implementada usando lista ligada ou vetor.



- Pilhas estática e dinâmica.
- Aplicações de Pilha.
- Filas estática e dinâmica.
- Aplicações de Fila.

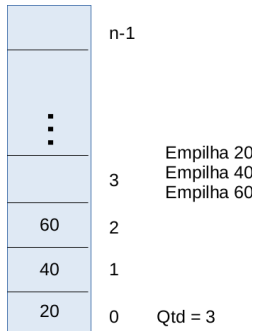
# Pilha Estática

Uma pilha estática é implementada como um vetor de tamanho  $n$ . Mesmo vazia, ela já ocupa  $n$  posições de memória.



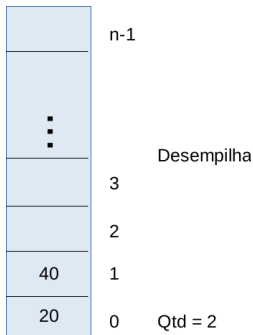
# Pilha Estática

As inserções são feitas sempre na posição igual à quantidade de elementos na pilha.



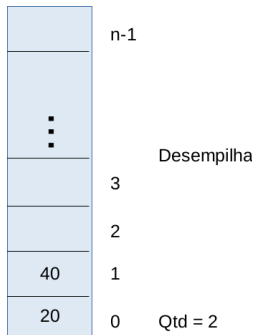
# Pilha Estática

As remoções são feitas sempre na posição igual à quantidade de elementos na pilha **menos um** (índice do topo).



# Pilha Estática

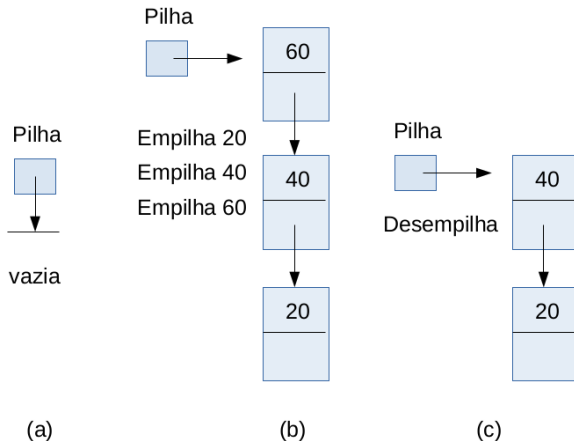
As remoções são feitas sempre na posição igual à quantidade de elementos na pilha **menos um** (índice do topo).



Vamos entender a implementação de uma pilha estática.



# Pilha Dinâmica



A implementação usa lista ligada: (a) A pilha está vazia, (b) as inserções são no início da lista, e (c) as remoções também.

Pilhas podem ser usadas para

- reverter a ordem dos caracteres de uma cadeia,
- para implementar algoritmos de backtracking,
- para controlar a ordem de chamada das funções de um programa,
- transformar algoritmos recursivos em iterativos,
- para avaliar expressões matemáticas, etc.

Vamos estudar três tarefas simples relacionadas a expressões matemáticas do tipo  $(5 + 3) * 4 - 8/2^2$  usando pilha.

- Balanceamento dos parênteses: espera-se que cada ocorrência de '(' tenha uma ocorrência de ')' correspondente.

Vamos estudar três tarefas simples relacionadas a expressões matemáticas do tipo  $(5 + 3) * 4 - 8/2^2$  usando pilha.

- Balanceamento dos parênteses: espera-se que cada ocorrência de '(' tenha uma ocorrência de ')' correspondente.
- Conversão de notação: a expressão acima está em notação **infixa**, pois os operadores estão entre operandos, e poderia ser convertida para notação **pósfixa** (operadores após operandos),  $53 + 4 * 822 / -$ , para fins de avaliação.

Vamos estudar três tarefas simples relacionadas a expressões matemáticas do tipo  $(5 + 3) * 4 - 8/2^2$  usando pilha.

