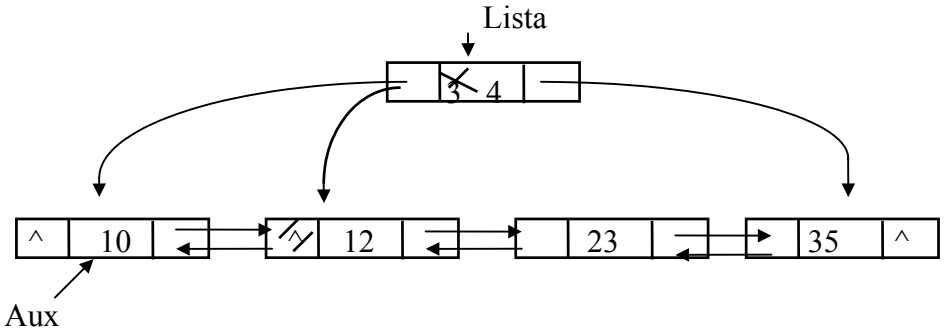
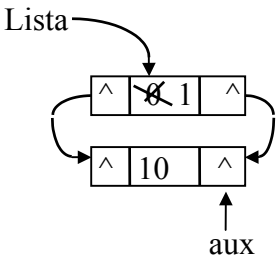
Instituto Luterano de Ensino Superior de Ji-Paraná
Curso Bacharelado em Informática
Estrutura de Dados I
Prof.: José Luiz A. Duizith

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
```



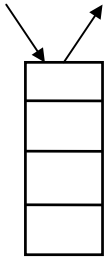


Listas com Disciplinas de Acesso

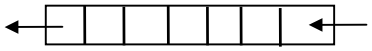
- Critérios usuais para inclusão e remoção de nodos em listas:
  - **LIFO** ("Last In First Out" → Último a entrar e 1º a sair) : dentre os elementos que ainda permanecem no conjunto, o primeiro elemento a ser retirado é o último que foi inserido.
  - **FIFO** ("First In First Out" → 1º a entrar e 1º 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 FIFO)
- Fila (critério FIFO)
- Deque (híbrido)

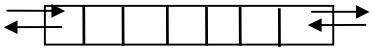
Pilha



Fila



Deque (Inclui e exclui em qualquer extremidade)



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:**

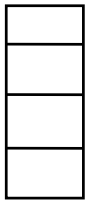
Criação
Destruição
Empilhar (Push) → inserção de novo elemento no topo da pilha.
Desempilhar (Pop) → remoção de elementos do topo da pilha.
Consultar (Top) → Consulta o elemento do topo da pilha.

Verificação se pilha está vazia (EMPTY)

(B) **Representação:**

- Por Contigüidade

Pilha



Topo:

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)

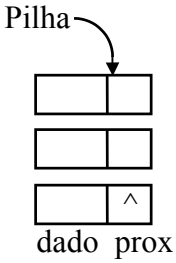
```
Se (Verifica_Pilha (Pilha,Topo,Min))
Entao  ERRO
Senao  Topo ← Topo - 1
```

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
```

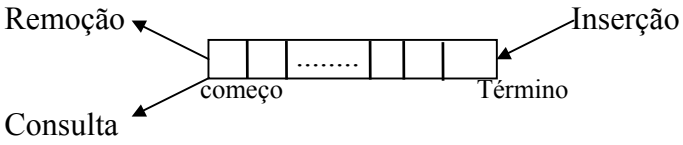
Procedimento Pop (Pilha)

```
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:

Fila

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
```

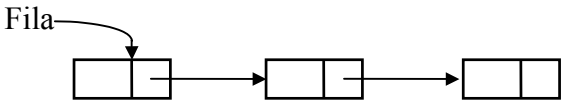
Procedimento Remove_Fila (Fila, Comeco, Termino)

