

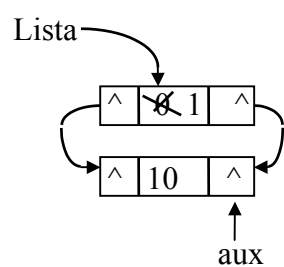
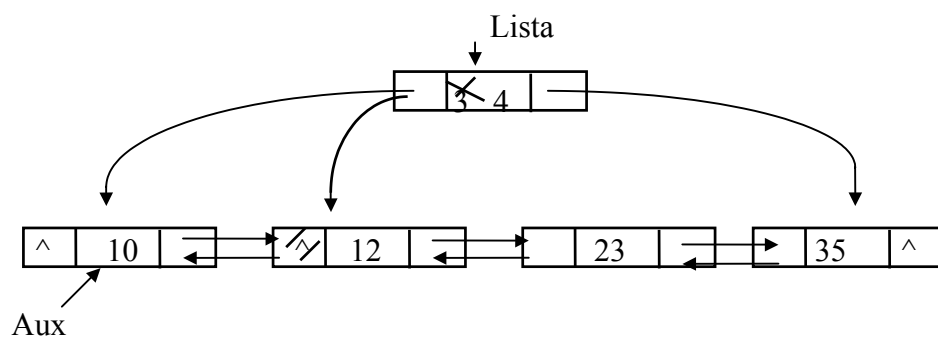
**Instituto Luterano de Ensino Superior de Ji-Paraná**  
**Curso Bacharelado em Informática**  
**Estrutura de Dados I**  
**Prof.: José Luiz A. Duizith**

**Procedimento Insere\_Esquerda (Lista,Valor)**

```

Aloque(Aux)
Se (Aux = Nil)
Entao ERRO
Senao
  Aux↑.Dado ← Valor
  Aux↑.Ant ← Nil
  Se (Lista↑.Qtd = 0)
  Entao
    Aux↑.Prox ← Nil
    Lista↑.Fim ← Aux
  Senao
    Aux↑.Prox ← Lista↑.Inicio
    Lista↑.Inicio↑.Ant ← Aux
  Lista↑.Inicio ← Aux
  Lista↑.Qtd ← Lista↑.Qtd + 1

```



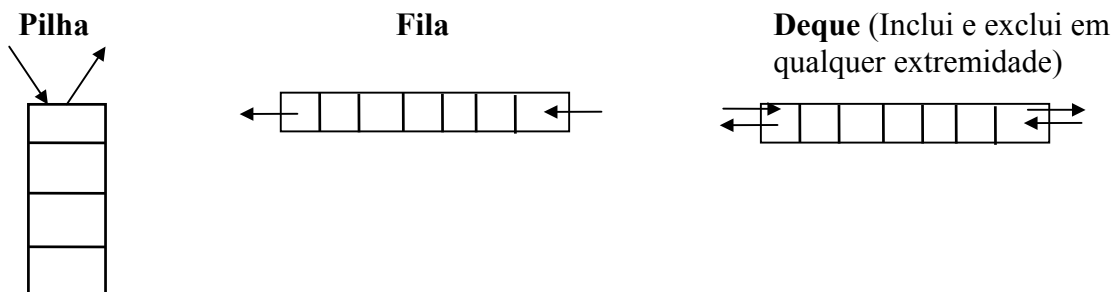
## Listas com Disciplinas de Acesso

■ Critérios usuais para inclusão e remoção de nodos em listas:

- **LIFO** (“Last In First Out” → Último a entrar e 1º a sair) : dentre os elementos que ainda permanecem no conjunto, o primeiro elemento a ser retirado é o último que foi inserido.
- **FIFO** (“First In First Out” → 1º a entrar e 1º a sair) : dentre os elementos que ainda permanecem no conjunto, o primeiro elemento a ser retirado é o primeiro que foi inserido.

Estruturas Lineares c/ disciplina de acesso:

- Pilha (critério LIFO)
- Fila (critério FIFO)
- Deque (híbrido)



### **Pilha (Stack)**

Lista linear na qual todas as inserções e remoções são feitas em apenas uma extremidade (início ou fim) da lista, chamado TOPO da pilha.

