

# TAD Fila

Prof. André Grégio

# Tipos Abstratos de Dados

Relembrando...

- **Representação** de itens/objetos/elementos
- Possui **atributos** que abstraem as características dos itens representados
- Define **operações** que podem ser feitas sobre os itens

# Tipos Abstratos de Dados

Operações comuns:

1. Inicializar TAD
2. Verificar se TAD está vazio
3. Criar elemento
4. Inserir elemento
5. Remover elemento
6. Buscar elemento

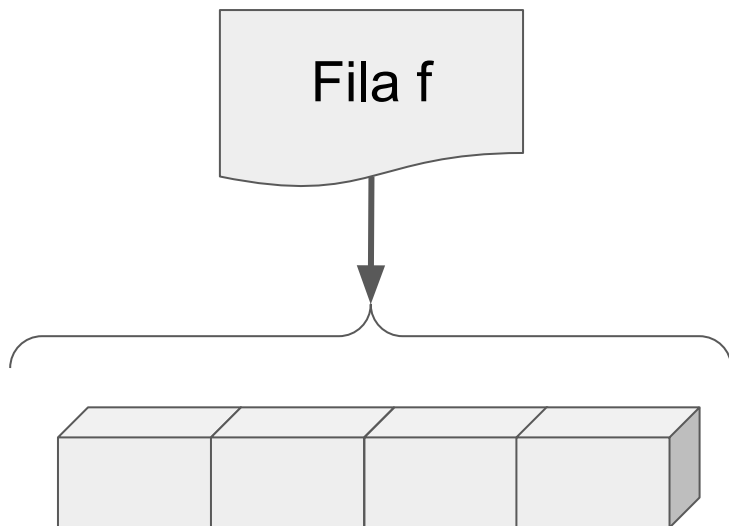
# Fila

A fila é um tipo abstrato de dados especial que representa um conjunto de objetos aos quais se tem acesso aos elementos do INÍCIO e do FIM



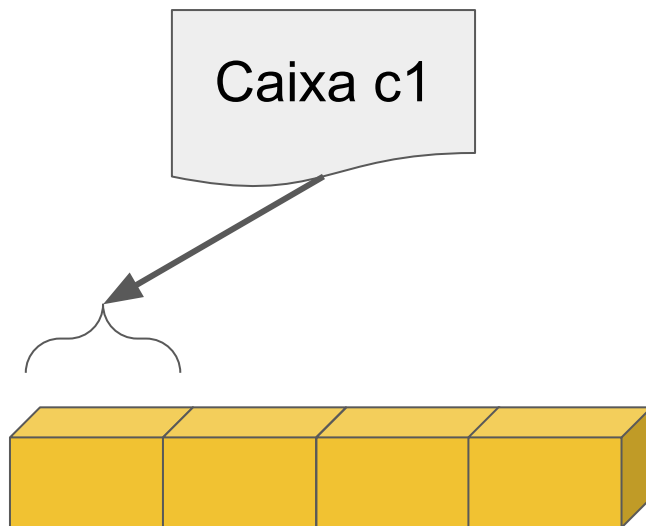
# Fila

Uma fila de exemplo com 4 “caixas”



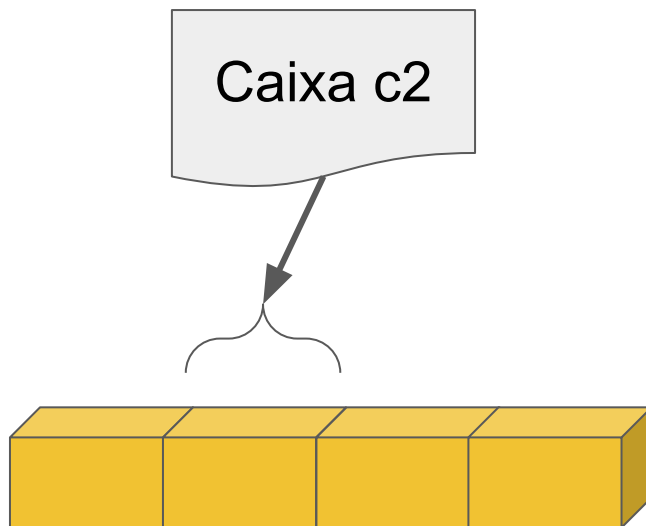
# Fila

Uma fila de exemplo com 4 “caixas”