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

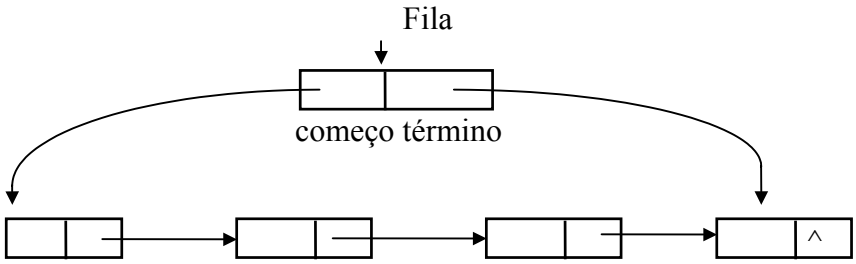
Funcao Consulta_Fila (Fila, Comeco, Termino) : Valor

```
        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")

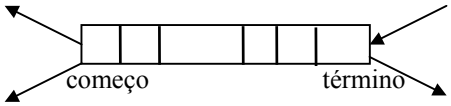
Queue - Fila

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 | Comeco ← Começo - 1
      | Deque [Começo] ←Valor
```

Procedimento Remove_Deque_Termino (Deque,Termino,Começo)

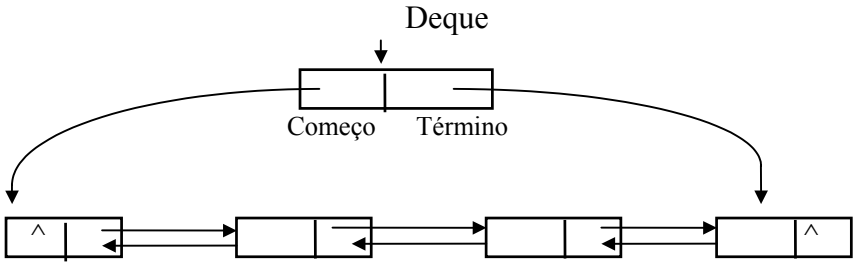
```
Se (Deque_Vazio (Deque,Termino,Comeco))
Entao ERRO
Senao Termino ← Termino - 1
```

Funcao Consulta_Deque_Termino (Deque, Termino, Comeco) : Valor
Se (Deque_Vazio(Deque,Termino,Começo))
Entao ERRO
Senao Valor ← 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 Algorítimo:

→ 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

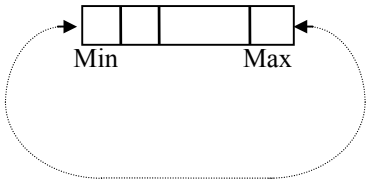
LISTA CIRCULAR

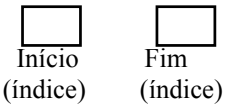
Listas cujas extremidades estão ligadas

(A) **Representação**

- Por Contiguidade

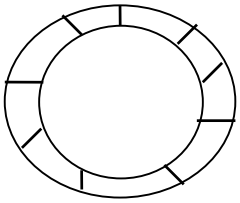
Lista:





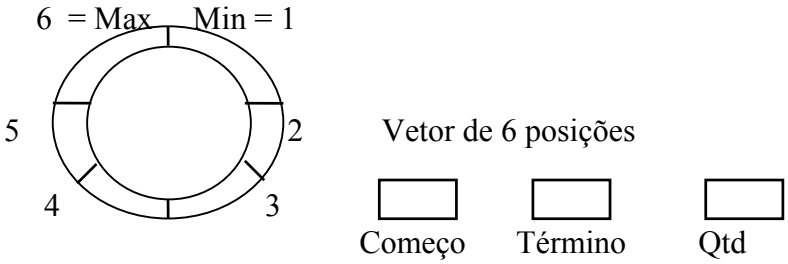
- Se quero chegar no Max e ele está cheio então tenho que que pular p/ o Min e recomeçar.

Max Min



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}
Termino ← Min - 1 {Termino = Min -1 → Termino = 0}
Qtd ← 0

Funcao Fila_Circular_Vazia (Fila,Qtd) : Logico

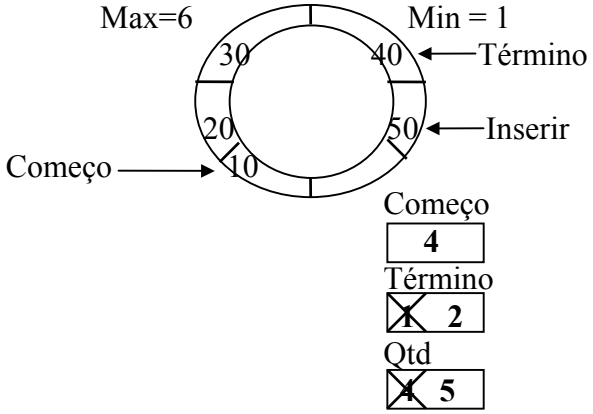
Logico ← Qtd = 0

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

Procedimento Insere_Fila_Circular (Fila,Termino,Qtd,Min,Max,Valor)