#### **(A) Operações:**

Criação

Destruição

Empilhar (Push) → inserção de novo elemento no topo da pilha.

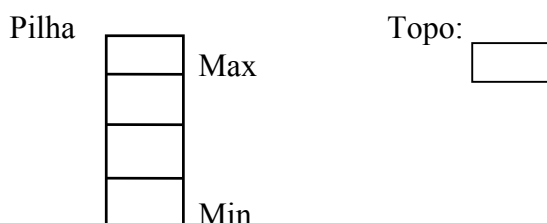
Desempilhar (Pop) → remoção de elementos do topo da pilha.

Consultar (Top) → Consulta o elemento do topo da pilha.

Verificação se pilha está vazia (EMPTY)

#### **(B) Representação:**

- Por Contigüidade



Em Pascal:

```

Const
    Min = ...
    Max = ...
Var
    Pilha = Array [ Min ... Max] of Informacao
    Topo : Integer

```

**Procedimento Cria\_Pilha (Pilha,Topo,Min,Max)**

```

Cria_Vetor (Pilha,Min,Max)
Topo ← Min - 1

```

**Procedimento Destroi\_Pilha (Pilha)**

```

Destroi_Vetor (Pilha)

```

**Function Verifica\_Pilha (Pilha, Topo, Min) : Lógico**

```

Se Topo < Min
Entao Logico ← Verdadeiro
Senao Logico ← Falso

```

**Procedimento Push (Pilha,Topo,Valor,Max)**

```

{Procedimento de Inserção}
Se Topo = Max
Entao ERRO
Senao Topo ← Topo + 1
    Pilha [Topo] ← Valor

```

**Procedimento Pop (Pilha, Topo, Min)**

```

Se (Verifica_Pilha (Pilha,Topo,Min))
Entao ERRO
Senao Topo ← Topo - 1

```

**Funcao Top (Pilha, Topo, Min) : Valor**

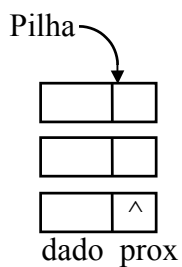
```

Se (Verifica_Pilha (Pilha,Topo,Min))
Entao ERRO
Senao Valor ← Pilha [Topo]

```

**Pilha**

- Por encadeamento

**Pascal:**

Type

```

Aponta_Nodo = ^Nodo;
  Nodo = RECORD
    Dado : Informacao
    Prox : Aponta_Nodo
  end;

```

Var

```

Pilha : Aponta_Nodo;

```

**Procedimento Cria\_Pilha (Pilha)**

```

Pilha ← Nil

```

**Procedimento Destroi\_Pilha (Pilha)**

```

Enquanto (Pilha <> Nil)
  Faca
    Aux ← Pilha
    Pilha ← Pilha↑.Prox
  Desaloque (Aux)

```

**Funcao Pilha\_Vazia (Pilha) : Logico**

```

Se (Pilha = Nil)
  Entao Logico ← True
Senao Logico ← False

```

**Procedimento Push (Pilha, Valor)**

```

Aloque(Aux)
Se (Aux = Nil)
  Entao ERRO {Overflow}
Senao
  Aux↑.Dado ← Valor
  Aux↑.Prox ← Pilha
  Pilha ← Aux

```

**Procedimento Pop (Pilha)**

```

Se (Pilha_Vazia (Pilha))
Entao ERRO
Senao
  Aux ← Pilha      {aux aponta p/ o início da pilha}
  Pilha ← Pilha↑.Prox
  Desaloque (Aux)

```

### Funcao Top (Pilha) : Valor

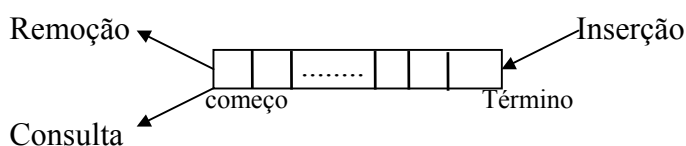
{ Função Consulta }

```

Se (Pilha_Vazia(Pilha))
Entao ERRO
Senao Valor ← Pilha↑.Dado

