UFRJ

INSTITUTO DE COMPUTAÇÃO UFRJ

Programação de Computadores II

Fila

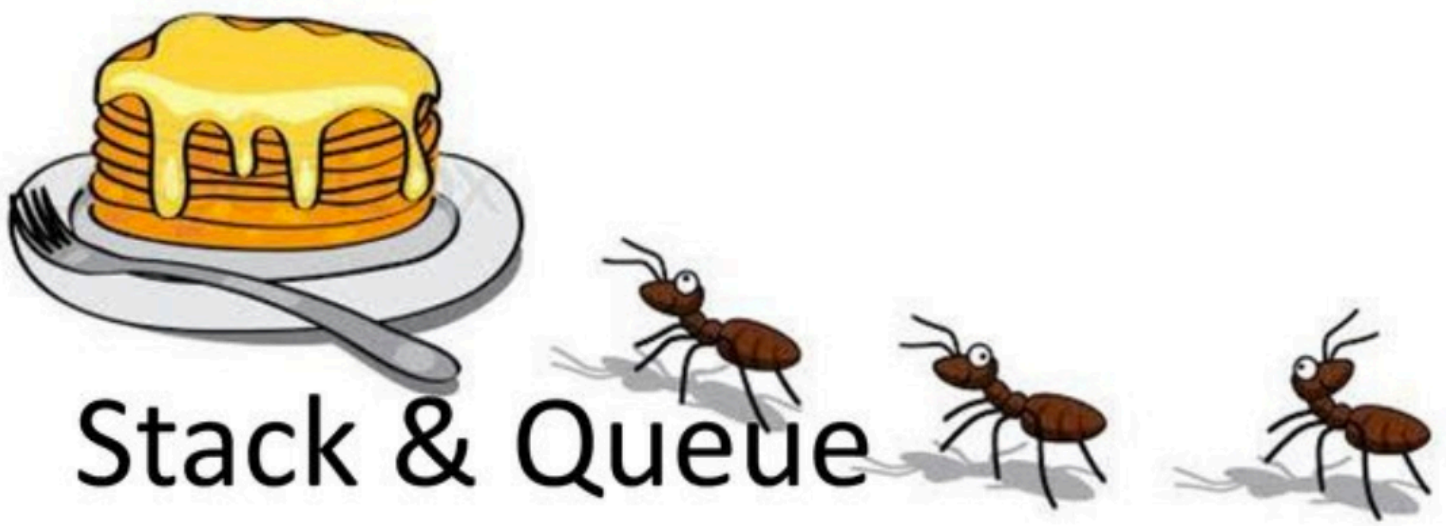
Profa. Giseli Rabello Lopes

Sumário

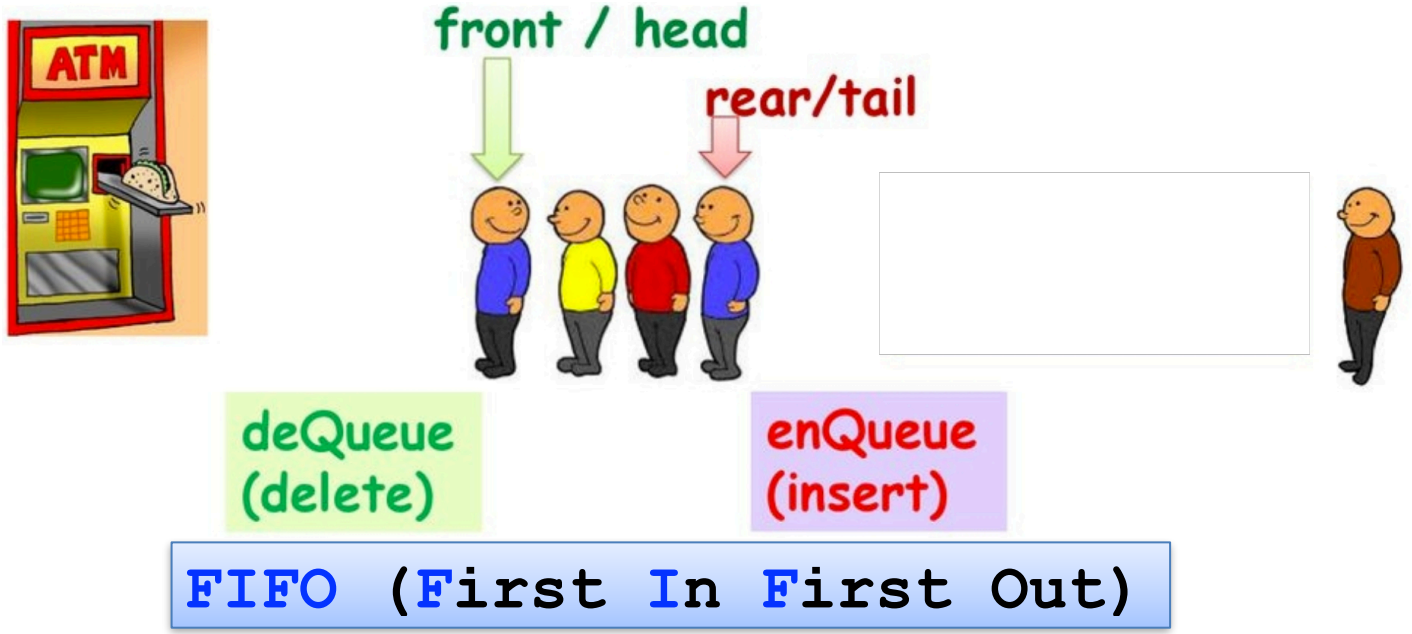
- Fila
- Funções de gerenciamento
- Implementação estática
- Implementação dinâmica

Adaptado de material preparado por Profs. Luciano Digiampietri e Norton T. Roman
e material de apoio do livro Data Structures and Algorithm Analysis in C, C++ by Mark Allen Weiss

Pilha e Fila



Fila

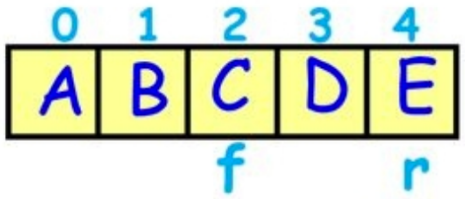


- Uma coleção ordenada de itens (estrutura linear)
- Existem duas extremidades, *head* (início) e *tail* (fim)
- Itens são inseridos no fim da fila —> **`enQueue (insert)`**
- Itens são removidos do início da fila —> **`deQueue (delete)`**

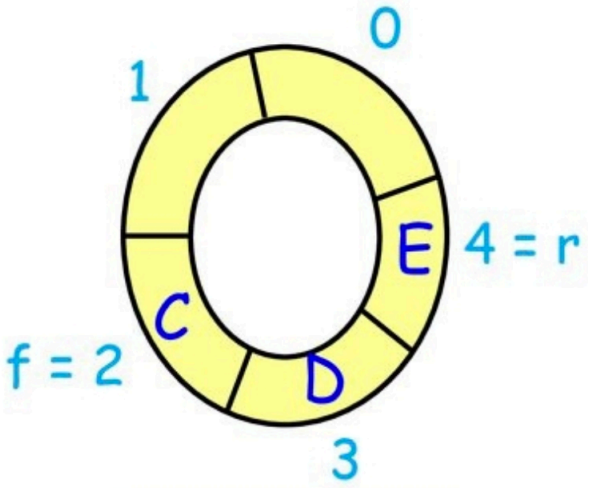
Funções de gerenciamento

- Inicializar a estrutura
- Retornar a quantidade de elementos válidos
- Exibir os elementos da estrutura
- Inserir elementos na estrutura (enQueue)
- Excluir elementos da estrutura (deQueue)
- Reinicializar a estrutura

