# Estrutura Abstrata de Dados

# FILA

## Implementação com ligações simples

## (usando ponteiros e memória dinâmica)

# Conceitos gerais

## Elementos

- Os elementos duma **Fila** (*Queue*) são processados pela ordem de chegada
  - o **primeiro** elemento a **entrar** é o **primeiro** a **sair (FIFO** - "First In First Out")

## Operações

- Algumas operações realizam-se na **frente** e outras na **cauda** da Fila
  - a **inserção** de um novo elemento
    - consiste em acrescentar este elemento na cauda da fila (torna-se a cauda)
  - a **remoção** de um elemento
    - retira o elemento que se encontra na frente da fila
    - torna o elemento que está a seguir, caso exista, na frente da fila
    - só é possível de realizar se a Fila não estiver vazia
  - a **fila vazia**
    - acontece depois de ser retirado o último elemento da fila
    - só é possível realizar a operação de inserção na fila

# Operações básicas (únicas) sobre uma EAD FILA

## Criar uma Fila (vazia)

Q = **criarFila** ()

## Verificar se uma Fila está vazia

**filaVazia** (Q)

## Inserir um novo elemento numa Fila (na cauda)

Q = **juntar** (X, Q)

## Remover um elemento de uma Fila (o que está na frente)

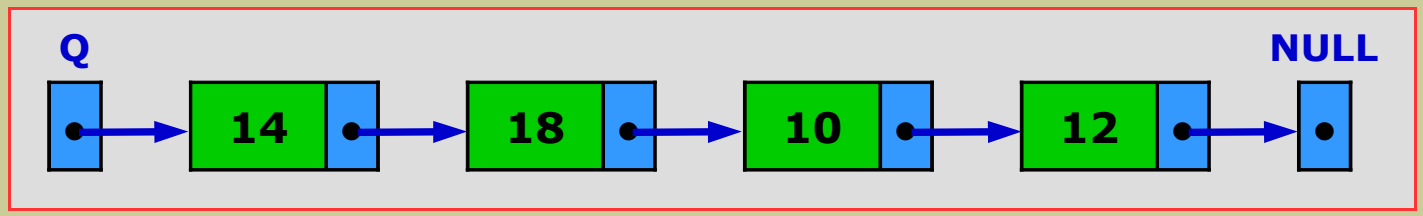
Q = **remover** (Q)

## Consultar uma Fila (devolver o elemento da frente)

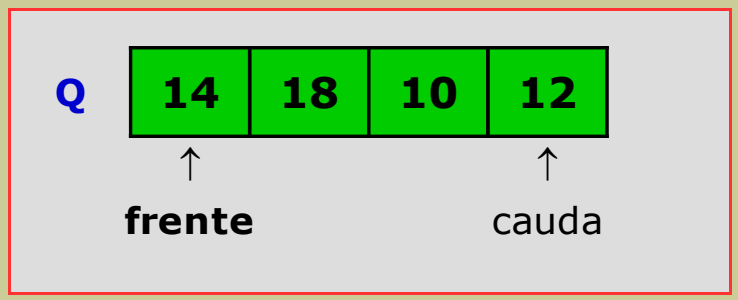
X = **frente** (Q)

# Representação gráfica

## EAD Fila

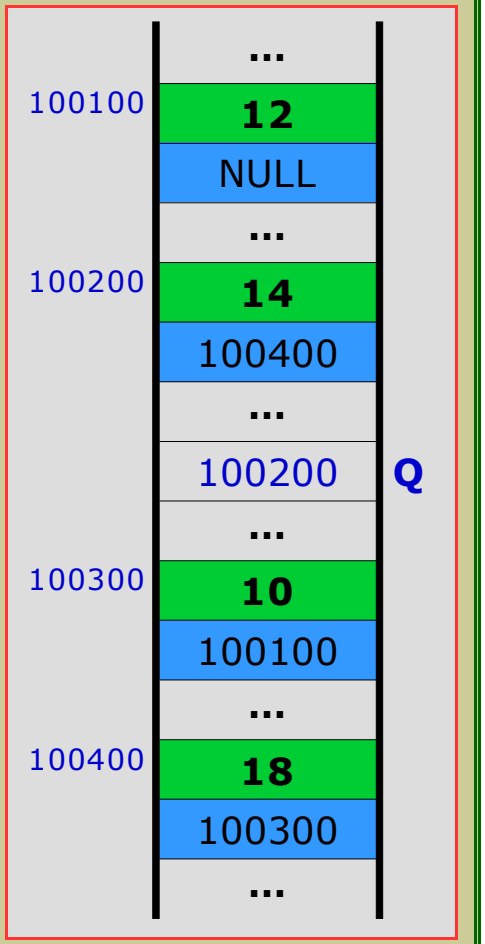



## Analogia




## Identificador Q

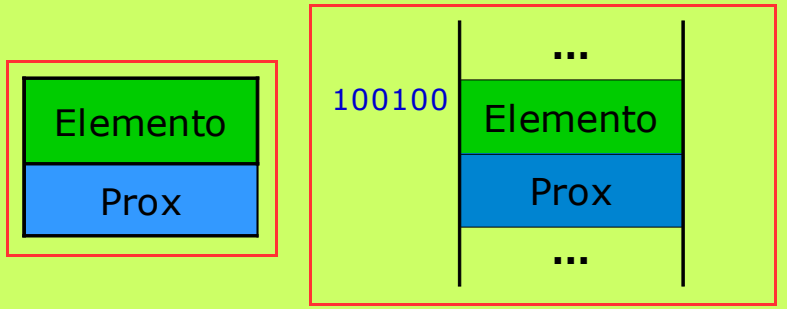
- É um **ponteiro** para a **frente** da **fila**