```

### Fila (FIFO)

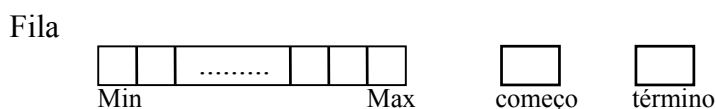


#### (A) Operação:

- Criação
- Destruição
- Inserção de novo elemento (no término)
- Remoção de um elemento (do começo)
- Consulta de elemento (do começo)
- Verificação de fila vazia

#### (B) Representação:

■ Por contigüidade:



#### Procedimento Cria\_Fila (Fila, Comeco, Termino, Min, Max)

```

Cria_Vetor (Fila, Min, Max)
Comeco ← Min
Termino ← Min - 1

```

#### Funcao Fila\_Vazia (Fila, Termino, Comeco) : Logico

```

Logico ← Termino < Comeco

```

#### Procedimento Destroi\_Fila (Fila)

```

Destroi_Vetor (Fila)

```

#### Procedimento Insere\_Fila (Fila, Termino, Valor, Max)

```

Se (Termino = Max) {fila cheia}

```

```

Entao ERRO      {overflow}
Senao | Termino ← Termino + 1
      | Fila [Termino] ← Valor

```

**Procedimento Remove\_Fila (Fila, Comeco, Termino)**

```

Se (Fila_Vazia (Fila, Termino, Comeco))
Entao ERRO
Senao Comeco ← Comeco + 1

```

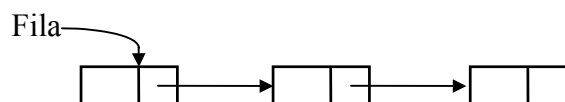
**Funcao Consulta\_Fila (Fila, Comeco, Termino) : Valor**

```

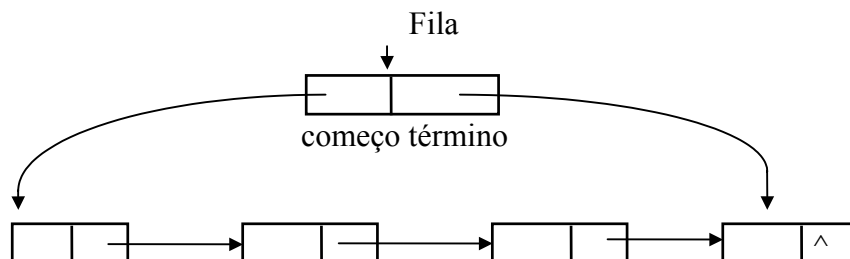
Se (Fila_Vazia(Fila, Termino, Comeco))
Entao ERRO {fila vazia}
Senao Valor ← Fila [Comeco]

```

■ Por encadeamento:



Seria melhor:



**Procedimento Cria\_Fila (Fila)**

```

Aloque (Aux)
Se (Fila = Nil)
Entao ERRO
Senao | Fila↑.Comeco ← Nil
      | Fila↑.Termino ← Nil

```

**Funcao Fila\_Vazia (Fila) : Logico**

```

Logico ← (Fila↑.Comeco = Nil) e (Fila↑.Termino = Nil)

```

**Procedimento Destroi\_Fila (Fila)**

```

Enquanto (Fila↑.comeco <> Nil)
Faça | Aux ← Fila↑.Comeco

```

```

    Fila↑.Comeco ← Aux↑.Prox
    Desaloque (Aux)
Desaloque (Fila)
Fila ← Nil

```

### **Procedimento Insere\_Fila (Valor,Fila)**

```

Aloque (Aux)
Se (Aux = Nil)
Entao ERRO {Overflow}
Senao Aux↑.Dado ← Valor
    Aux↑.Prox ← Nil
    Se (Fila_Vazia (Fila) )
    Entao Fila↑.Comeco ← Aux
    Senao Fila↑.Termino↑.Prox ← Aux
    Fila↑.Termino ← Aux