Fila - Implementação estática



Straight array



Circular array

Fila - Implementação estática

- Utilizaremos um arranjo (*array*) de elementos de tamanho predefinido
- Controlaremos a posição do elemento que está no início da fila
- Controlaremos o número de elementos da fila

Modelagem (fila estática)

```
#include <stdio.h>
#define MAX 50

#define true 1
#define false 0

typedef int bool;

typedef int TIPOCHAVE;
```

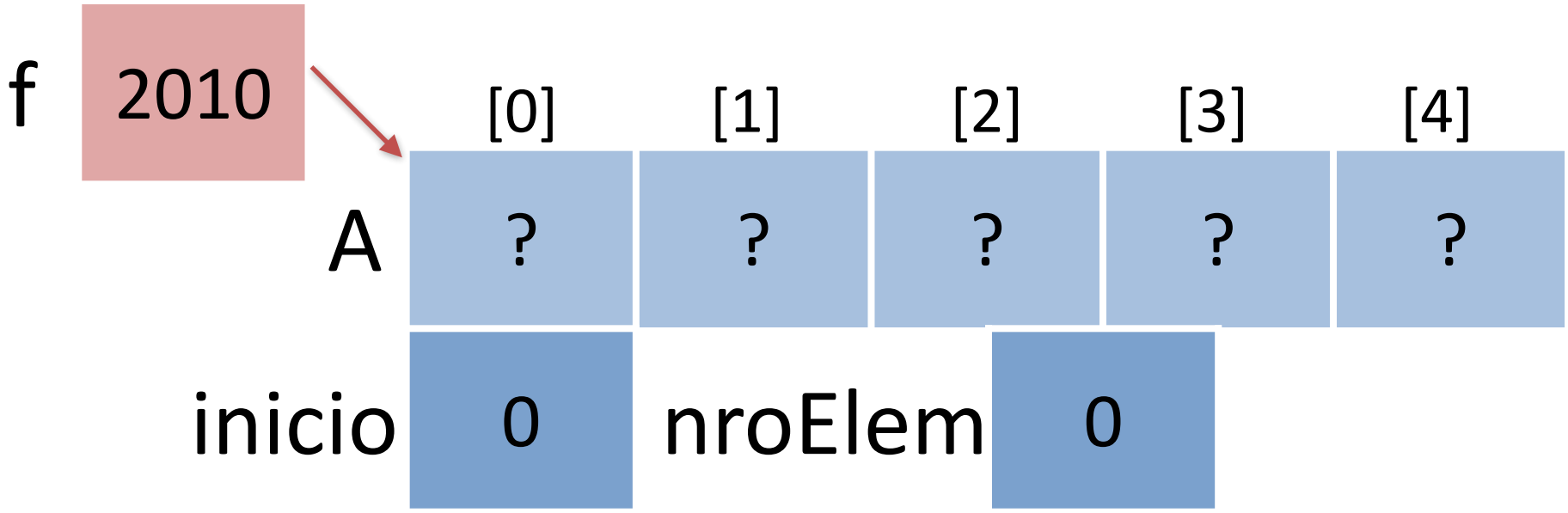
```
typedef struct {
  TIPOCHAVE chave;
} REGISTRO;

typedef struct {
  REGISTRO A[MAX];
  int inicio;
  int nroElem;
} FILA;
```

Inicialização

- Para inicializar uma fila (implementação estática), precisamos:
  - Acertar o valor do campo **`nroElem`** (para indicar que não há nenhum elemento válido)
  - Acertar o valor do campo **`inicio`** (índice do primeiro elemento válido)

Inicialização

```
void inicializarFila(FILA* f) {
  f->inicio=0;
  f->nroElem=0;
}
```

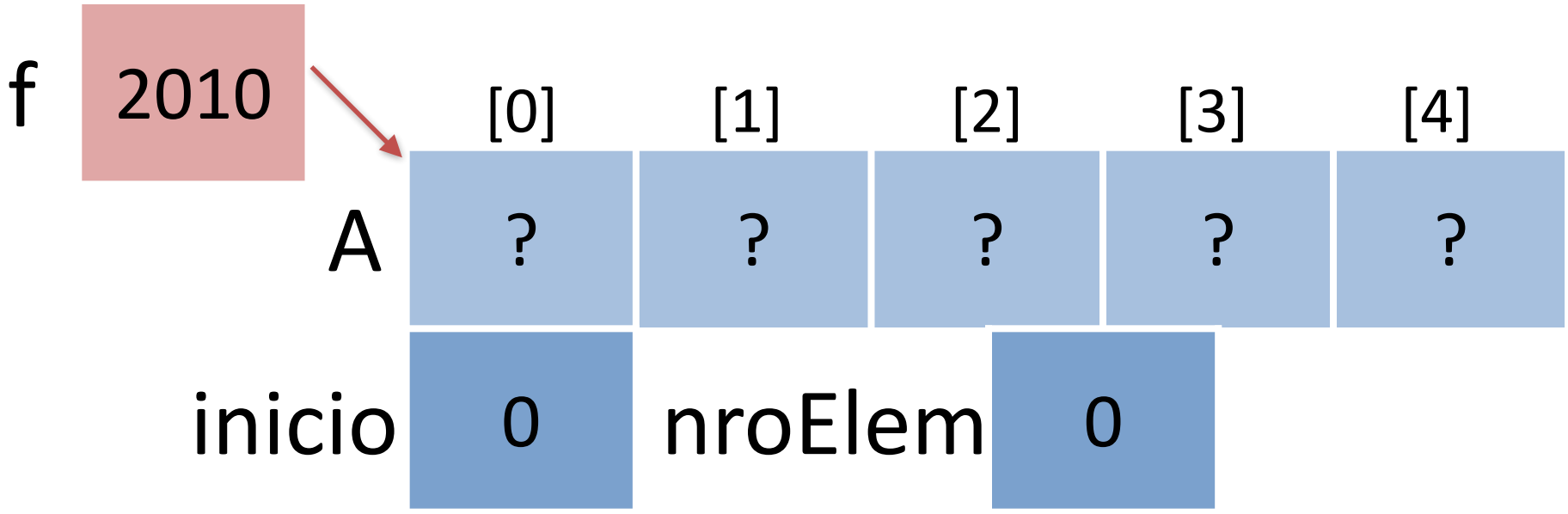


Retornar número de elementos

- Basta retornarmos o valor do campo **`nroElem`**

Retornar número de elementos

```
int tamanhoFila(FILA* f)
{
  return f->nroElem;
}
```