{Insere no término}

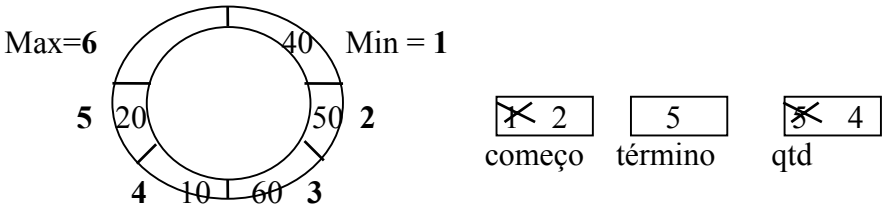
```
Se (Qtd = Max - Min + 1)     {Lista Cheia}
Entao ERRO                   { Overflow}
Senao Se (Termino = Max)     {está no limite}
      Entao Termino ← Min
      Senao Termino ← Termino + 1
      Fila [Termino] ← Valor
      Qtd ← Qtd + 1
```



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
```



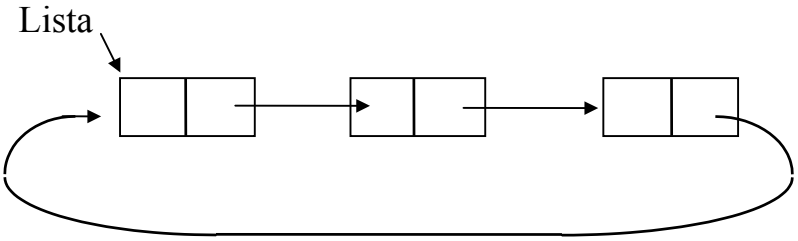
Funcao Consulta_Fila_Circular (Fila,Começo,Qtd) : Valor

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

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