```

### **Procedimento Remove\_Fila (Fila)**

```

Se (Fila_Vazia (Fila) )
Entao ERRO {Fila Vazia}
Senao Aux ← Fila↑.Comeco
    Fila↑.Comeco ← Aux↑.Prox
    Se (Aux↑.Prox = Nil)
    Entao Fila↑.Termino ← Nil
    Desaloque (Aux)

```

### **Funcao Consulta\_Fila (Fila) : Valor**

```

Se (Fila_Vazia (Fila))
Entao ERRO {Fila vazia}
Senao Valor ← Fila↑.Comeco↑.Dado

```

## DEQUE (“Double Ended Queue”)

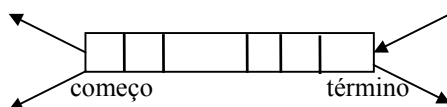
### Queue - Fila

Em um “deque” as operações de inserção e remoção podem ser realizadas em ambas as extremidades. Pode-se restringir um “deque” indicando quais as operações válidas para determinada extremidade.

#### (A) Operações:

- Criação
- Destruição
- Inserção no começo
- Inserção no término (idem a fila)
- Remoção no começo (idem a fila)
- Remoção no término
- Consulta no término
- Consulta no começo (idem a fila)
- Verifica se deque está vazio (idem a fila)

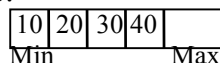
Deque:



#### (B) Representação:

- Por contigüidade

Deque:



#### **Procedimento Cria\_Deque**

*{idem a fila}*

#### **Procedimento Destruição\_Deque**

*(idem a fila)*

#### **Procedimento Insere\_Deque\_Começo (Deque, Começo, Valor, Min)**

Se (Começo = Nil)

Entao ERRO

Senao  $\left\{ \begin{array}{l} \text{Começo} \leftarrow \text{Começo} - 1 \\ \text{Deque} [\text{Começo}] \leftarrow \text{Valor} \end{array} \right.$

#### **Procedimento Remove\_Deque\_Termino (Deque, Termino, Começo)**

Se (Deque\_Vazio (Deque, Termino, Começo))

Entao ERRO

Senao Termino  $\leftarrow$  Termino - 1

**Funcao Consulta\_Deque\_Termino (Deque, Termino, Comeco) : Valor**

Se (Deque\_Vazio(Deque, Termino, Comeco))

Entao ERRO

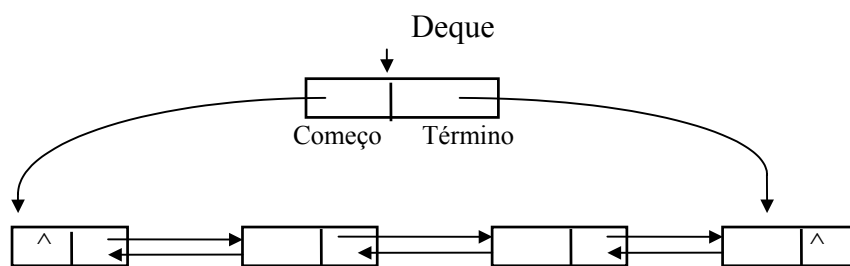
Senao Valor  $\leftarrow$  Deque [Término]

**Funcao Deque\_Vazio**

*{idem a fila}*

**■ Por Encadeamento**

*{é o mesmo que a lista c/ header, porém este não tem o campo QTD }*

Adaptação do Algoritmo:

→ No Deque testa se Começo e Término = NIL (Deque Vazio) e se Começo = Término (1 elemento)

**{ os procedimentos da lista são os mesmos para o deque }**

**Instituto Luterano de Ensino Superior de Ji-Paraná**  
**Curso Bacharelado em Informática**  
**Estrutura de Dados I**  
**Prof.: José Luiz A. Duizith**