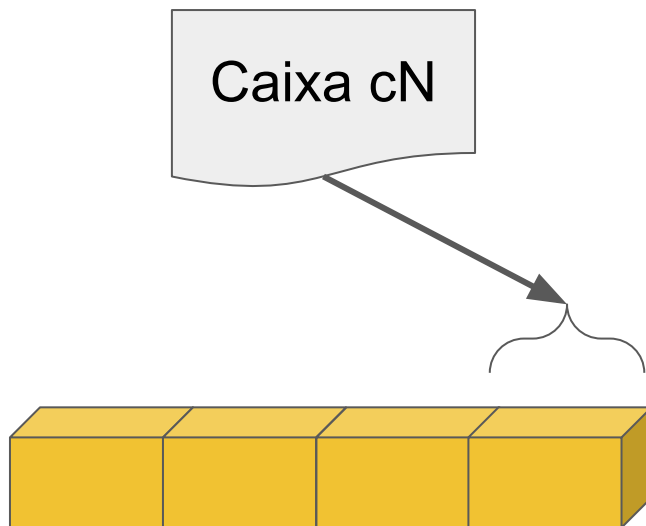
# Fila

Uma fila de exemplo com 4 “caixas”



# Fila

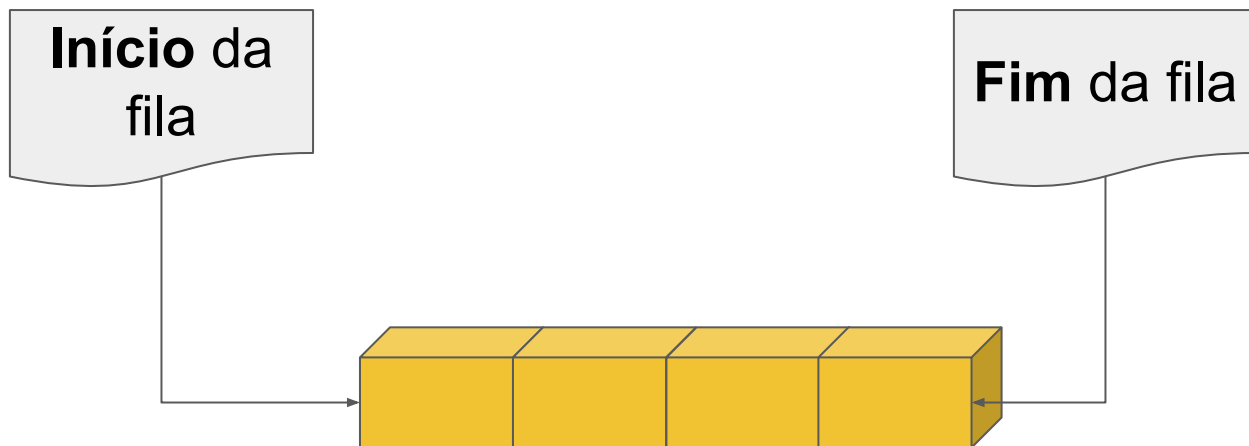
Uma fila de exemplo com 4 “caixas”





# Fila

Uma fila de exemplo com 4 “caixas”



# Estrutura da fila

Uma fila pode possuir os seguintes atributos:

- Início, para marcação do primeiro elemento “acessível”
- Fim, para marcação do último elemento inserido
  - Após o fim está a posição do próximo elemento a ser inserido
- Tamanho, para indicar quantos elementos estão na fila
- Comprimento, para demarcar quantos elementos a fila suporta
- Espaço de armazenamento, para guardar os elementos