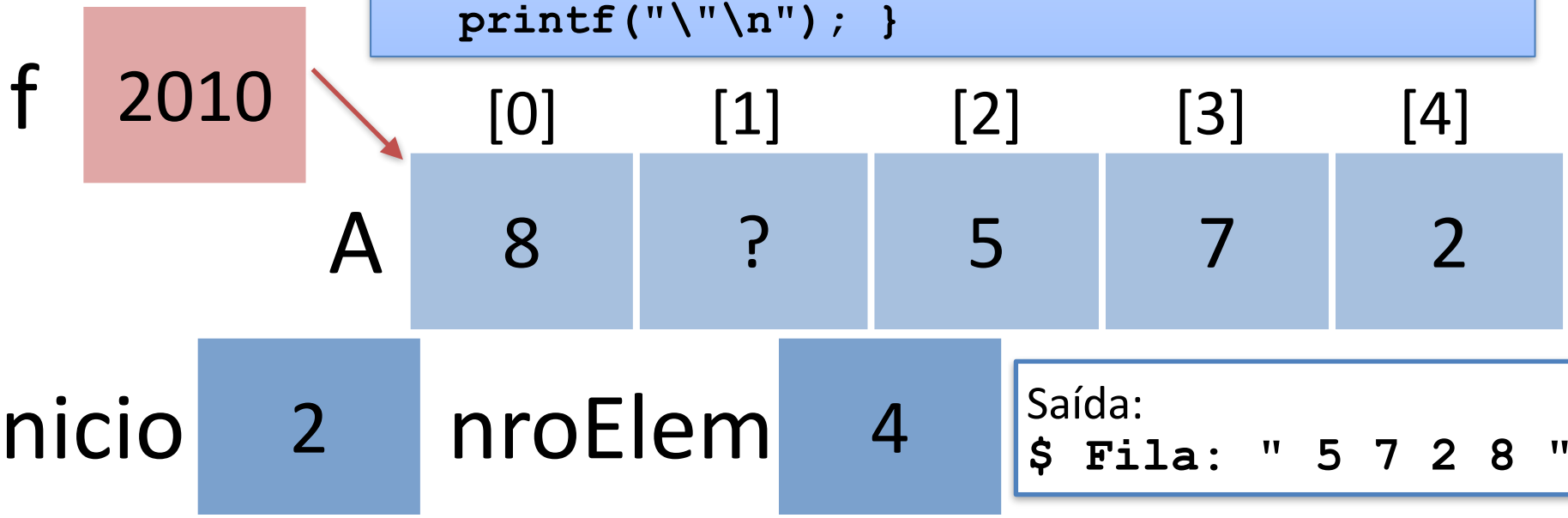


Exibição/Impressão

- Para exibir os elementos da estrutura precisaremos iterar pelos elementos válidos
- **Atenção:**
  - Há **`nroElem`** elementos válidos e o primeiro está na posição **`inicio`** do arranjo
  - Após o elemento da última posição do arranjo (posição **`MAX-1`**) está o elemento da posição **0** (trataremos o arranjo como se fosse circular)

Exibição/Impressão

```
void exibirFila(FILA* f) {
   printf("Fila: \" ");
   int i = f->inicio;
   int temp;
   for (temp=0;temp<f->nroElem;temp++) {
      printf("%i ", f->A[i].chave);
      i = (i + 1) % MAX; }
   printf("\"\n"); }
```



Saída:

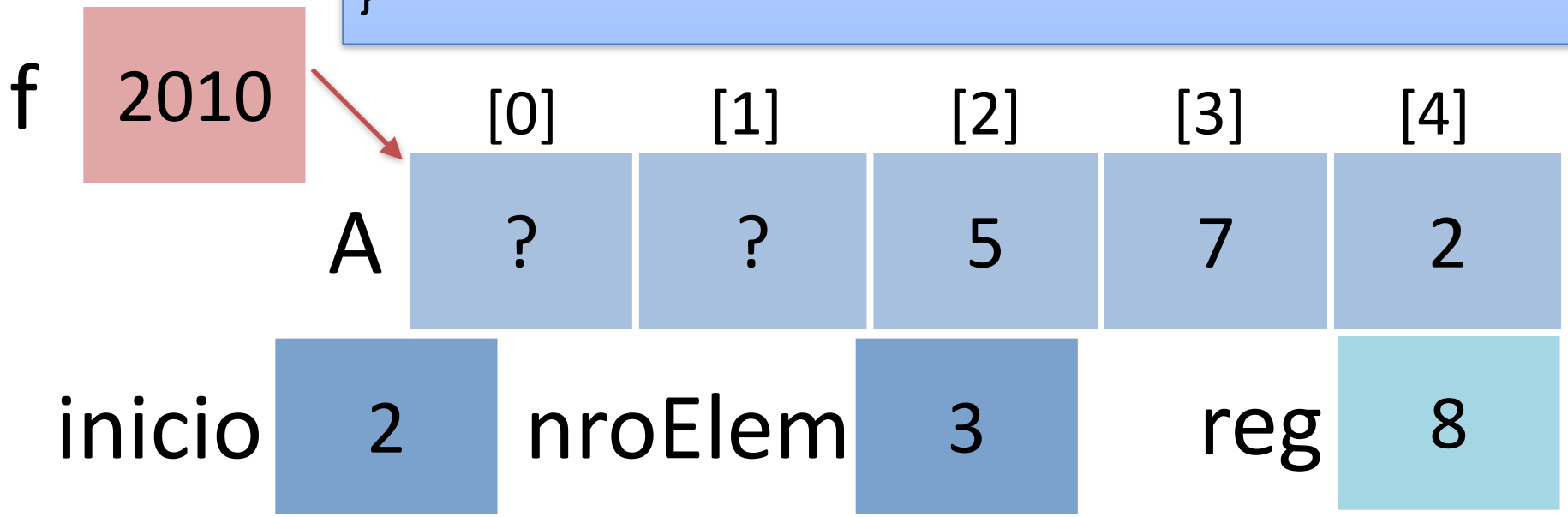
```
$ Fila: " 5 7 2 8 "
```

Inserção de um elemento (**`enQueue`**)

- O usuário passa como parâmetro um registro a ser inserido no final da fila
- Se a fila não estiver cheia, precisamos:
  - Identificar a posição no arranjo na qual o registro será inserido e inseri-lo lá
  - Alterar o valor campo **`nroElem`**

Inserção de um elemento (**enQueue**)

```
bool inserirElementoFila(FILA* f, REGISTRO reg)
{
  if (f->nroElem >= MAX) return false;
  int posicao = (f->inicio + f->nroElem) % MAX;
  f->A[posicao] = reg;
  f->nroElem++;
  return true;
}
```