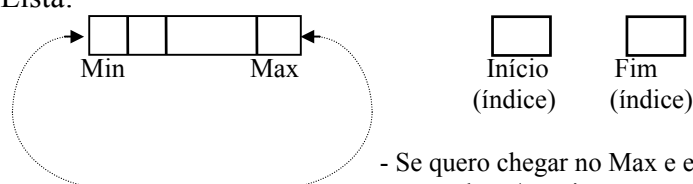
## LISTA CIRCULAR

Listas cujas extremidades estão ligadas

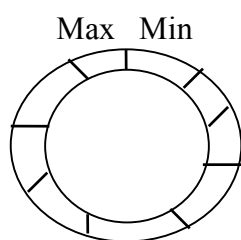
### (A) Representação

#### ■ Por Contiguidade

Lista:

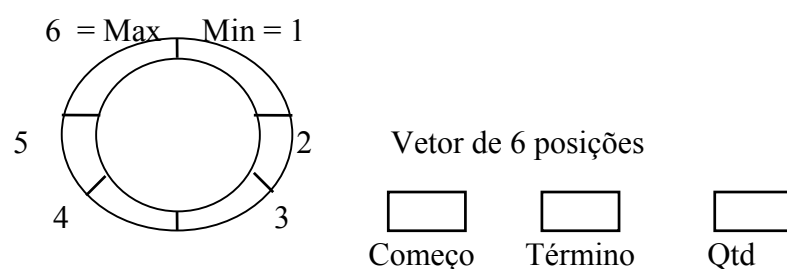


- Se quero chegar no Max e ele está cheio então tenho que pular p/ o Min e recomeçar.



**Ex:**

Fila Circular Contigua



#### **Procedimento Cria\_Fila\_Circular (Fila,Começo,Término,Qtd,Min,Max)**

Cria\_Vetor (Fila,Min,Max)

Começo ← Min                    {Começo = Min → começo = 1}

Término ← Min - 1            {Término = Min - 1 → Término = 0}

Qtd ← 0

#### **Funcao Fila\_Circular\_Vazia (Fila,Qtd) : Logico**

Logico ← Qtd = 0

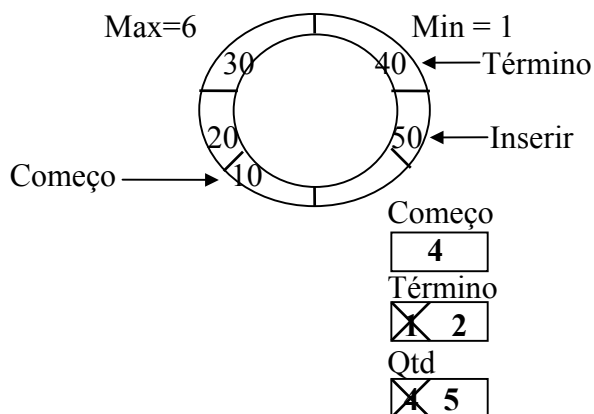
**Obs.:** para testar se está cheia, é sempre pela qtd de elementos.

#### **Procedimento Insere\_Fila\_Circular (Fila,Término,Qtd,Min,Max,Valor)**

*{Insere no término}*

```

Se (Qtd = Max - Min + 1)  {Lista Cheia}
Entao ERRO                { Overflow}
Senao Se (Termino = Max)  {está no limite}
    Entao Termino ← Min
    Senao Termino ← Termino + 1
    Fila [Termino] ← Valor
    Qtd ← Qtd + 1
  
```

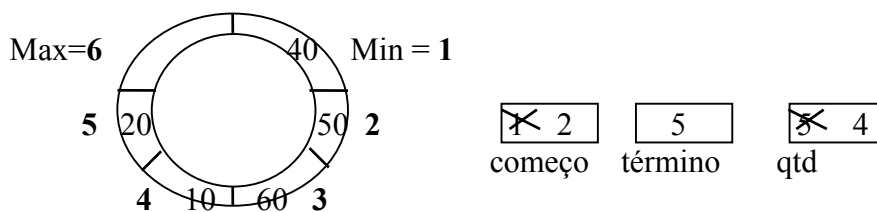


### Procedimento Remove\_Fila\_Circular (Fila,Começo,Qtd,Min,Max)

*{remoção do começo}*

```

Se (Fila_Circular_Vazia(Fila,Qtd))
Entao ERRO
Senao Se (Começo = Max)
    Entao Começo ← Min
    Senao Começo ← Começo + 1
    Qtd ← Qtd - 1
  