# Nodos e ligações

## Definição de um nodo de uma AED FILA (em linguagem C)

```
struct NodoFila {
    DadosFila  Elemento;
    struct NodoFila *Prox;
};
typedef struct NodoFila *PNodoFila;
```




- Cada nodo aponta para o nodo seguinte da fila
- O nodo da cauda da fila aponta para NULL
- A memória para os elementos (nodos) é
  - atribuída quando um elemento é inserido na fila
  - libertada quando um elemento é removido da fila

# Implementação

## Utilização de um ponteiro associado à frente da Fila

- Ponteiro que aponta para o elemento mais antigo da fila
- Será o primeiro a ser removido
- Permite o acesso à cauda da fila
    - é a seguir à cauda que um novo elemento será inserido
- Quando nulo (NULL) significa fila vazia

## Fila vazia

- Quando a frente da fila é um ponteiro nulo (NULL)

## Fila cheia

- Quando não há memória para alocar um novo elemento

# Operações sobre a estrutura DadosFila

## Mostrar os dados de um elemento do tipo DadosFila

void **mostrarElementoFila** (DadosFila X)

- operação que mostra/guarda os dados/informação do elemento X do tipo DadosFila

## Criar um elemento do tipo DadosFila

DadosFila **criarElementoFila** ()

- operação que cria um elemento do tipo DadosFila, com dados de uma fonte adequada

## Comparar dois elementos do tipo DadosFila

int **compararElementosFila** (DadosFila X, DadosFila Y)

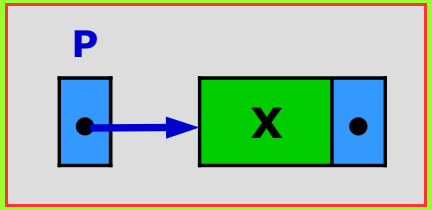
- operação que compara dois elementos do tipo DadosFila (segundo o campo chave)
- devolve: **-1** (X < Y), **0** (X = Y) ou **1** (X > Y)

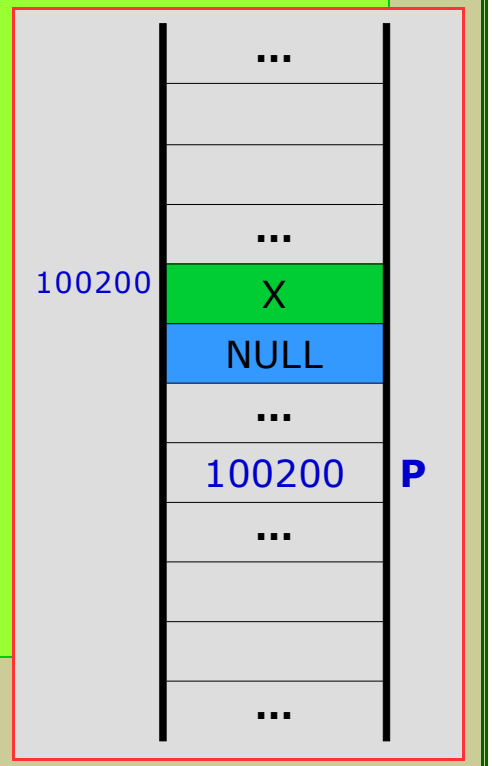
# Operações básicas sobre um nodo

## Criar um nodo de uma Fila

Entrada: a informação propriamente dita (elemento X), que fará parte de um nodo
Saída: um ponteiro para um **nodo** com informação (elemento X) e ligação a NULL

```
PNodoFila criarNodoFila (DadosFila X)
{
  PNodoFila P;
  P = (PNodoFila) malloc (sizeof(struct NodoFila));
  if (P == NULL)
     return  NULL;
  P→Elemento = X;
  P→Prox = NULL;
  return P;
}
```