# Estrutura da fila

Uma fila pode possuir os seguintes atributos:

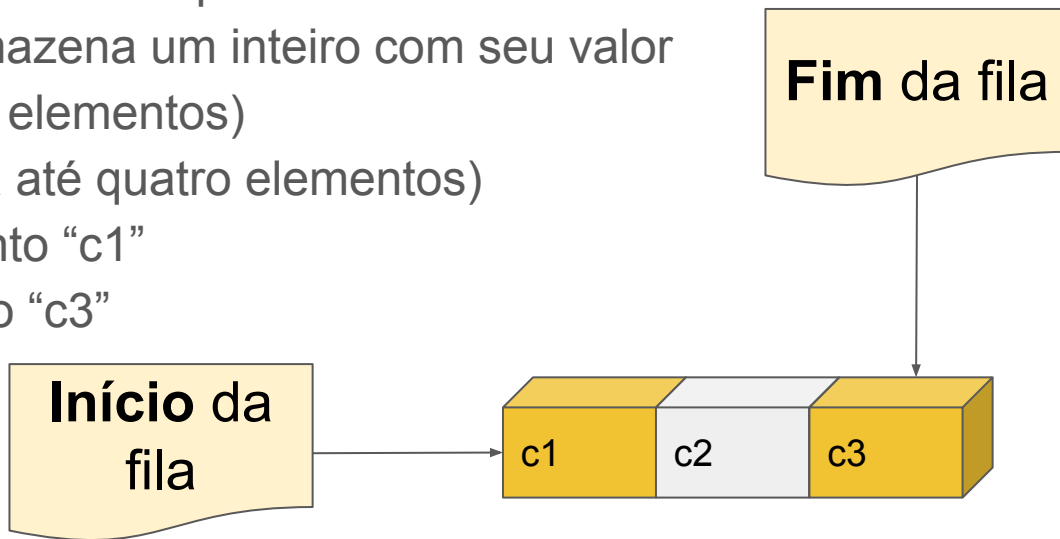
- Início, para marcação do primeiro elemento “acessível”
- Fim, para marcação do último elemento inserido
  - Após o fim está a posição do próximo elemento a ser inserido
- Tamanho, para indicar quantos elementos estão na fila
- Comprimento, para demarcar quantos elementos a fila suporta
- Espaço de armazenamento, para guardar os elementos

**Nem todos os atributos listados acima são necessários!**

# Estrutura da fila

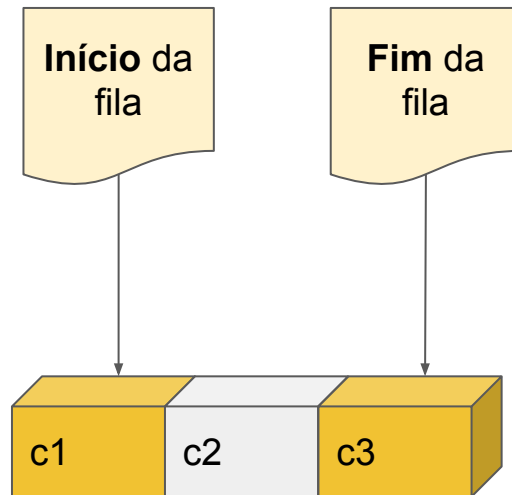
Suponha a seguinte fila:

- Já inicializada com elementos do tipo “caixa”
  - Elemento “caixa” armazena um inteiro com seu valor
- Tamanho = 3 (possui três elementos)
- Comprimento = 4 (guarda até quatro elementos)
- Início aponta para elemento “c1”
- Fim aponta para elemento “c3”



# Operações sobre a fila

- Inserir elemento
  - Cria um novo elemento do tipo “caixa”

