AULA 12
ESTRUTURA DE DADOS

Fila (implementação dinâmica)

Norton T. Roman & Luciano A. Digiampietri

Fila

É uma estrutura linear na qual:

- As inserções ocorrem no final da fila;
- As exclusões ocorrem no início da fila.
- Utiliza a mesma lógica de uma fila de pessoas.

Fila - implementação dinâmica

Fila - implementação dinâmica

- Alocaremos e desalocaremos a memória para os elementos **sob demanda**;

Fila - implementação dinâmica

- Alocaremos e desalocaremos a memória para os elementos sob demanda;
- Cada elemento indicará quem é seu sucessor (quem é o “próximo” na fila);

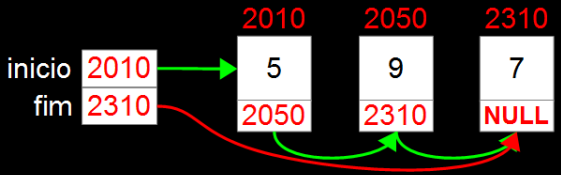
Fila - implementação dinâmica

- Alocaremos e desalocaremos a memória para os elementos sob demanda;
- Cada elemento indicará quem é seu sucessor (quem é o “próximo” na fila);
- Controlaremos os endereços dos elemento que estão no início e no fim da fila.

Fila (ideia geral)

Temos dois campos para indicar os endereços de quem está no início no fim da fila

Cada elemento aponta para seu sucessor

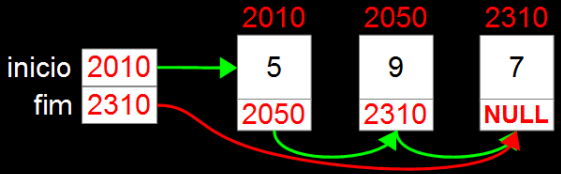


Fila (ideia geral)

Temos dois campos para indicar os endereços de quem está no início no fim da fila

Cada elemento aponta para seu sucessor

Como inserimos o elemento 8?

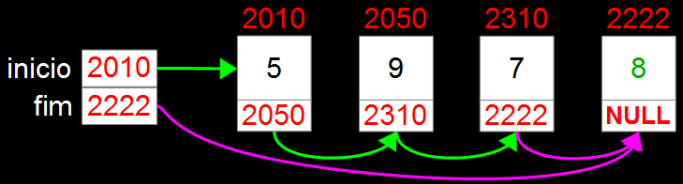


Fila (ideia geral)

Temos dois campos para indicar os endereços de quem está no início no fim da fila

Cada elemento aponta para seu sucessor

Como inserimos o elemento 8?



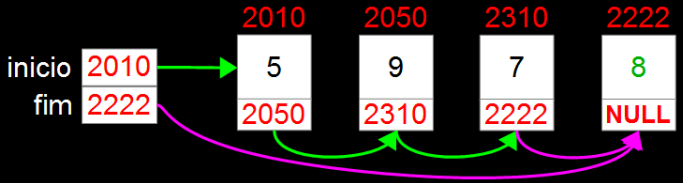
Fila (ideia geral)

Temos dois campos para indicar os endereços de quem está no **início** no **fim** da fila

Cada elemento aponta para seu **sucessor**

Como inserimos o **elemento 8?**

Como **excluímos um elemento?**



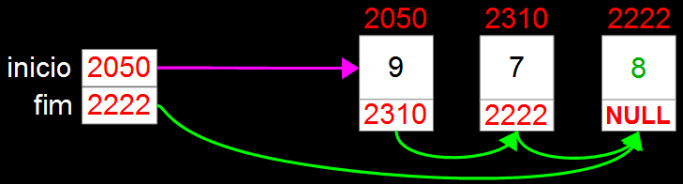
Fila (ideia geral)

Temos dois campos para indicar os endereços de quem está no **início** no **fim** da fila

Cada elemento aponta para seu **sucessor**

Como inserimos o **elemento 8?**

Como **excluímos um elemento?**



Modelagem

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

typedef int TIPOCHAVE;

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

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

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

Modelagem

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

typedef int TIPOCHAVE;

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

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

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

Modelagem

```c
#include <stdio.h>
#include <malloc.h>

typedef int TIPOCHAVE;

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

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

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

Modelagem