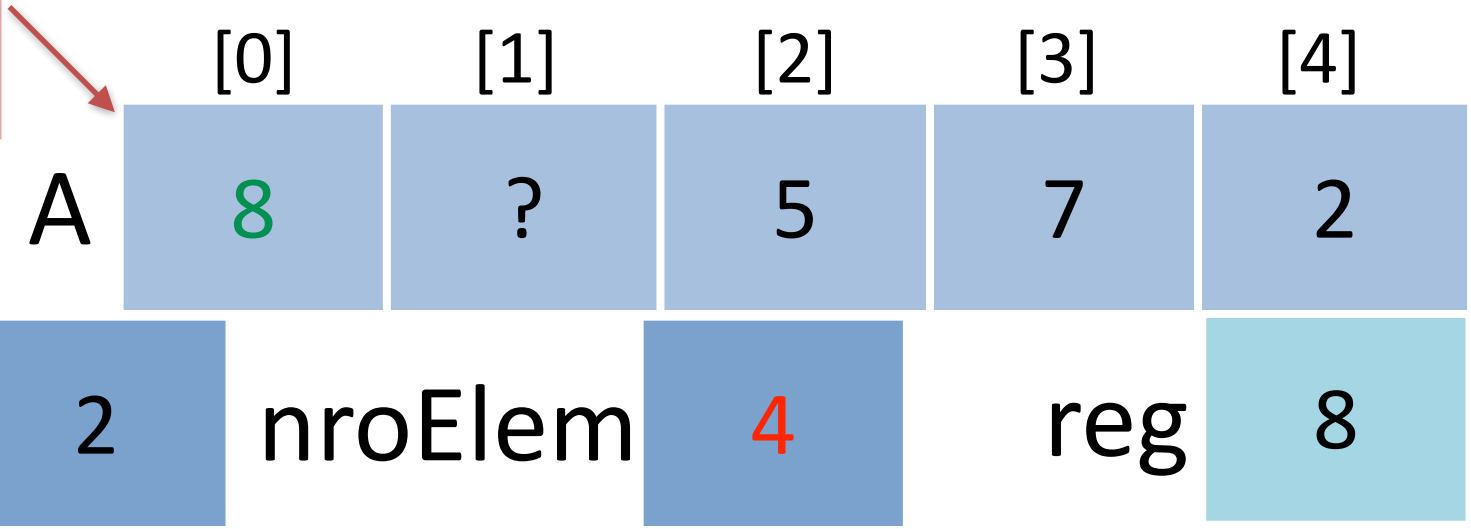
Inserção de um elemento (**enQueue**)

posição

0

```
bool inserirElementoFila(FILA* f, REGISTRO reg)
{
  if (f->nroElem >= MAX) return false;
  int posicao = (f->inicio + f->nroElem) % MAX;
  f->A[posicao] = reg;
  f->nroElem++;
  return true;
}
```

f 2010

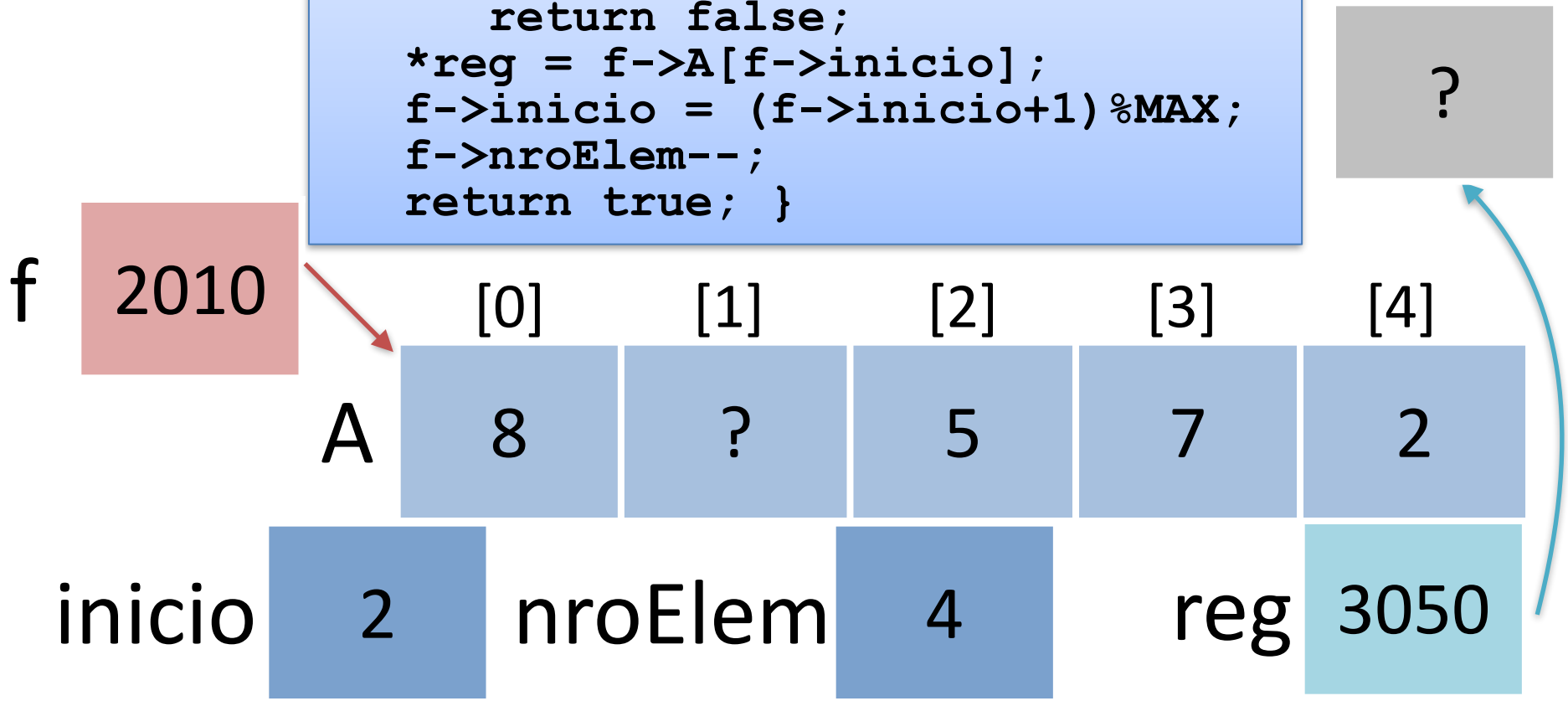


Exclusão de um elemento (**`deQueue`**)

- O usuário solicita a exclusão do elemento do início da fila:
- Se a fila não estiver vazia:
  - Iremos copiar esse elemento para um local indicado pelo usuário
  - Acertar o valor do campo **`nroElem`**
  - Acertar o valor do campo **`inicio`**