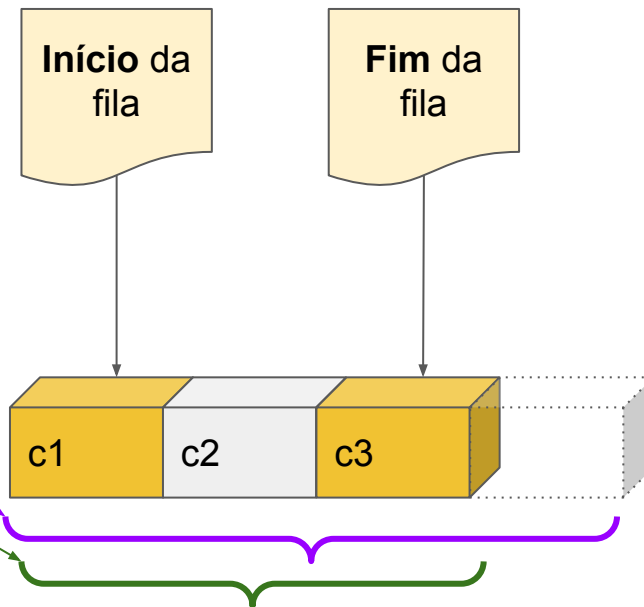


# Operações sobre a fila

- Inserir elemento
  - Cria um novo elemento do tipo “caixa”



- Verifica se há espaço na fila
  - Comprimento = 4
  - Tamanho = 3

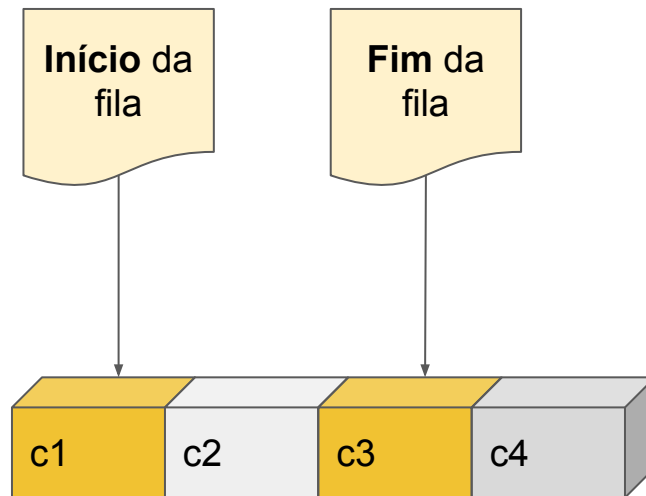


# Operações sobre a fila

- Inserir elemento
  - Cria um novo elemento do tipo “caixa”



- Verifica se há espaço na fila
  - Comprimento = 4
  - Tamanho = 3
- Insere elemento no fim da fila

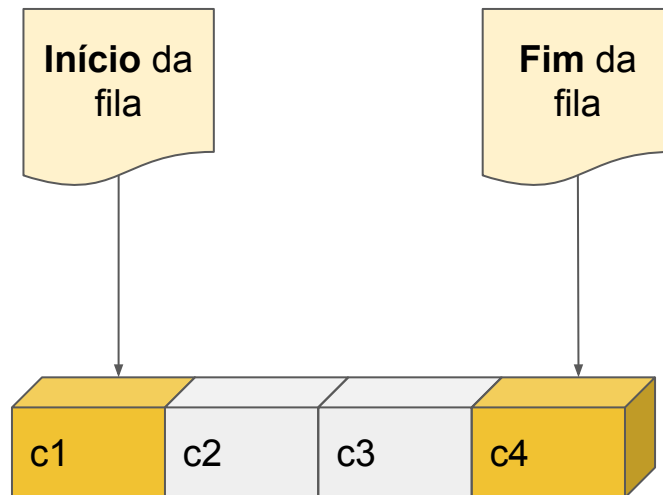


# Operações sobre a fila

- Inserir elemento
  - Cria um novo elemento do tipo “caixa”