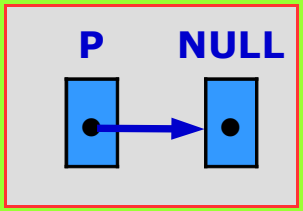


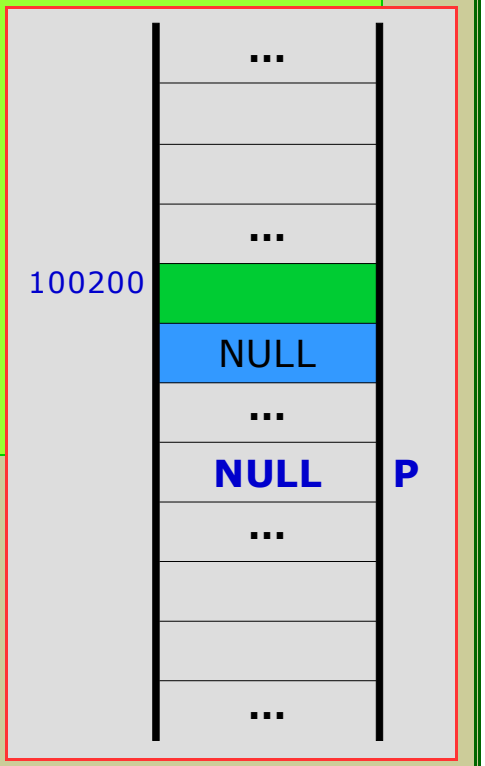



# Libertar/destruir um nodo de uma Fila

Operação: libertar todos os nodos de uma Fila
Entrada: um ponteiro para o nodo que se pretende destruir
Saída: o ponteiro a apontar para NULL

```
PNodoFila libertarNodoFila (PNodoFila P)
{
  P→Prox = NULL;
  free(P);
  P = NULL;
  return P;
}
```



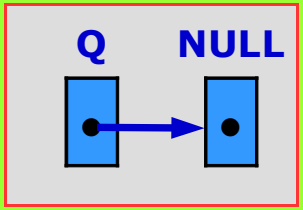
# Operações básicas (únicas) sobre uma Fila

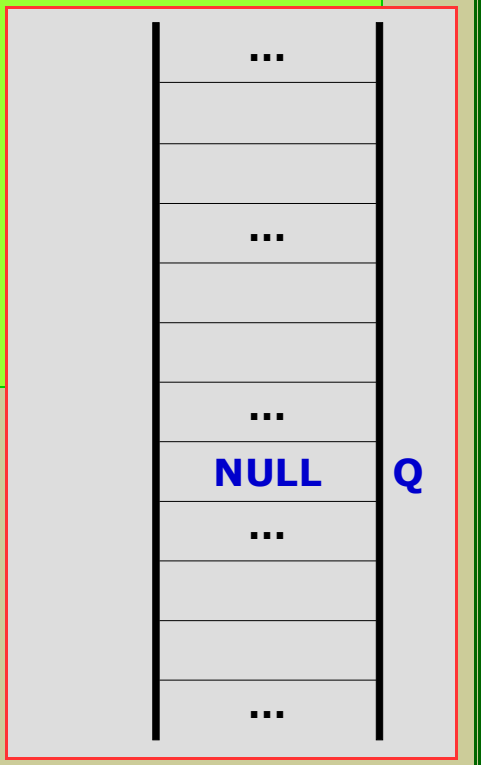
## Criar uma Fila

Entrada: nada

Saída: devolve uma Fila Q vazia (sem elementos)

```
PNodoFila criarFila ()
{
  PNodoFila  Q;
  Q = NULL;
  return  Q;
}
```



## Verificar se uma Fila está vazia

Entrada: uma Fila Q

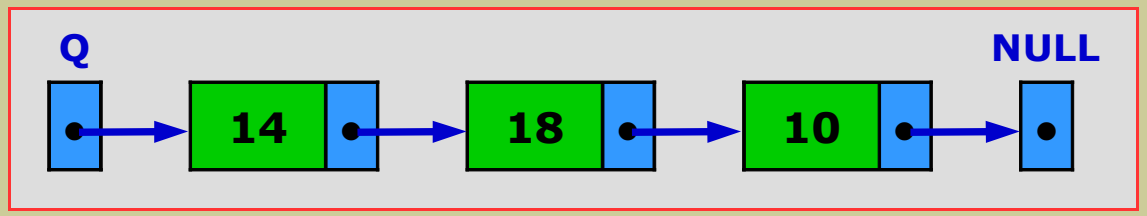
Saída: 1 (se a Fila Q está vazia) ou 0 (se a Fila Q não está vazia)

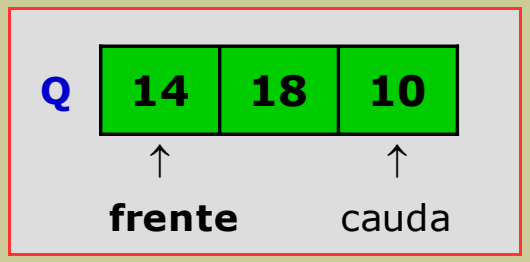
```
int filaVazia (PNodoFila Q)
{
  if (Q == NULL)
      return 1;
  else
      return 0;
}
```