```c
#include <stdio.h>
#include <malloc.h>

typedef int TIPOCHAVE;

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

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

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

Modelagem

```c
#include <stdio.h>
#include <malloc.h>

typedef int TIPOCHAVE;

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

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

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

Funções de gerenciamento

Implementaremos funções para:

- Inicializar a estrutura
- Retornar a quantidade de elementos válidos
- Exibir os elementos da estrutura
- Inserir elementos na estrutura (no fim)
- Excluir elementos da estrutura (no início)
- Reinicializar a estrutura

Inicialização

Para inicializarmos nossa fila (implementação dinâmica), precisamos:

Inicialização

Para inicializarmos nossa 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);

Inicialização

Para inicializarmos nossa 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 inicializarFila(FILA* f) {
  f->inicio = NULL;
  f->fim = NULL;
}
```

Inicialização

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

Retornar número de elementos

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) {




}
```

Retornar número de elementos

```
int tamanho(FILA* f) {
  PONT end = f->inicio;
  int tam = 0;




}
```

Retornar número de elementos

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

}
```

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.

Exibição/Impressão

Para exibir os elementos da estrutura precisaremos percorrer os elementos.

- **Começamos do início da fila até chegarmos no final**

Exibição/Impressão

```
void exibirFila(FILA* f) {




}
```

Exibição/Impressão

```
void exibirFila(FILA* f) {
  PONT end = f->inicio;




}
```

Exibição/Impressão

```
void exibirFila(FILA* f) {
  PONT end = f->inicio;
  printf("Fila: \" ");




  printf("\"\n");
}
```

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");
}
```

Inserção de um elemento

O usuário passa como parâmetro um registro a ser inserido no final da fila, precisamos:

Inserção de um elemento

O usuário passa como parâmetro um registro a ser inserido no final da fila, precisamos:

- **Alocar a memória para este novo elemento;**

Inserção de um elemento

O usuário passa como parâmetro um registro a ser inserido no final da fila, precisamos:

- **Alocar a memória para este novo elemento;**
- **Colocá-lo após o último elemento da fila;**

Inserção de um elemento

O usuário passa como parâmetro um registro a ser inserido no final da fila, precisamos:

- **Alocar a memória para este novo elemento;**
- **Colocá-lo após o último elemento da fila;**
- **Alterar o valor do campo *fim*.**

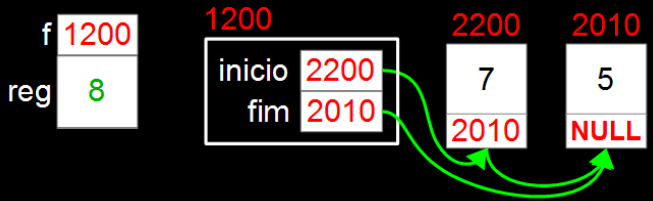
Inserção de um elemento

O usuário passa como parâmetro um registro a ser inserido no final da 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.**

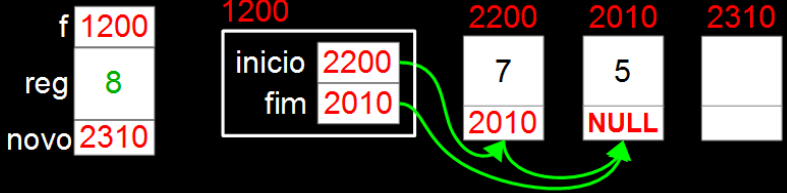
Inserção de um elemento

```
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;
}
```



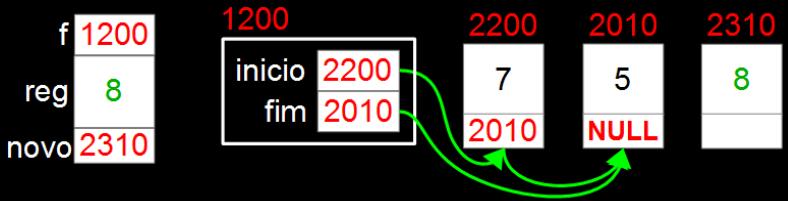
Inserção de um elemento