- Verifica se há espaço na fila
  - Comprimento = 4
  - Tamanho = 3
- Insere elemento no fim da fila
- Atualiza fim da fila





# Operações sobre a fila

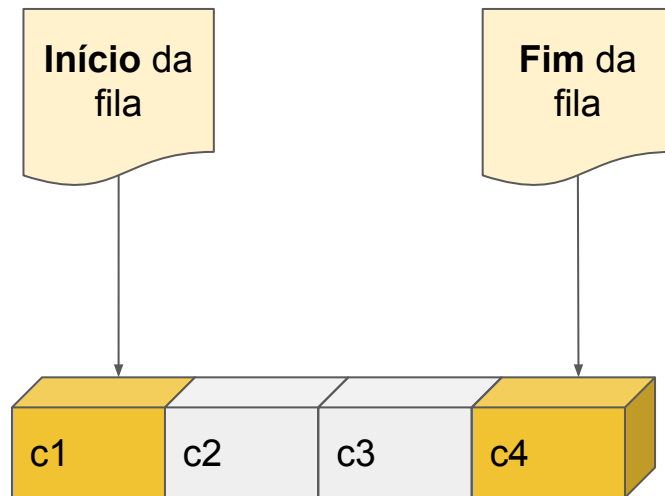
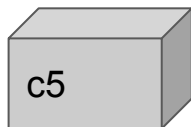
- A operação de **inserir elemento** em uma fila é chamada de **ENQUEUE**
- Argumentos de Entrada:
  - Uma fila *f*
  - Um elemento *x* a ser inserido no fim da fila

Protótipo da função ENQUEUE:

- Enqueue(fila *f*, elemento *x*)

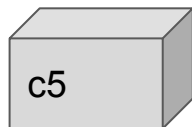
# Operações sobre a fila

- Verificação de *overflow*
- Inserir elemento “c5”:
  - Cria um novo elemento do tipo “caixa”

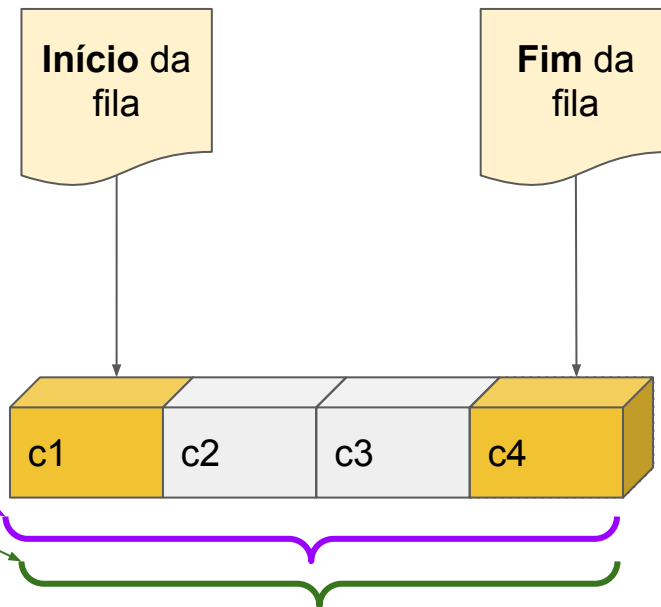


# Operações sobre a fila

- Verificação de *overflow*
- Inserir elemento “c5”:
  - Cria um novo elemento do tipo “caixa”