# Inserir elemento numa Fila

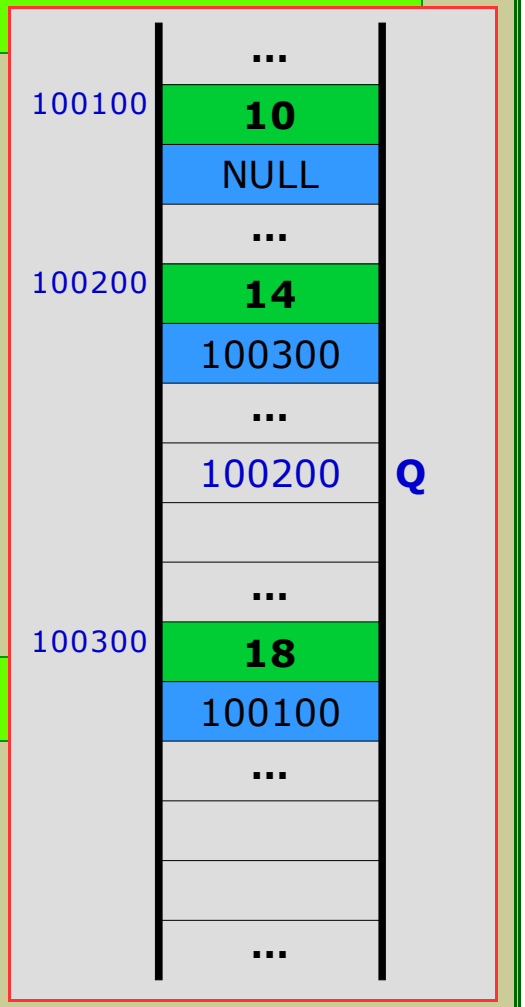
- Operação **juntar**: Q = juntar(X, Q)

Fila Q antes da operação




Q = juntar(X, Q)



# Inserir elemento numa Fila

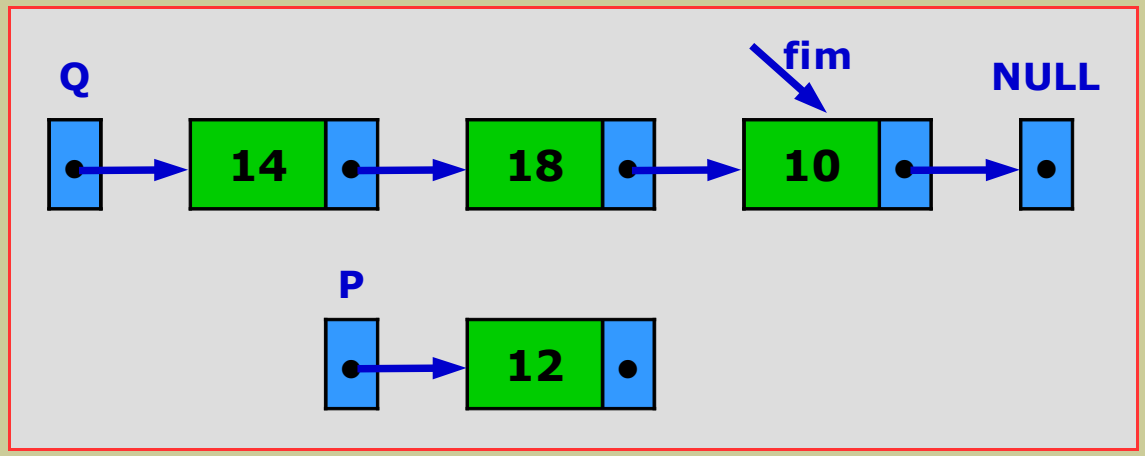
- Operação **juntar**: Q = juntar(X, Q)

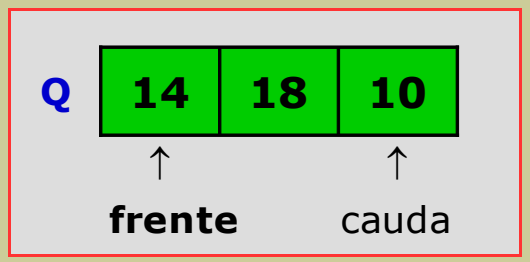
**Passo 1**

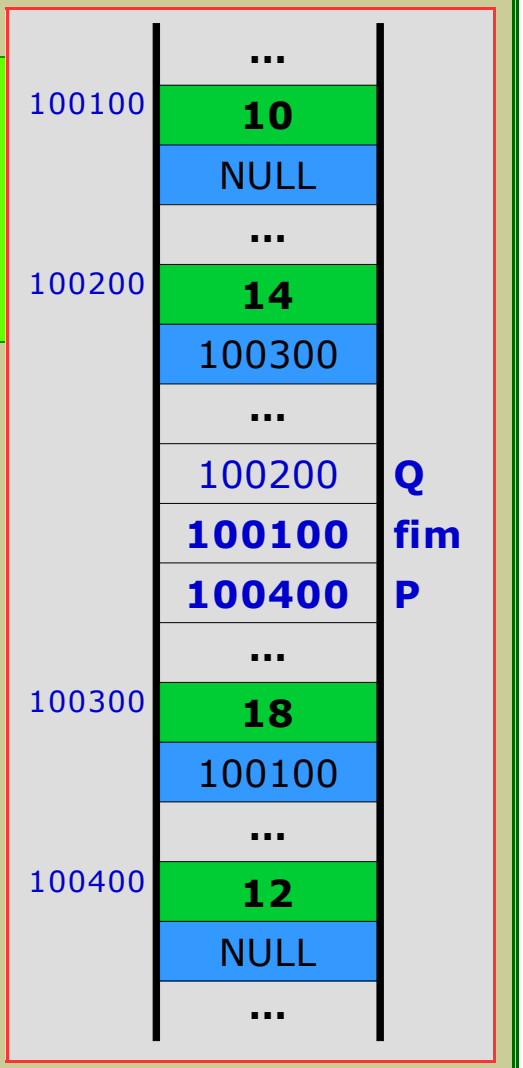
**criar** um nodo com o elemento **12** (novo), apontado por **P**