```



### Funcao Consulta\_Fila\_Circular (Fila,Começo,Qtd) : Valor

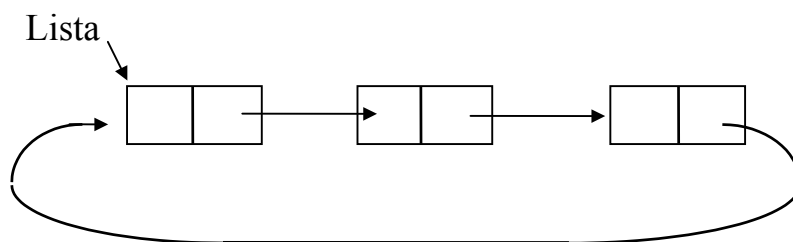
*{Obs.: Só posso consultar no começo}*

```

Se (Fila_Circular_Vazia (Fila,Qtd))
Entao ERRO
Senao Valor ← Fila [Começo]
  
```

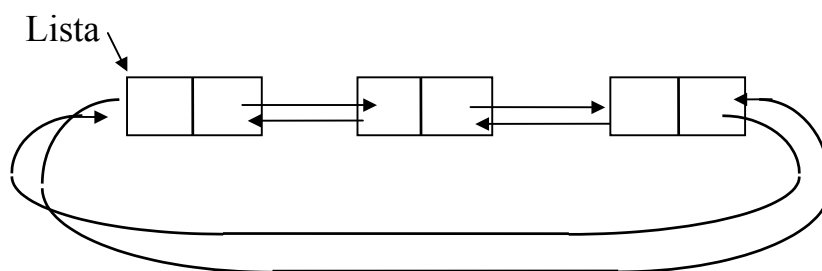
■ Por Encadeamento:

■ Simplesmente

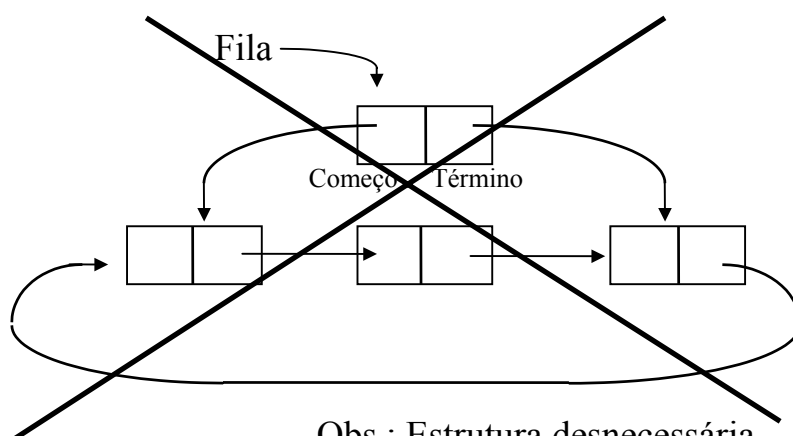


- último nodo não será mais NIL, pois ele aponta para o Começo.

■ Duplamente

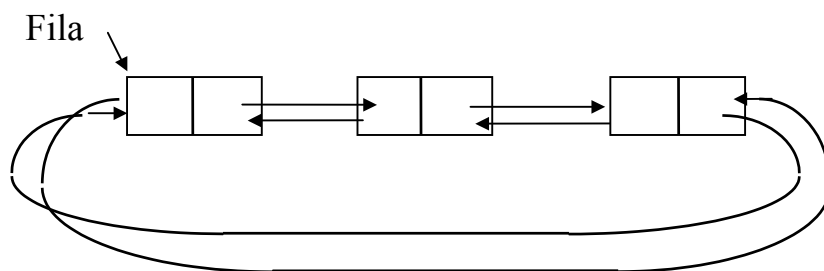


Ex: Fila Circular Encadeada (c/ header)