- Verifica se há espaço na fila
  - Comprimento = 4
  - Tamanho = 4

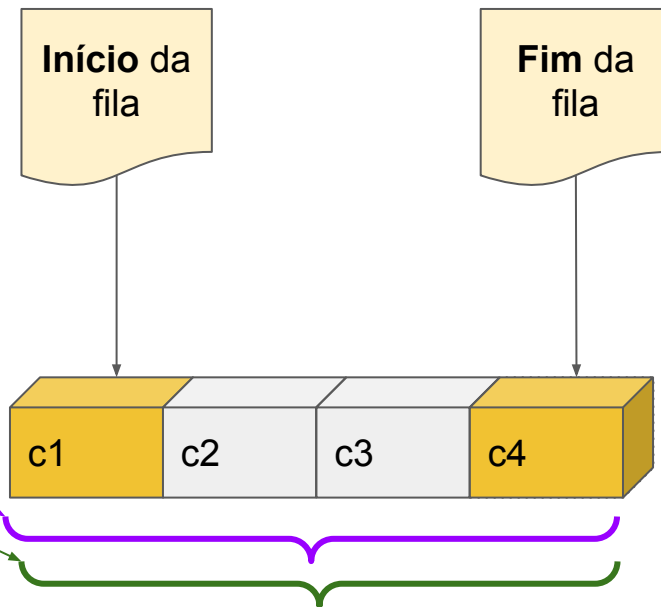


# Operações sobre a fila

- Verificação de *overflow*
- Inserir elemento “c5”:
  - Cria um novo elemento do tipo “caixa”



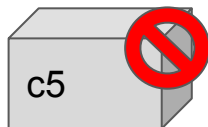
- Verifica se há espaço na fila
  - Comprimento = 4
  - Tamanho = 4



- Destrói novo elemento e avisa usuário

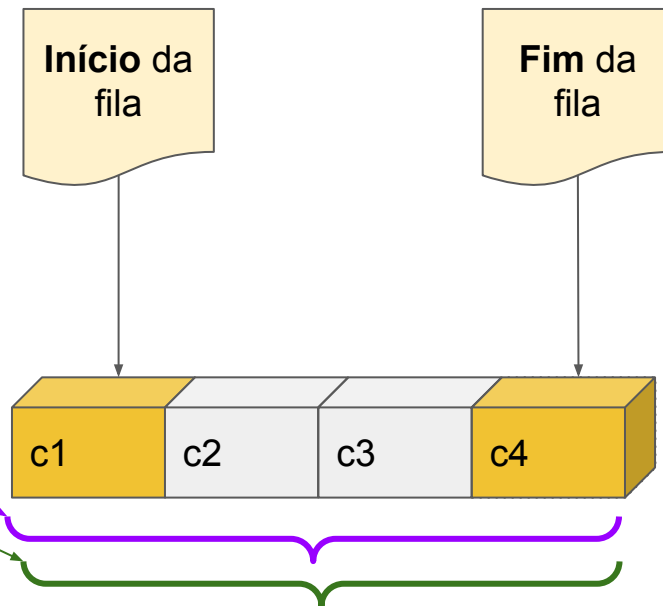
# Operações sobre a fila

- Verificação de *overflow*
- Inserir elemento “c5”:
  - Cria um novo elemento do tipo “caixa”



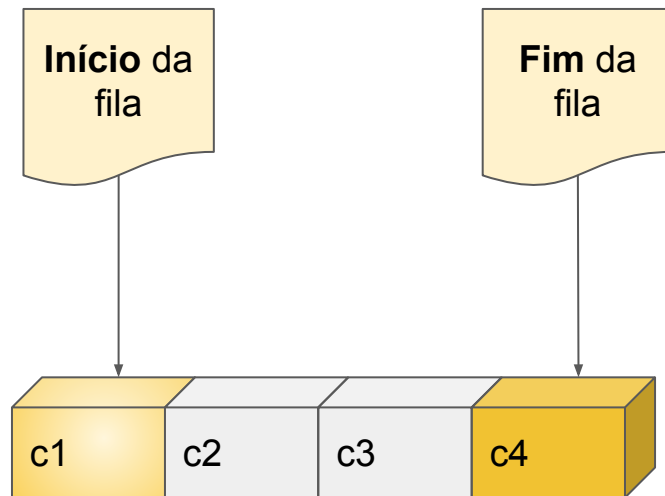
- Verifica se há espaço na fila
  - Comprimento = 4
  - Tamanho = 4