**Passo 2**

**procurar** a cauda da Fila e apontá-la com o ponteiro **fim**



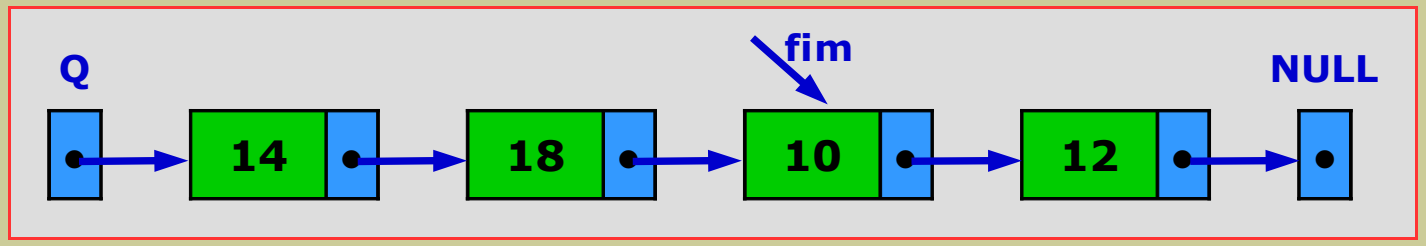



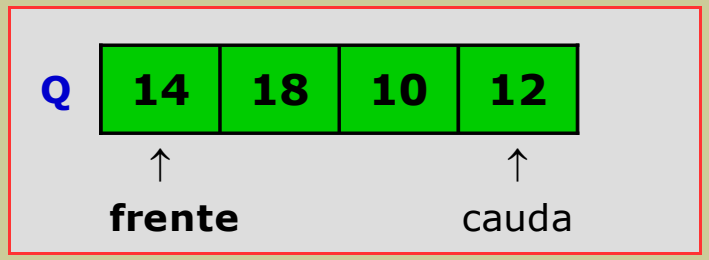
# Inserir elemento numa Fila

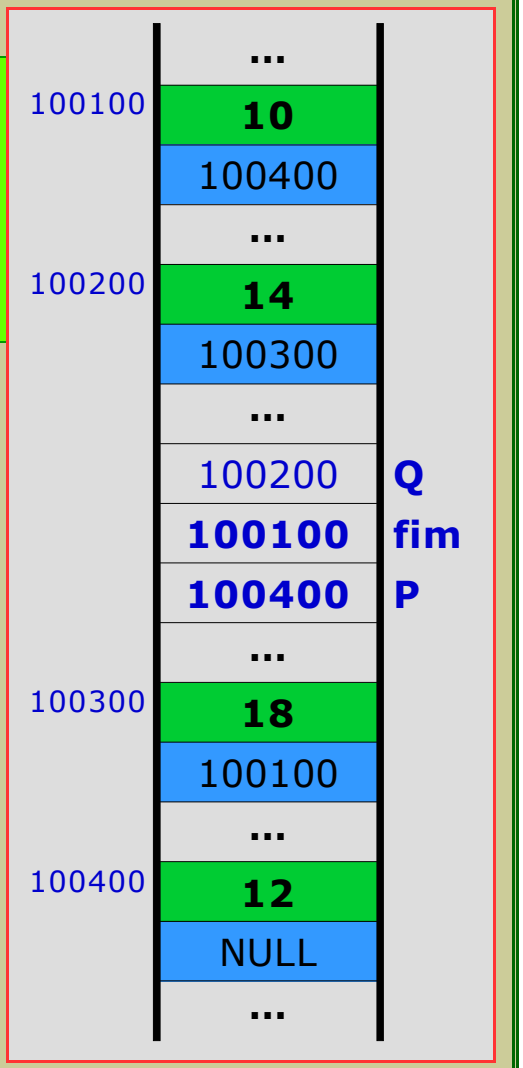
- Operação **juntar**: Q = juntar(X, Q)

**Passo 3**

**inserir** na cauda o nodo com o elemento **12** (novo elemento)
(apontado por **P**)



## Inserir elemento numa Fila

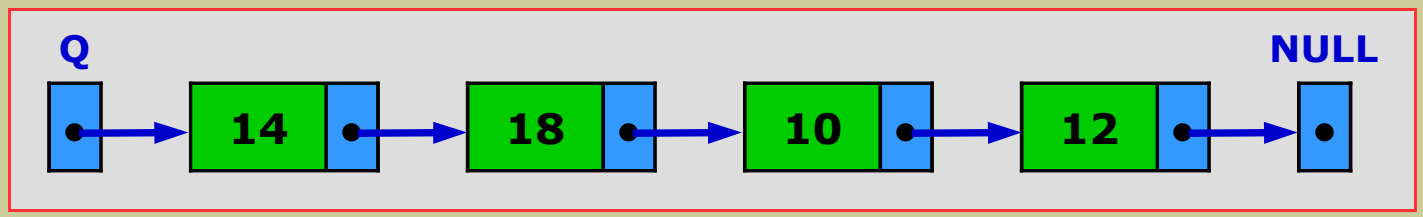
Operação: inserir um elemento na cauda de uma Fila (**juntar**)
Entrada: um elemento X a inserir e uma Fila Q
Saída: a Fila Q atualizada (com mais um elemento)