- Balanceamento dos parênteses: espera-se que cada ocorrência de '(' tenha uma ocorrência de ')' correspondente.
- Conversão de notação: a expressão acima está em notação **infixa**, pois os operadores estão entre operandos, e poderia ser convertida para notação **pósfixa** (operadores após operandos),  $53 + 4 * 822 / -$ , para fins de avaliação.
- Avaliação de uma expressão em notação pósfixa.

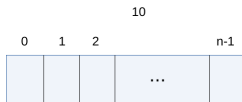
Vamos estudar três tarefas simples relacionadas a expressões matemáticas do tipo  $(5 + 3) * 4 - 8/2^2$  usando pilha.

- Balanceamento dos parênteses: espera-se que cada ocorrência de '(' tenha uma ocorrência de ')' correspondente.
- Conversão de notação: a expressão acima está em notação **infixa**, pois os operadores estão entre operandos, e poderia ser convertida para notação **pósfixa** (operadores após operandos),  $53 + 4 * 822 / -$ , para fins de avaliação.
- Avaliação de uma expressão em notação pósfixa.

Vamos implementar esses exemplos com pilha dinâmica.

## Fila Estática

Uma fila estática pode ser implementada como um vetor de tamanho  $n$ , com os índices de início e fim.



Início = fim

(a) Vazia (qtde = 0)



## Inicio

film

(b) Após inserção de 10, 20, e 30 (qtde = 3)



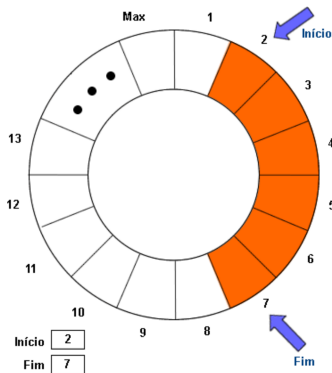
## Inicio

fim

(c) Após remoção do 10 (qtde = 2)

# Fila Estática Circular

No entanto, sua implementação **circular** oferece maior vantagem já que os elementos na fila ocupam posições do índice de início ao índice de fim **menos um**, e ambos se deslocam no vetor.



O índice de **fim** sempre está na posição onde será realizada uma nova inserção.



- Após inserir novo elemento na **fim**, este índice deve ser incrementado por  $fim \leftarrow (fim + 1) \% n$ .

# Fila Estática e Circular

- Após inserir novo elemento na **fim**, este índice deve ser incrementado por  $fim \leftarrow (fim + 1) \% n$ .
- Os elementos são removidos da posição **início**, a qual é subsequentemente incrementada por  $inicio \leftarrow (inicio + 1) \% n$ .

- Após inserir novo elemento na **fim**, este índice deve ser incrementado por  $fim \leftarrow (fim + 1) \% n$ .
- Os elementos são removidos da posição **início**, a qual é subsequentemente incrementada por  $inicio \leftarrow (inicio + 1) \% n$ .
- A fila está **cheia** apenas quando a quantidade de elementos  $qtde = n$ .

- Após inserir novo elemento na **fim**, este índice deve ser incrementado por  $fim \leftarrow (fim + 1) \% n$ .
- Os elementos são removidos da posição **início**, a qual é subsequentemente incrementada por  $inicio \leftarrow (inicio + 1) \% n$ .
- A fila está **cheia** apenas quando a quantidade de elementos  $qtde = n$ .
- Note que  $inicio = fim$  pode ocorrer em duas situações, fila vazia e fila cheia, portanto é importante armazenar a quantidade de elementos na fila.

# Fila Estática e Circular

- Após inserir novo elemento na **fim**, este índice deve ser incrementado por  $fim \leftarrow (fim + 1) \% n$ .
- Os elementos são removidos da posição **início**, a qual é subsequentemente incrementada por  $inicio \leftarrow (inicio + 1) \% n$ .
- A fila está **cheia** apenas quando a quantidade de elementos  $qtde = n$ .
- Note que  $inicio = fim$  pode ocorrer em duas situações, fila vazia e fila cheia, portanto é importante armazenar a quantidade de elementos na fila.