- Destrói novo elemento e avisa usuário



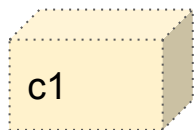
# Operações sobre a fila

- Remover elemento
  - Obtém o elemento do início

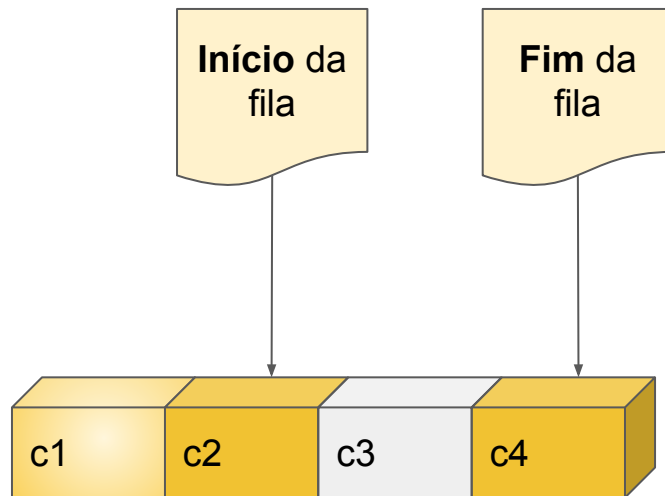


# Operações sobre a fila

- Remover elemento
  - Obtém o elemento do início



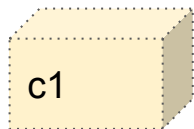
- Atualiza início



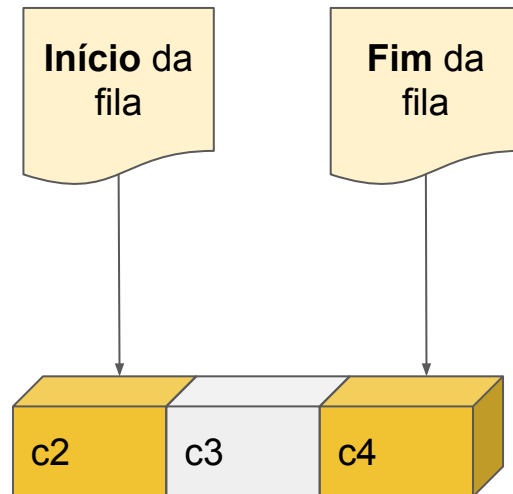
# Operações sobre a fila

- Remover elemento

- Obtém o elemento do início



- Atualiza início
- Libera memória (remove elemento do início)

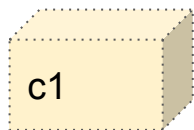




# Operações sobre a fila

- Remover elemento

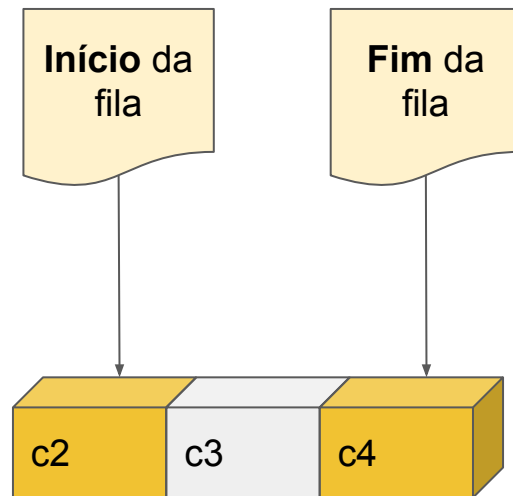
- Obtém o elemento do início



- Atualiza início
- Libera memória (remove elemento do início)

ou

- Devolve elemento para quem chamou a função



# Operações sobre a fila

- A operação de **remover elemento** da fila é chamada de **DEQUEUE**
- Argumentos de Entrada:
  - Uma fila f