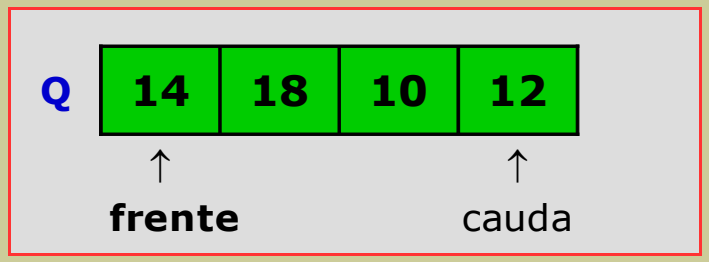
```
PNodoFila juntar (DadosFila X, PNodoFila Q) {
  PNodoFila  fim, P;
  P = criarNodoFila(X);            // passo 1
  if (P == NULL)
     return Q;
  if (filaVazia(Q))
     return P;
  fim = Q;
  while (fim→Prox != NULL)         // passo 2
     fim = fim→Prox;
  fim→Prox = P;                    // passo 3
  return Q;
}
```

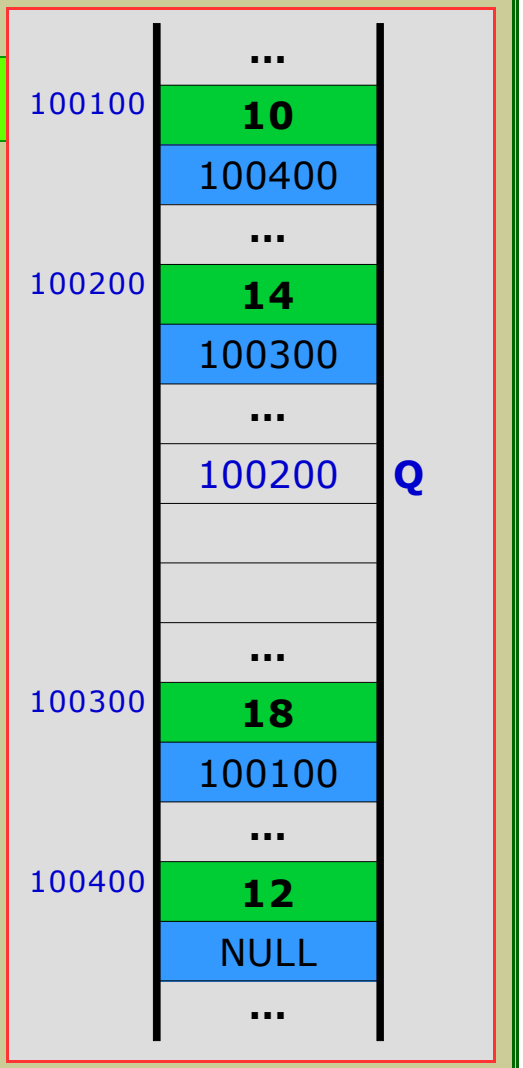
# Remover elemento de uma Fila

- Operação **remover**: Q = remover(Q)