```
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;
}
```



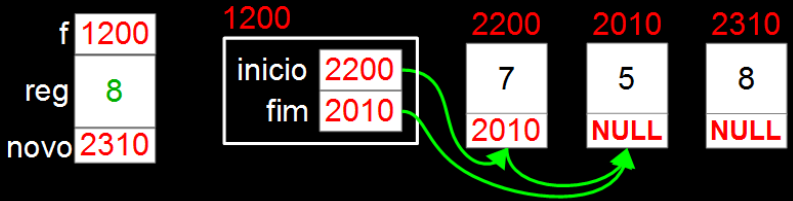
Inserção de um elemento

```
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;
}
```



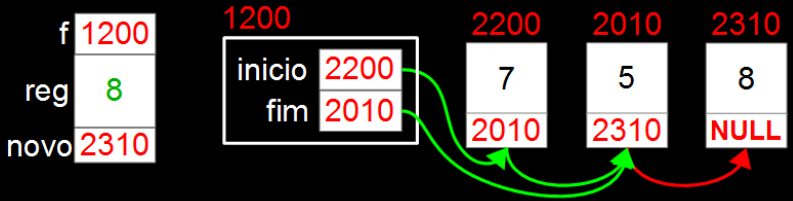
Inserção de um elemento

```
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;
}
```



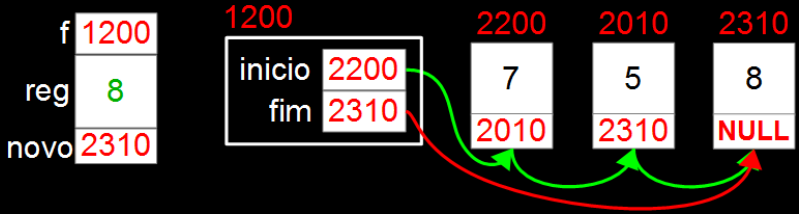
Inserção de um elemento

```
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;
}
```



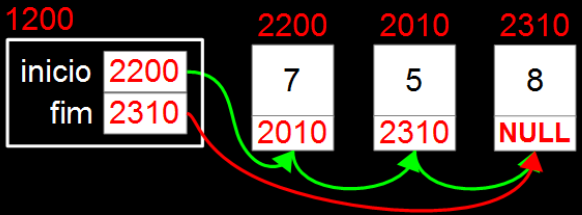
Inserção de um elemento

```
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;
}
```



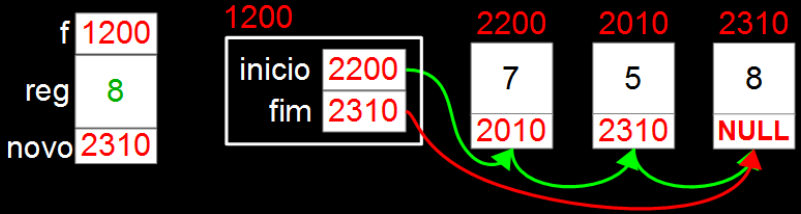
Inserção de um elemento

```
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;
}
```



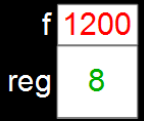
Inserção de um elemento

```
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;
}
```



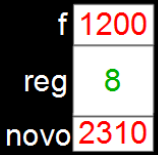
Inserção de um elemento

```
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

```
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

O usuário solicita a exclusão do elemento do início da fila. Se a fila não estiver vazia:

Exclusão de um elemento

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

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {




}
```

Exclusão de um elemento

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;









}
```

Exclusão de um elemento

```
bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;
  *reg = f->inicio->reg;




}
```

Exclusão de um elemento

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




}
```

Exclusão de um elemento

```
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;



}
```

Exclusão de um elemento

```
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);


}
```

Exclusão de um elemento

```
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;

}
```

Exclusão de um elemento

```
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 da fila dinâmica

Reinicialização da fila dinâmica

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

Reinicialização da fila dinâmica

```
void reinicializarFila(FILA* f) {









}
```

Reinicialização da fila dinâmica

```
void reinicializarFila(FILA* f) {
  PONT end = f->inicio;




}
```

Reinicialização da fila dinâmica

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


}
```

Reinicialização da fila dinâmica

```
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;
}
```

AULA 12
ESTRUTURA DE DADOS

Fila (implementação dinâmica)

Norton T. Roman & Luciano A. Digiampietri