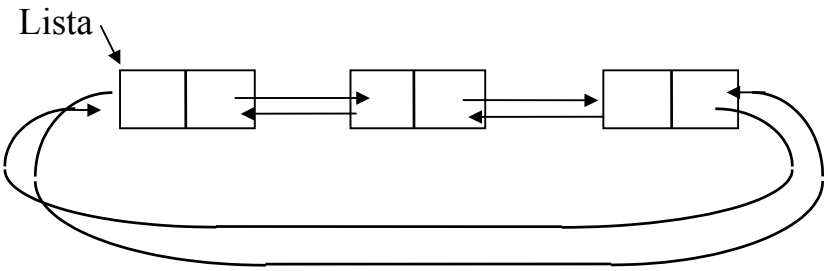
■ Por Encadeamento:

■ Simplesmente

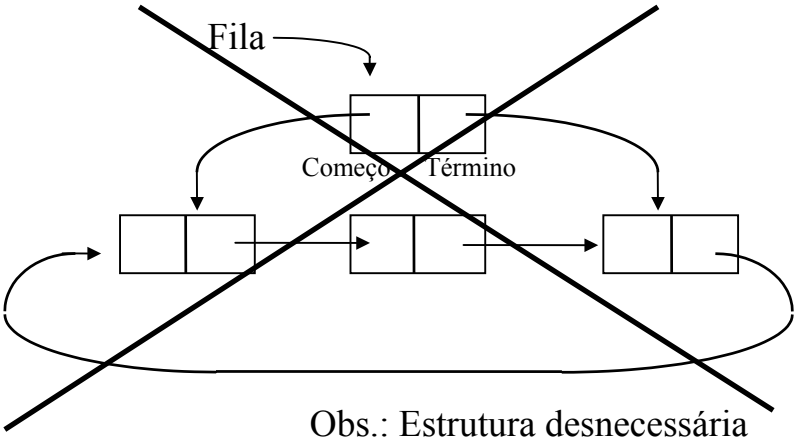


- último nodo não será mais NIL, pois ele aponta para o Começo.

■ Duplamente

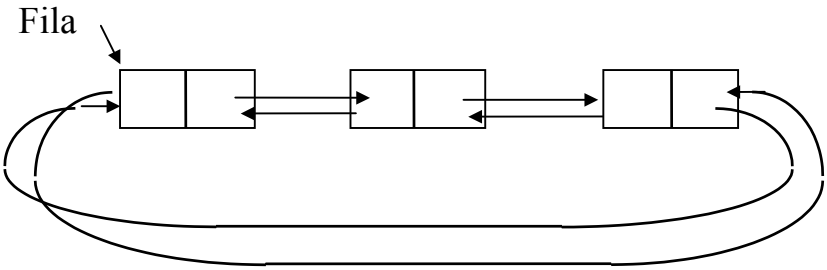


Ex: Fila Circular Encadeada (c/ header)



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

■ Fila Circular Duplamente (s/ header)

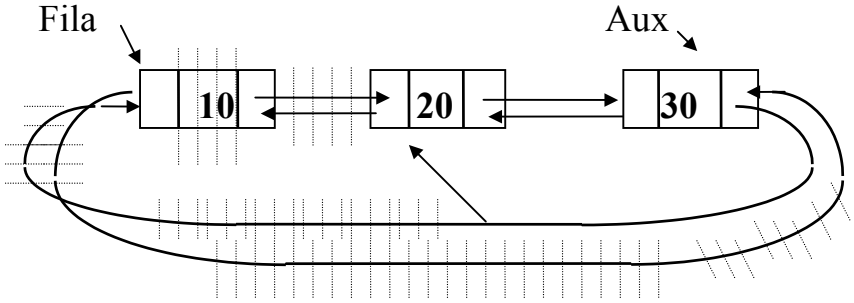


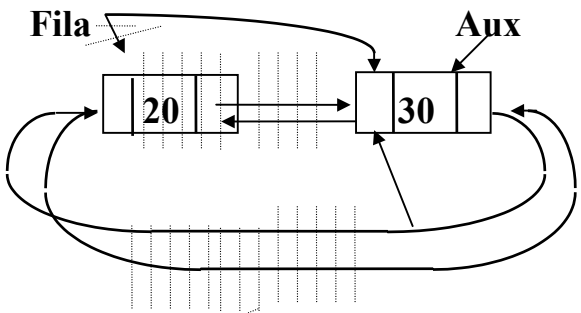
Obs.: Fila s/ header - precisa ser circular

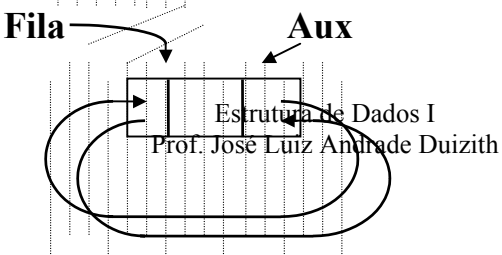
```
Procedimento Cria_Fila_Circular (Fila)
        Fila ← NIL