Vamos entender a implementação de uma fila estática e circular.

Com o que vimos em Listas, filas dinâmicas podem ser facilmente implementadas usando

Com o que vimos em Listas, filas dinâmicas podem ser facilmente implementadas usando

- uma lista ligada simples,

Com o que vimos em Listas, filas dinâmicas podem ser facilmente implementadas usando

- uma lista ligada simples,
- uma lista ligada dupla, ou



Com o que vimos em Listas, filas dinâmicas podem ser facilmente implementadas usando

- uma lista ligada simples,
- uma lista ligada dupla, ou
- uma lista ligada dupla e circular.

Com o que vimos em Listas, filas dinâmicas podem ser facilmente implementadas usando

- uma lista ligada simples,
- uma lista ligada dupla, ou
- uma lista ligada dupla e circular.

As operações de inserção após o último nó e de remoção do nó inicial são mais fáceis no último caso.

Com o que vimos em Listas, filas dinâmicas podem ser facilmente implementadas usando

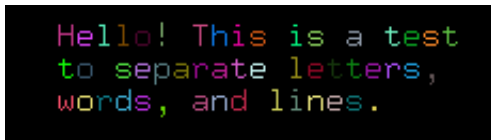
- uma lista ligada simples,
- uma lista ligada dupla, ou
- uma lista ligada dupla e circular.

As operações de inserção após o último nó e de remoção do nó inicial são mais fáceis no último caso. **Esta implementação fica como exercício.**

Filas podem ser usadas para

- organizar a ordem de execução de programas de mesma prioridade,
- simular filas do mundo real, tais como filas para a compra de ingressos,
- determinar o número de caixas de supermercado com base no tempo médio de espera dos clientes em fila,
- transferir dados de forma assíncrona,
- para rotular componentes conexos de um grafo.

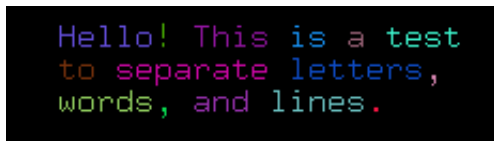
Vamos estudar o caso de rotulação de letras, palavras, e linhas de uma imagem binária com um texto.



```
Hello! This is a test  
to separate letters,  
words, and lines.
```

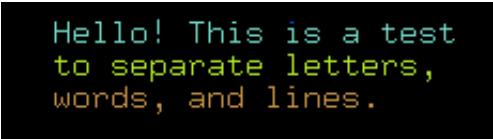
Cada rótulo (número inteiro de 1 em diante) é convertido em uma cor aleatória.

Vamos estudar o caso de rotulação de letras, palavras, e linhas de uma imagem binária com um texto.



Cada rótulo (número inteiro de 1 em diante) é convertido em uma cor aleatória.

Vamos estudar o caso de rotulação de letras, palavras, e linhas de uma imagem binária com um texto.



```
Hello! This is a test  
to separate letters,  
words, and lines.
```

Cada rótulo (número inteiro de 1 em diante) é convertido em uma cor aleatória.

# Rotulação de componentes conexos

- A definição de componente conexo em uma imagem depende de uma **relação de adjacência**.



# Rotulação de componentes conexos

- A definição de componente conexo em uma imagem depende de uma **relação de adjacência**.
- Um pixel  $q = (x_q, y_q)$  é adjacente a um pixel  $p = (x_p, y_p)$  se satisfizer uma relação baseada na posição relativa entre eles.

# Rotulação de componentes conexos

- A definição de componente conexo em uma imagem depende de uma **relação de adjacência**.
- Um pixel  $q = (x_q, y_q)$  é adjacente a um pixel  $p = (x_p, y_p)$  se satisfizer uma relação baseada na posição relativa entre eles.
- Podemos definir uma **relação retangular** em que  $p$  é o pixel central e  $q$  é adjacente a  $p$  se ambos tiverem o mesmo brilho,  $|x_q - x_p| \leq \frac{w}{2}$ , e  $|y_q - y_p| \leq \frac{h}{2}$ .