Obs.: Estrutura desnecessária  
Fila c/ header (não precisa ser circular)

■ Fila Circular Duplamente (s/ header)



Obs.: Fila s/ header - precisa ser circular

Procedimento Cria\_Fila\_Circular (Fila)

Fila  $\leftarrow$  NIL

Função Fila\_Circular\_Vazia (Fila) : Lógico

Lógico  $\leftarrow$  Fila = NIL

Procedimento Destroi\_Fila\_Circular (Fila)

Se (Não Fila\_Circular\_Vazia(Fila))

Então Aux  $\leftarrow$  Fila $\uparrow$ .Ant

Enquanto (Fila  $\diamond$  Aux)

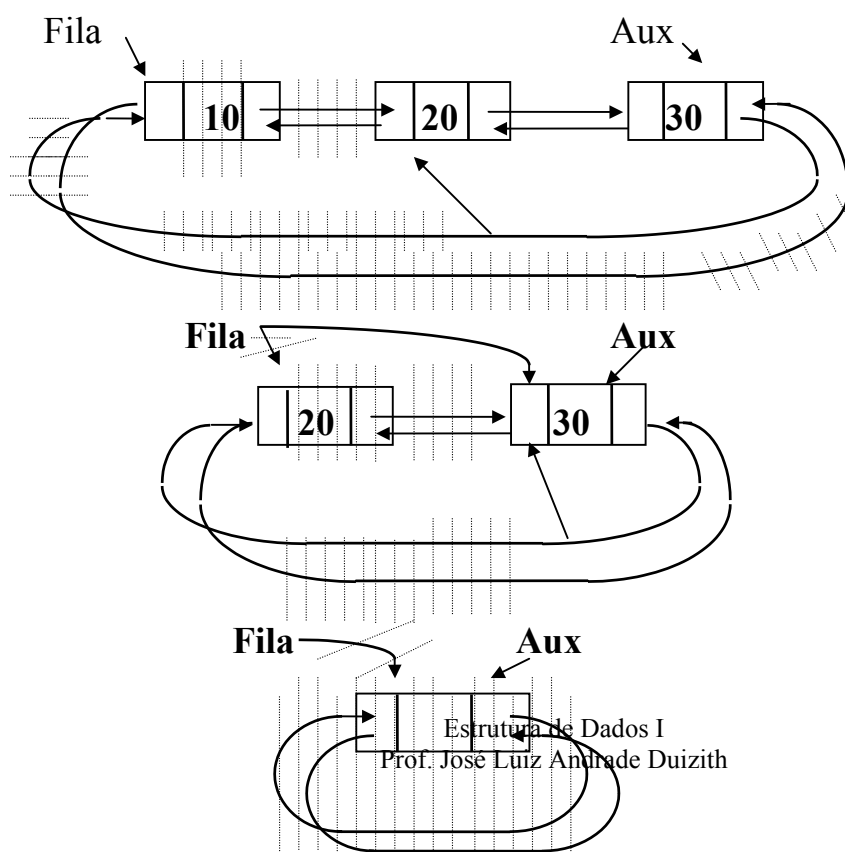
Faça Aux $\uparrow$ .Prox  $\leftarrow$  Fila $\uparrow$ .Prox

Desaloque (Fila)

Fila  $\leftarrow$  Aux $\uparrow$ .Prox

Desaloque (Fila)

Fila  $\leftarrow$  NIL



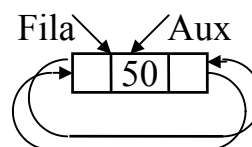
## 30

**Fila** → ^

Procedimento Insere\_Fila\_Circular (Fila, Valor)

```

Aloque (Aux)
Se (Aux = NIL)
Entao ERRO
Senao | Aux↑.Dado ← Valor
      | Se (Fila_Circular_Vazia (Fila))
      | Entao | Fila ← Aux
      |       | Aux↑.Ant ← Aux
      |       | Aux↑.Prox ← Aux
      | Senao | Aux↑.Ant ← Fila↑.Ant
      |       | Aux↑.Prox ← Fila
      |       | Fila↑.Ant↑.Prox ← Aux
      |       | Fila↑.Ant ← Aux
  