Função Fila_Circular_Vazia (Fila) : Lógico
        Lógico ← Fila = NIL

Procedimento Destroi_Fila_Circular (Fila)
        Se (Não Fila_Circular_Vazia(Fila))
        Entao|Aux ← Fila↑.Ant
             |Enquanto (Fila <> Aux)
             |Faça |Aux↑.Prox ← Fila↑.Prox
             |     |Desaloque (Fila)
             |     |Fila ← Aux↑.Prox
             |Desaloque (Fila)
             |Fila ← NIL
```



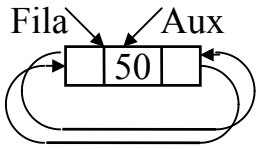




30

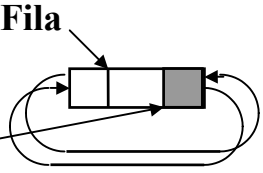
Fila → ^

```
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)
```



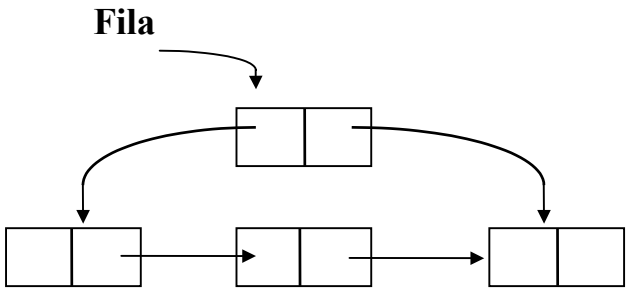
Resumo:

Lista Lineares (Genérica, Fila, Pilha, Deque)

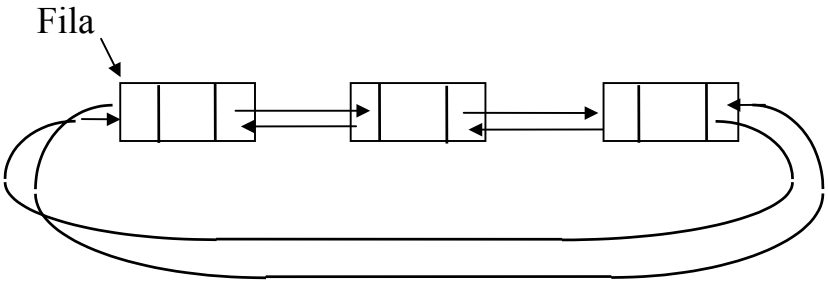
não Circular → Contígua
Circular → Encadeada → Simples → s/ header
Dupla → c/ header

- Fila simplesmente c/ header → Estrutura desnecessária
- Fila c/ header → nÃo precisa ser circular
- Fila s/ header → precisa ser circular

Fila : simples c/ header encadeada
para facilitar



Dupla s/ header circular → tem que ser circular
para facilitar o acesso direto ao último nodo.



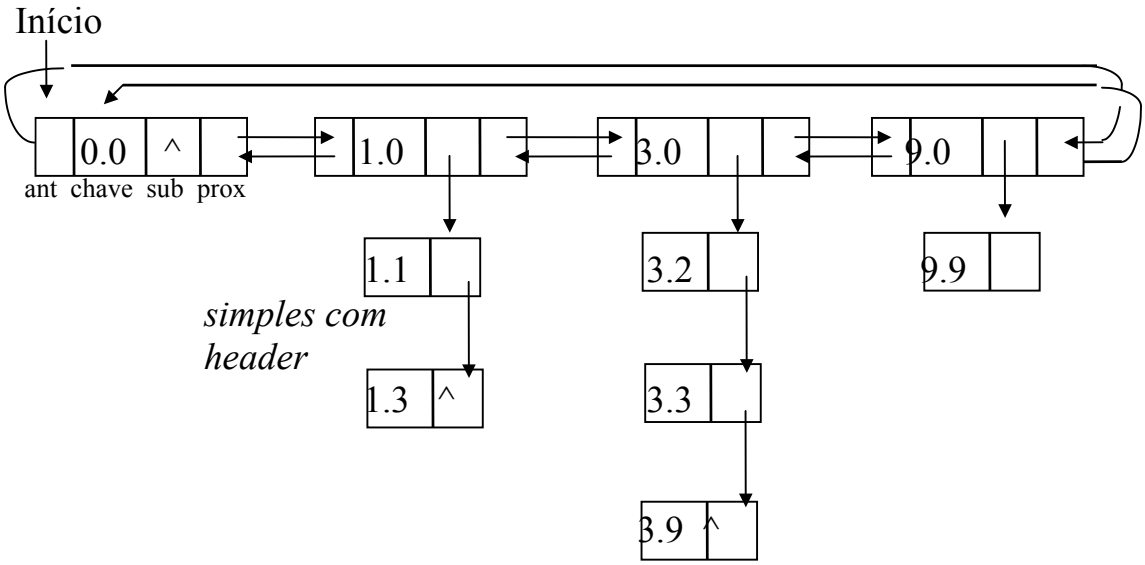
Deque→ por encadeamento → duplo → circular → c/ header
s/ header

Pilha → por encadeamento → simples → s/ header → Não circular

Listas de Listas (não é linear)

Lista na qual o nodo pode possuir a referência a outras listas.

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



Suponha ainda que:

a) Não devem haver chaves repetidas na estrutura e que as chaves estão classificadas em ordem crescente. *(sempre)*.

b) 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.

c) Uma sub_chave só pode ser inserida se a chave principal já existe.

d) A remoção de uma chave da lista principal só pode ser realizada se todas as subchaves já foram removidas.

e) Inicialmente : Início ← NIL.