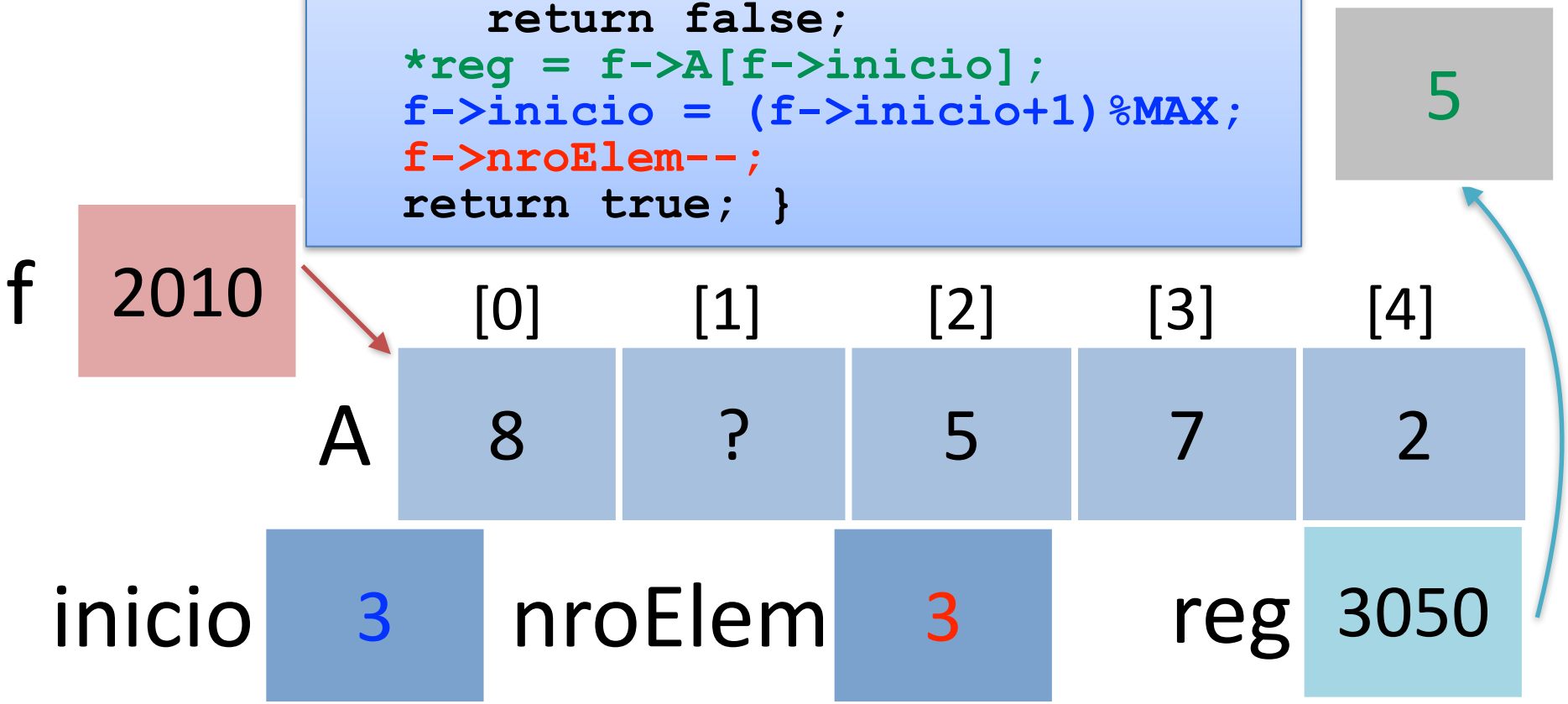
Exclusão de um elemento (**deQueue**)

```
bool excluirElementoFila(FILA* f,
REGISTRO* reg) {
   if (f->nroElem==0)
      return false;
   *reg = f->A[f->inicio];
   f->inicio = (f->inicio+1)%MAX;
   f->nroElem--;
   return true; }
```



Exclusão de um elemento (**deQueue**)

```
bool excluirElementoFila(FILA* f,
REGISTRO* reg) {
    if (f->nroElem==0)
        return false;
    *reg = f->A[f->inicio];
    f->inicio = (f->inicio+1)%MAX;
    f->nroElem--;
    return true; }
```