Fila Q antes da operação



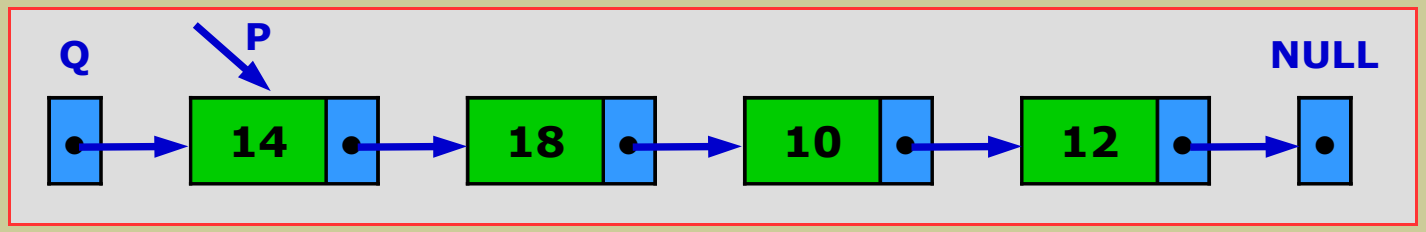



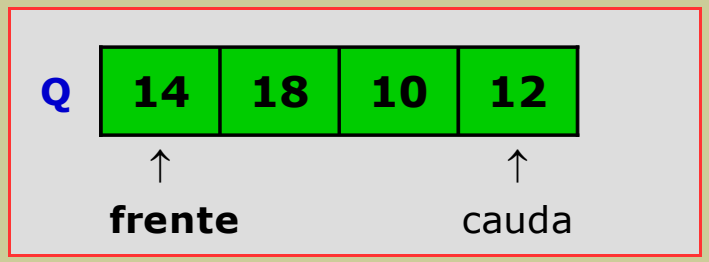
# Remover elemento de uma Fila

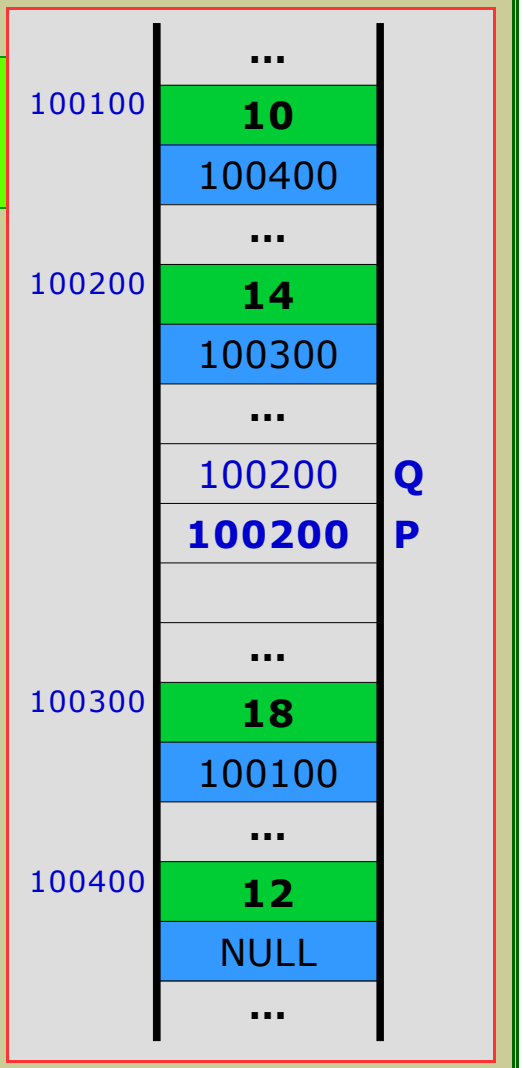
- Operação **remover**: Q = remover(Q)

**Passo 1**

**marcar** a frente da Fila Q (elemento **14**) com um ponteiro **P**



# Remover elemento de uma Fila

- Operação **remover**: Q = remover(Q)

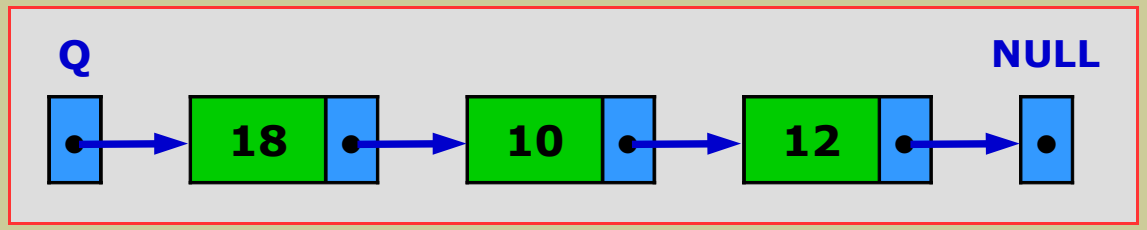
**Passo 2**

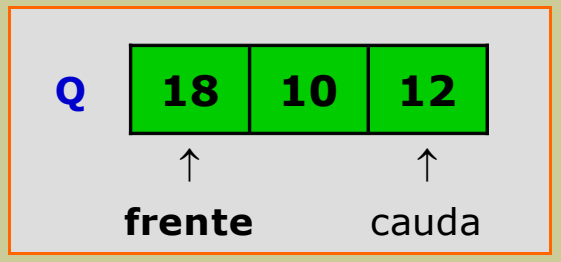
o nodo com o elemento **18** é agora a frente da Fila

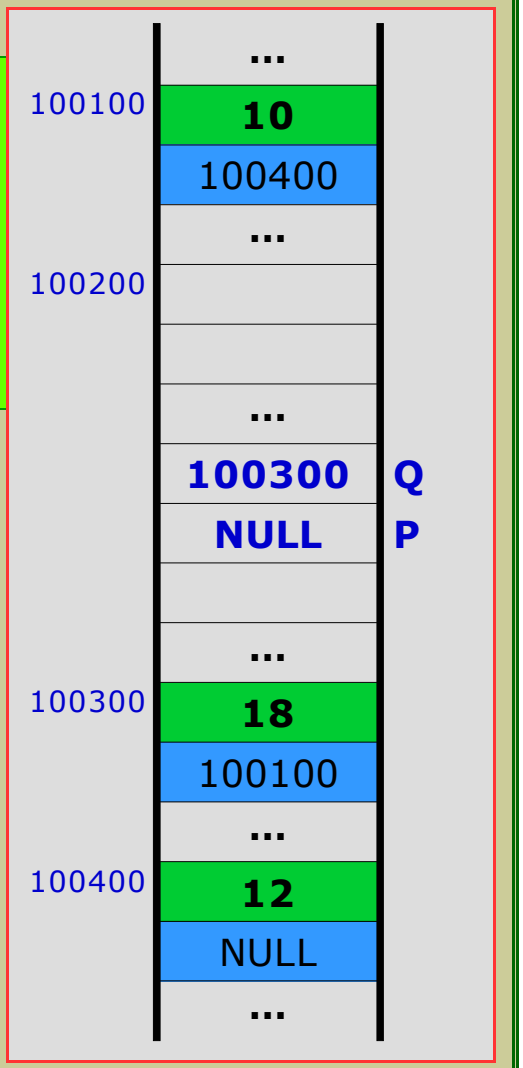
(o nodo com o elemento **14** já não pertence à Fila)