# Rotulação de componentes conexos

- A definição de componente conexo em uma imagem depende de uma **relação de adjacência**.
- Um pixel  $q = (x_q, y_q)$  é adjacente a um pixel  $p = (x_p, y_p)$  se satisfizer uma relação baseada na posição relativa entre eles.
- Podemos definir uma **relação retangular** em que  $p$  é o pixel central e  $q$  é adjacente a  $p$  se ambos tiverem o mesmo brilho,  $|x_q - x_p| \leq \frac{w}{2}$ , e  $|y_q - y_p| \leq \frac{h}{2}$ .
- Um pixel  $q$  é **conexo** a um pixel  $p$  se existir uma sequência  $\langle p_1 = p, p_2, p_3, \dots, p_k = q \rangle$  de pixels adjacentes que conecte  $p$  a  $q$ .

# Rotulação de componentes conexos

Um componente conexo em uma imagem binária é um conjunto maximal de pixels de mesmo brilho em que todos os pares de pixels são conexos.

1	1		1	1	
		1			
1				1	
1		1	1	1	
1					

Dependendo da escolha de  $w$  e  $h$ , poderemos identificar 3 componentes, 2 ou um único componente conexo de brilho 1.

# Rotulação de componentes conexos

Um componente conexo em uma imagem binária é um conjunto maximal de pixels de mesmo brilho em que todos os pares de pixels são conexos.

1	1		1	1	
		1			
2				3	
2		3	3	3	
2					

$w = 3$  e  $h = 3$

Dependendo da escolha de  $w$  e  $h$ , poderemos identificar 3 componentes, 2 ou um único componente conexo de brilho 1.

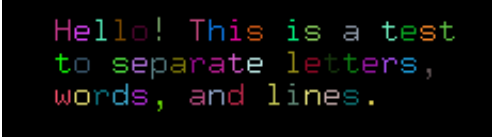
# Rotulação de componentes conexos

Um componente conexo em uma imagem binária é um conjunto maximal de pixels de mesmo brilho em que todos os pares de pixels são conexos.

1	1		1	1	
		1			
2				2	
2		2	2	2	
2					

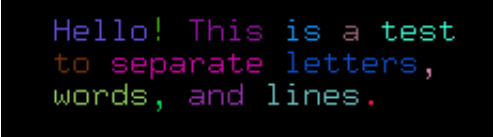
$w = 5$  e  $h = 3$

Dependendo da escolha de  $w$  e  $h$ , poderemos identificar 3 componentes, 2 ou um único componente conexo de brilho 1.



```
Hello! This is a test  
to separate letters,  
words, and lines.
```

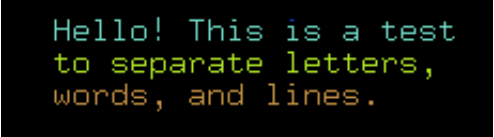
Aqui escolhemos  $w = 5$  e  $h = 11$  para letras,  $w = 11$  e  $h = 11$  para palavras, e  $w = 31$  e  $h = 5$  para as linhas. Vamos entender a implementação.



```
Hello! This is a test
to separate letters,
words, and lines.
```

Aqui escolhemos  $w = 5$  e  $h = 11$  para letras,  $w = 11$  e  $h = 11$  para palavras, e  $w = 31$  e  $h = 5$  para as linhas. Vamos entender a implementação.





```
Hello! This is a test
to separate letters,
words, and lines.
```

Aqui escolhemos  $w = 5$  e  $h = 11$  para letras,  $w = 11$  e  $h = 11$  para palavras, e  $w = 31$  e  $h = 5$  para as linhas. Vamos entender a implementação.