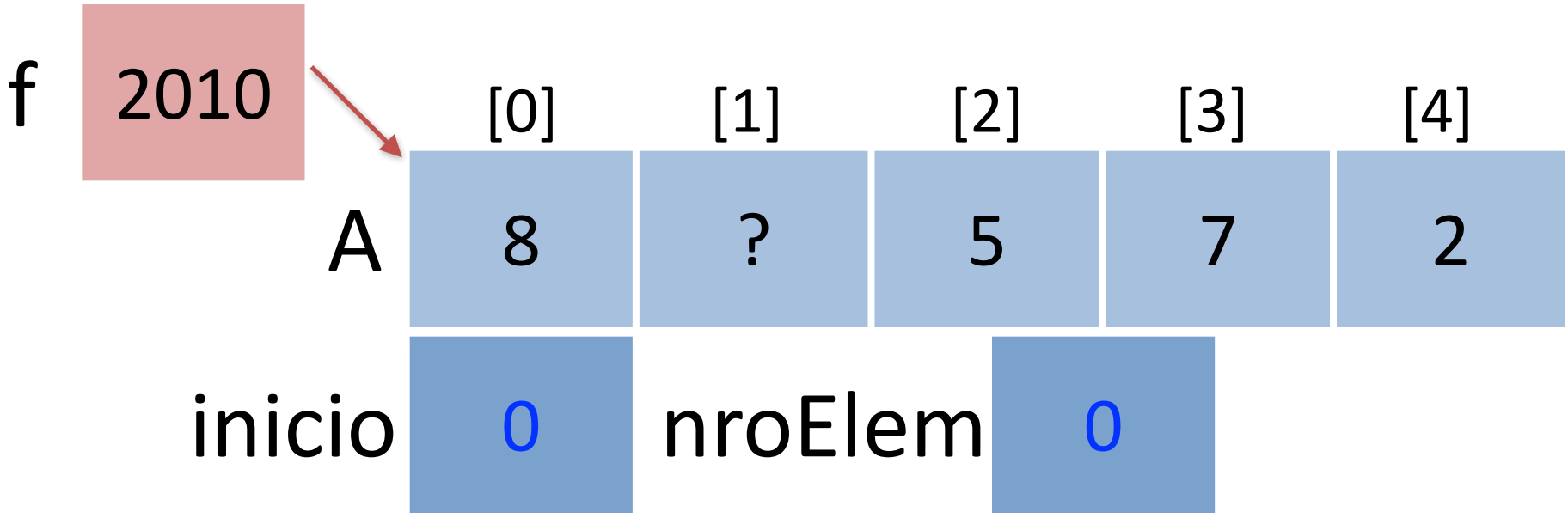


Reinicialização

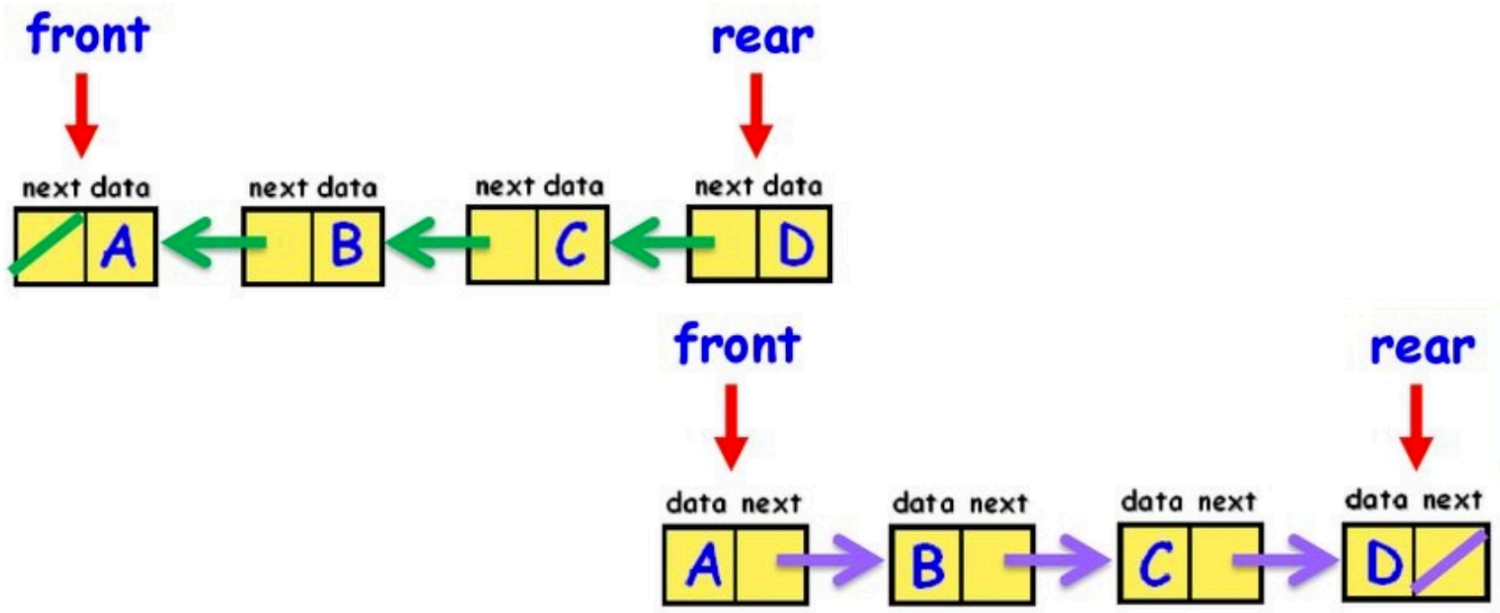
- Para reinicializar esta estrutura basta chamarmos a função de inicialização ou executarmos os mesmos comandos lá executados

Reinicialização

```
void reinicializarFila(FILA* f) {
   inicializarFila(f); }
void reinicializarFila2(FILA* f) {
   f->inicio=0;
   f->nroElem=0; }
```



Fila - Implementação dinâmica



Fila - Implementação dinâmica

- Alocaremos e desalocaremos a memória para os elementos sob demanda
- **Vantagem:** não precisamos gastar memória que não estamos usando
- Cada elemento indicará quem é seu sucessor (quem é o “próximo” na fila)
- Controlaremos os endereços dos elementos que estão no início e no fim da fila

Modelagem (fila dinâmica)

```
#include <stdio.h>
#include <stdlib.h>

#define true 1
#define false 0

typedef int bool;

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
 //outros campos...
} REGISTRO;
```

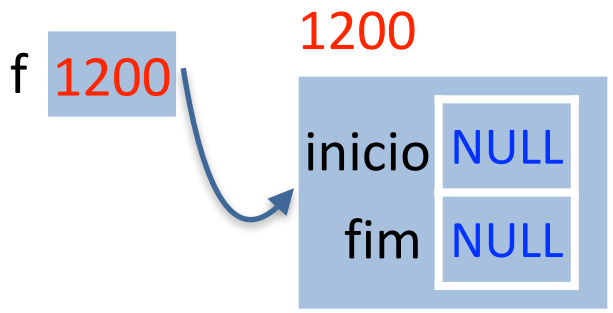
```
typedef struct aux {
  REGISTRO reg;
  struct aux* prox;
} ELEMENTO;

typedef ELEMENTO* PONT;

typedef struct {
  PONT inicio;
  PONT fim;
} FILA;
```

Inicialização

- Para inicializar uma fila (implementação dinâmica), precisamos:
  - Acertar o valor do campo **`inicio`** (para indicar que não há nenhum elemento válido)
  - Acertar o valor do campo **`fim`** (para indicar que não há nenhum elemento válido)
  - Nesta implementação não utilizaremos nó cabeça

Inicialização

```
void inicializarPilha(FILA* f) {
  f->inicio = NULL;
  f->fim = NULL;
}
```

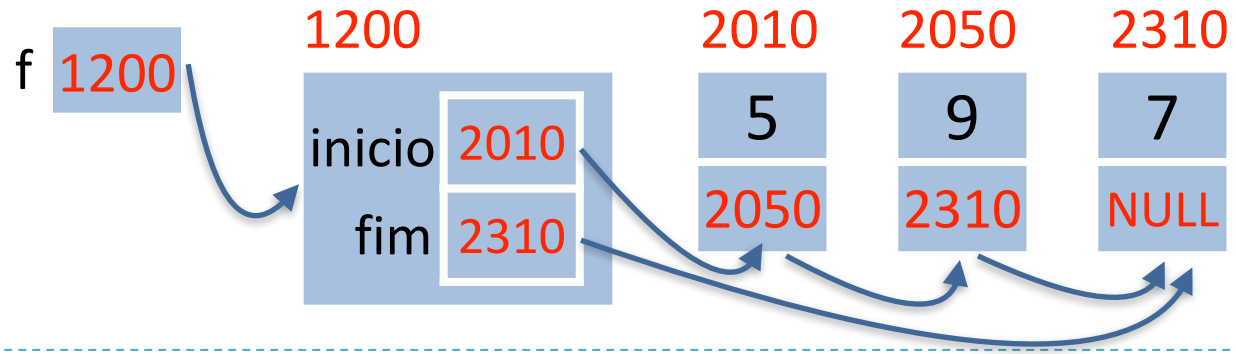


Retornar número de elementos

- Já que não temos um campo com o número de elementos na fila, precisaremos percorrer todos os elementos para contar quantos são

Retornar número de elementos

```
int tamanho(FILA* f) {
  PONT end = f->inicio;
  int tam = 0;
  while (end != NULL) {
    tam++;
    end = end->prox; }
  return tam; }
```

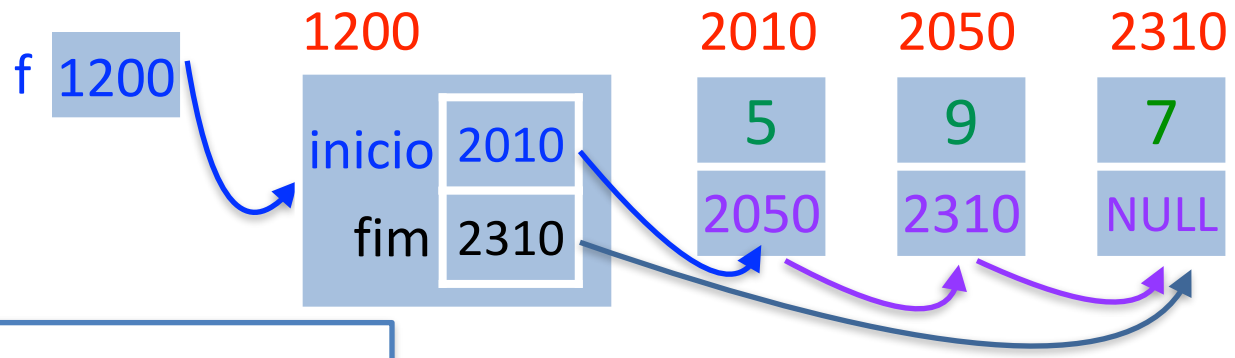


Exibição/Impressão

- Para exibir os elementos da estrutura precisaremos percorrer os elementos
- Começamos do início da fila até chegarmos no final