**Passo 3**

**libertar** o nodo com o elemento **14** (apontado por **P**)



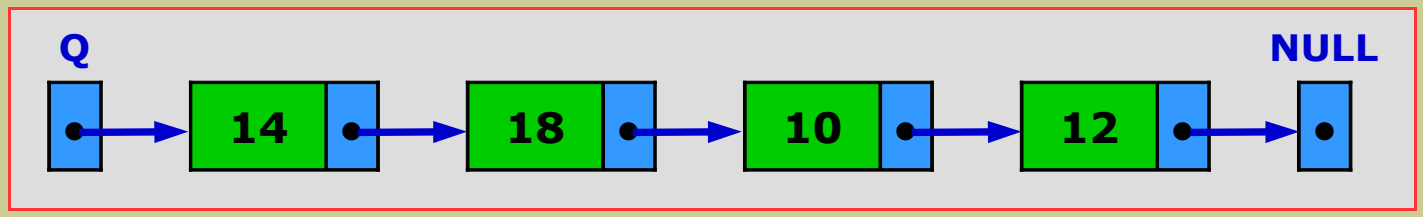
## Remover elemento de uma Fila

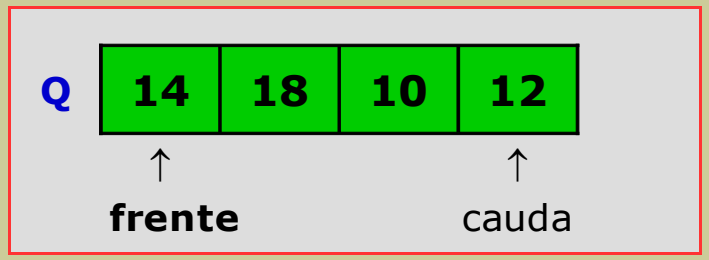
Operação: remove o elemento da frente de uma Fila (**remover**)
Entrada: uma Fila Q
Saída: a Fila Q atualizada (sem o elemento da frente)

```
PNodoFila remover (PNodoFila Q)
{
  PNodoFila P;
  P = Q;                        // passo 1
  Q = Q→Prox;                   // passo 2
  P = libertarNodoFila(P);      // passo 3
  return Q;
}
```

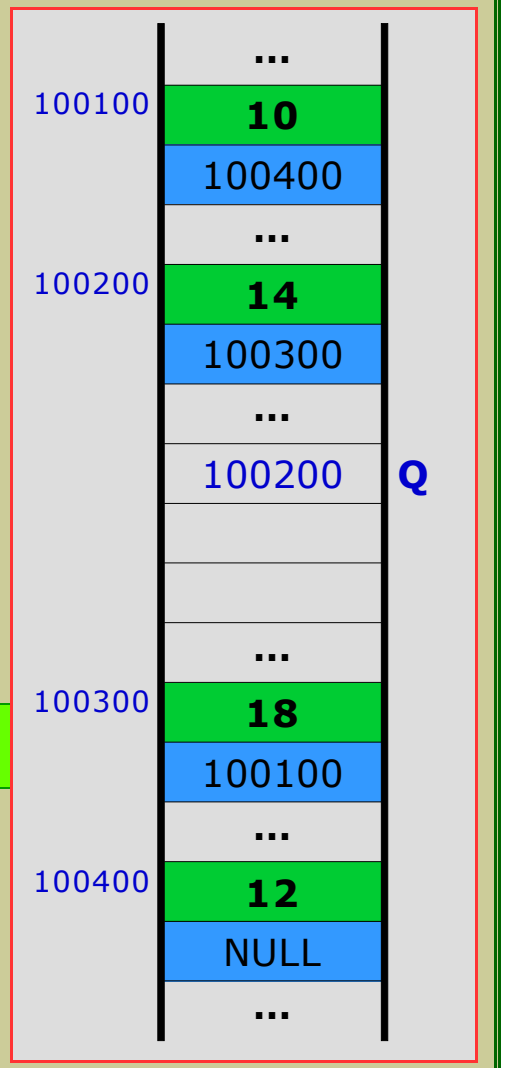
# Consultar uma Fila

- Representação gráfica




frente(Q) = **14** (Q→Elemento)



## Consultar uma Fila

Operação: consulta o elemento da frente de uma Fila (**frente**)
Entrada: uma Fila Q
Saída: o elemento que está na frente da Fila Q (do tipo DadosFila)

```
DadosFila frente (PNodoFila Q)
{
  return Q→Elemento;
}
```