```



Função Consulta\_Fila\_Circular (Fila) : Valor

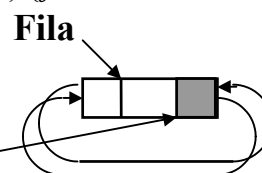
```

Se (Fila_Circular_Vazia(Fila))
Entao ERRO
Senao Valor ← Fila↑.Dado
  
```

Procedimento Remove\_Fila\_Circular (Fila)

```

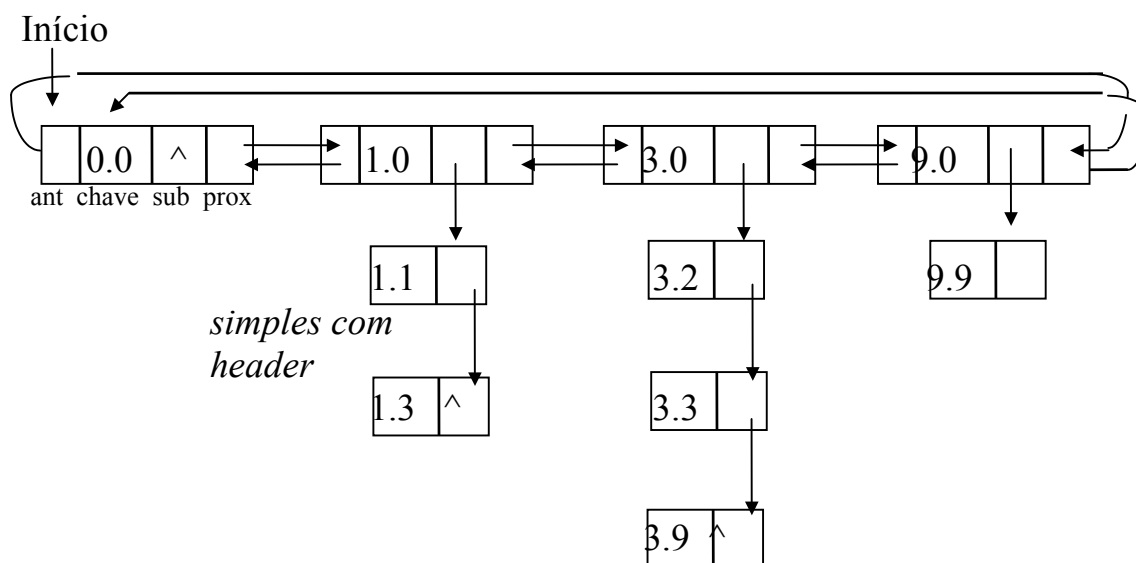
Se (Fila_Circular_Vazia(Fila))
Entao ERRO
Senao | Aux ← Fila
      | Se (Fila↑.Ant = Fila) E (Fila↑.Prox = Fila) {fila de 1 só nodo}
      | Entao Fila ← Nil
      | Senao | Fila ← Aux↑.Prox
      |       | Fila↑.Ant ← Aux↑.Ant
      |       | Aux↑.Ant↑.Prox ← Fila
      | Desaloque (Aux)
  
```



**Resumo:**



Suponha a seguinte estrutura. → *lista circular duplamente encadeada sem header*



**Suponha ainda que:**

- Não devem haver chaves repetidas na estrutura e que as chaves estão classificadas em ordem crescente. (*sempre*).
- Na inserção de novas chaves se a parte fracionária for zero esta deve ser inserida na lista principal, caso contrário é inserida na sub\_lista da chave correspondente.
- Uma sub\_chave só pode ser inserida se a chave principal já existe.
- A remoção de uma chave da lista principal só pode ser realizada se todas as subchaves já foram removidas.
- Inicialmente : Início  $\leftarrow$  NIL.