Verificar Exibição/Impressão

```
void exibirFila(FILA* f) {
  PONT end = f->inicio;
  printf("Fila: \" ");
  while (end != NULL) {
    printf("%i ", end->reg.chave);
    end = end->prox; }
  printf("\"\n"); }
```



Saída:

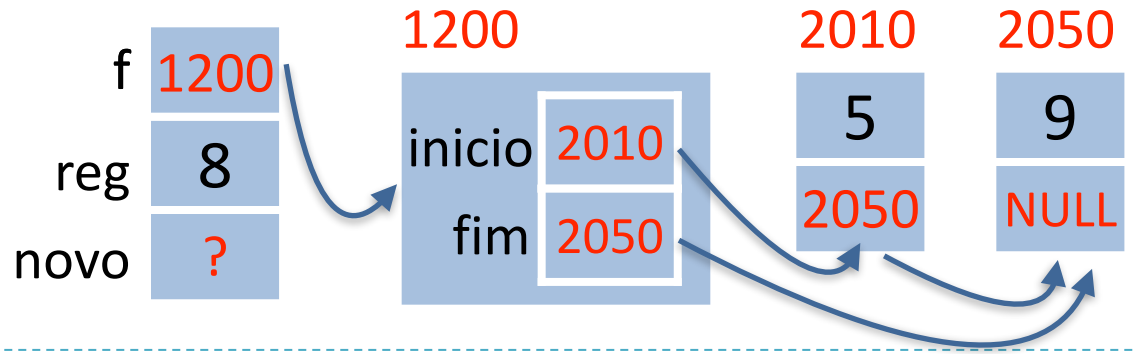
```
$ Fila: " 5 9 7 "
```

Inserção de um elemento (`enQueue`)

- O usuário passa como parâmetro um registro a ser inserido na fila, precisamos:
  - Alocar a memória para este novo elemento
  - Colocá-lo após o último elemento da fila
  - Alterar o valor do campo `fim`
  - **Atenção:** a fila poderia estar vazia

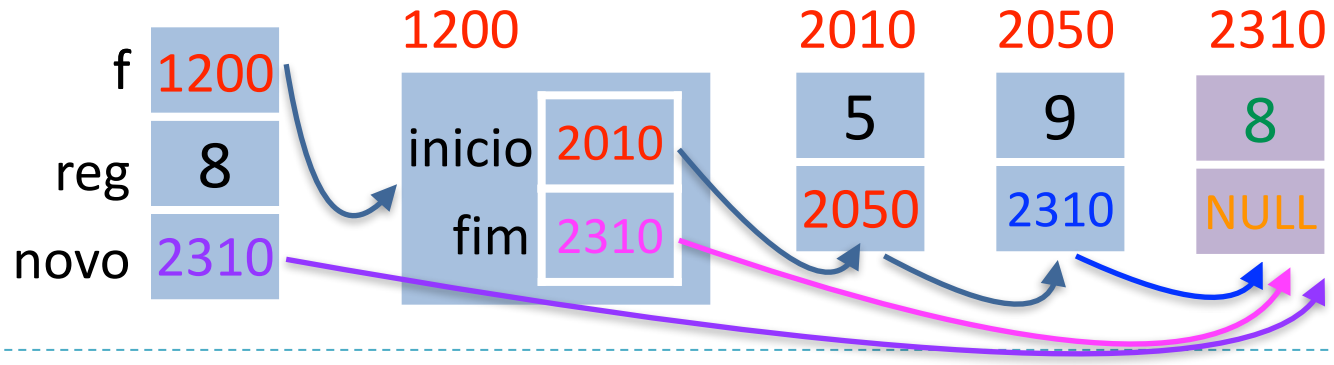
Inserção de um elemento (**enQueue**)

```
bool inserirNaFila(FILA* f,REGISTRO reg) {
   PONT novo = (PONT) malloc(sizeof(ELEMENTO));
   novo->reg = reg;
   novo->prox = NULL;
   if (f->inicio==NULL) f->inicio = novo;
   else f->fim->prox = novo;
   f->fim = novo;
   return true;
}
```



Inserção de um elemento (**enQueue**)

```
bool inserirNaFila(FILA* f,REGISTRO reg) {
   PONT novo = (PONT) malloc(sizeof(ELEMENTO));
   novo->reg = reg;
   novo->prox = NULL;
   if (f->inicio==NULL) f->inicio = novo;
   else f->fim->prox = novo;
   f->fim = novo;
   return true;
}
```

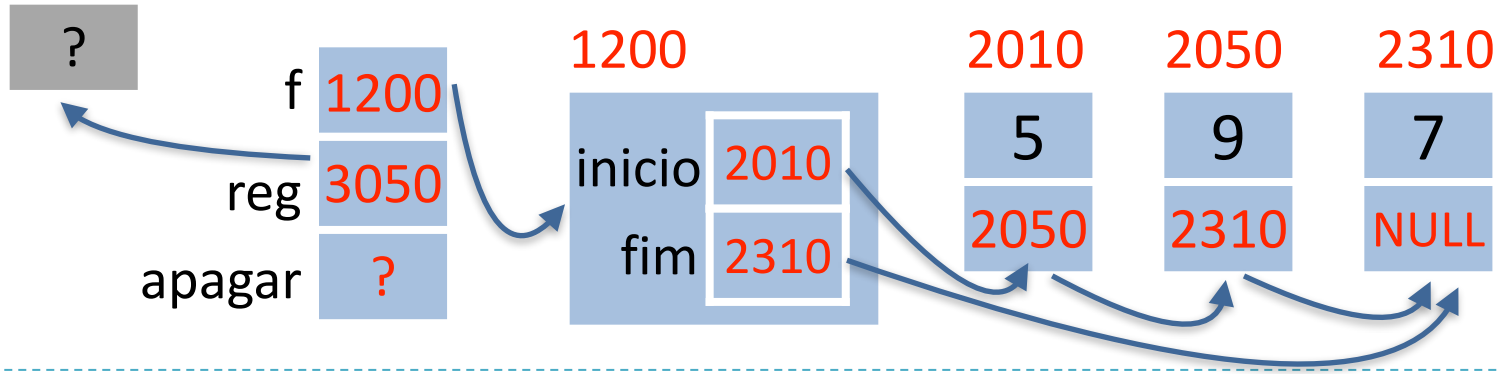


Exclusão de um elemento (`deQueue`)

- O usuário solicita a exclusão do elemento do início da fila. Se a fila não estiver vazia:
  - Iremos copiar esse elemento para um local indicado pelo usuário
  - Acertar o valor do campo **`inicio`**
  - Eventualmente acertar o valor do campo **`fim`**

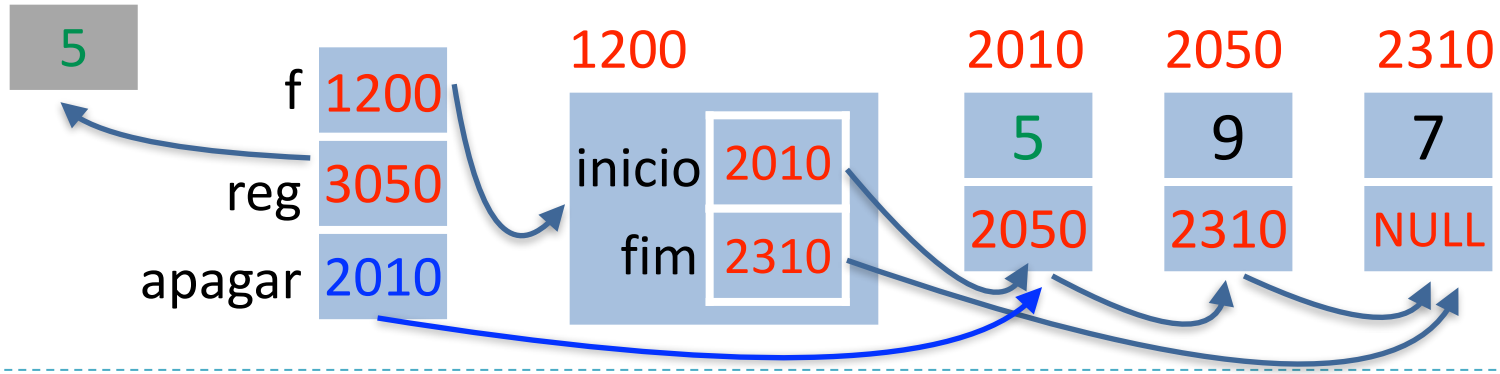
Exclusão de um elemento (**deQueue**)

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;
  *reg = f->inicio->reg;
  PONT apagar = f->inicio;
  f->inicio = f->inicio->prox;
  free(apagar);
  if (f->inicio == NULL) f->fim = NULL;
  return true; }
```



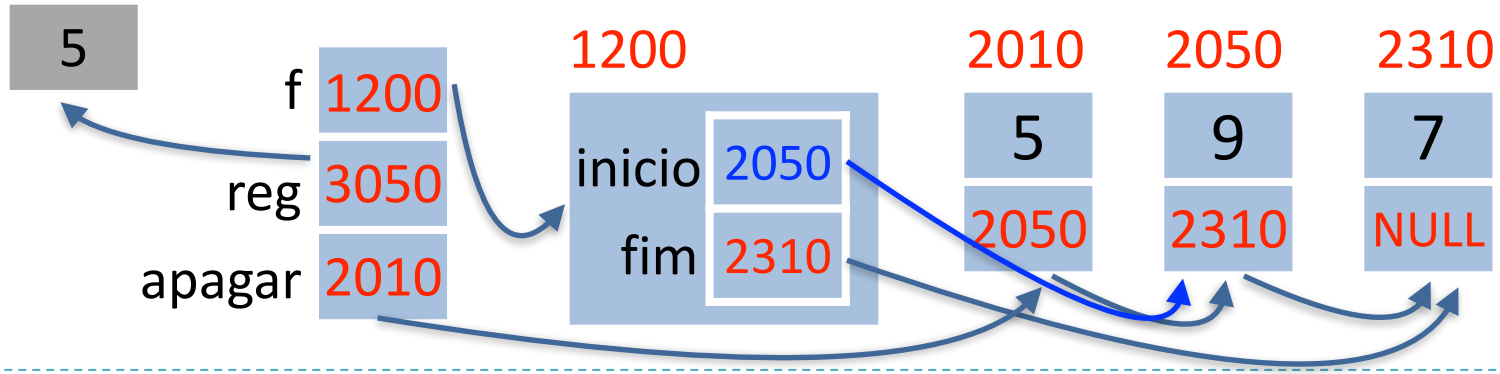
Exclusão de um elemento (**deQueue**)

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;
  *reg = f->inicio->reg;
  PONT apagar = f->inicio;
  f->inicio = f->inicio->prox;
  free(apagar);
  if (f->inicio == NULL) f->fim = NULL;
  return true; }
```



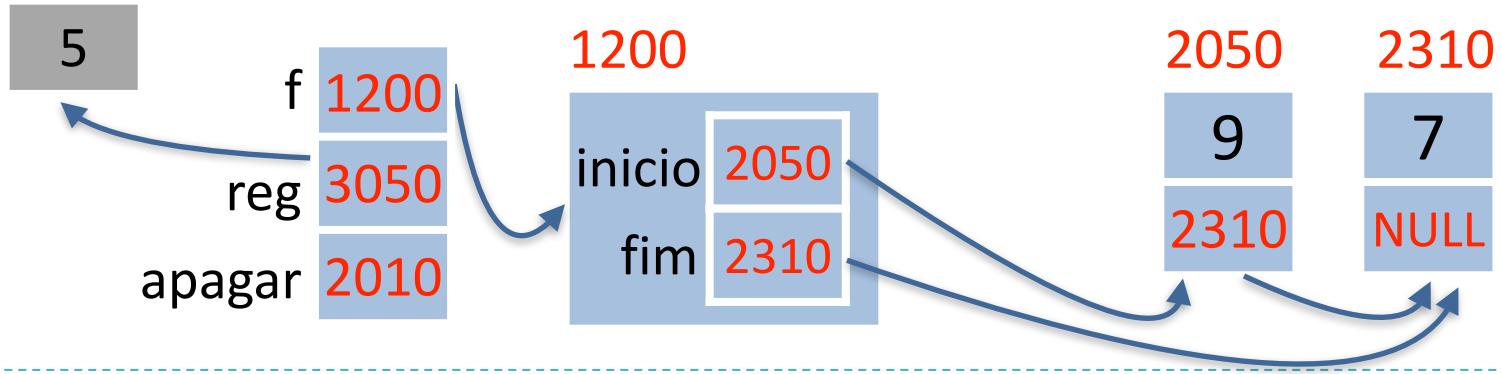
Exclusão de um elemento (**deQueue**)

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;
  *reg = f->inicio->reg;
  PONT apagar = f->inicio;
  f->inicio = f->inicio->prox;
  free(apagar);
  if (f->inicio == NULL) f->fim = NULL;
  return true; }
```



Exclusão de um elemento (**deQueue**)

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;
  *reg = f->inicio->reg;
  PONT apagar = f->inicio;
  f->inicio = f->inicio->prox;
  free(apagar);
  if (f->inicio == NULL) f->fim = NULL;
  return true; }
```



Reinicialização

- Para reinicializar a fila, precisamos excluir todos os seus elementos e colocar **`NULL`** nos campos **`inicio`** e **`fim`**

Reinicialização

```
void reinicializarFila(FILA* f) {
  PONT end = f->inicio;
  while (end != NULL) {
    PONT apagar = end;
    end = end->prox;
    free(apagar);
  }
  f->inicio = NULL;
  f->fim = NULL;
}
```