Protótipo da função DEQUEUE:

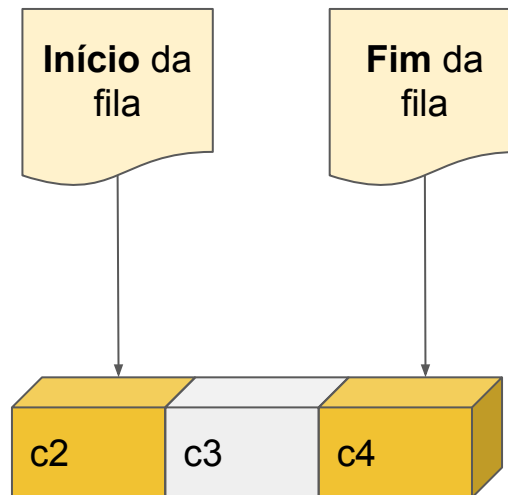
- Dequeue(fila f)

# Operações sobre a fila

- A operação de **remover elemento** da fila (**DEQUEUE**):
  - Não permite escolha do elemento, pois sempre remove o início

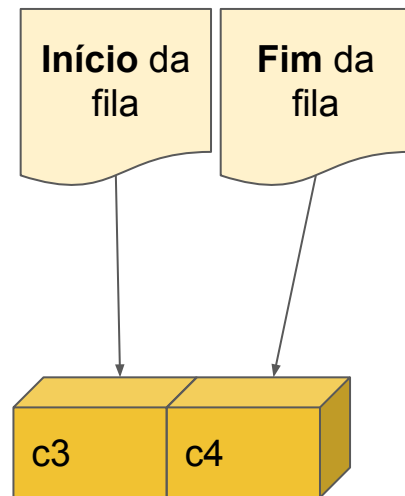
# Operações sobre a fila

- A operação de **remover elemento** da fila (**DEQUEUE**):
  - Não permite escolha do elemento, pois sempre remove o início
- Dequeue(f)



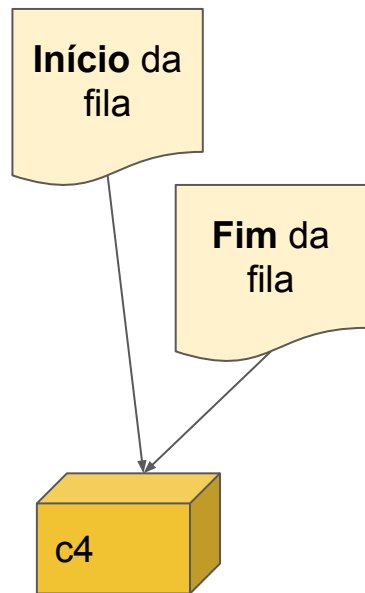
# Operações sobre a fila

- A operação de **remover elemento** da fila (**DEQUEUE**):
  - Não permite escolha do elemento, pois sempre remove o início
- Dequeue(f)
- Dequeue(f)



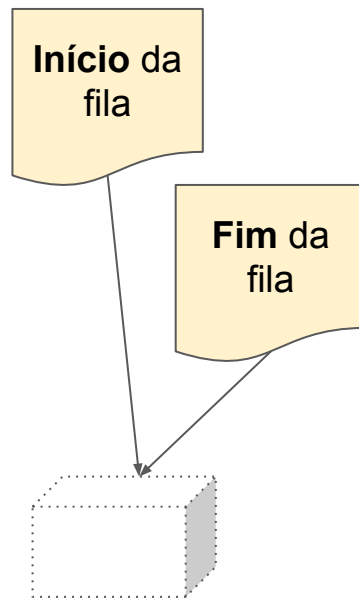
# Operações sobre a fila

- A operação de **remover elemento** da fila (**DEQUEUE**):
  - Não permite escolha do elemento, pois sempre remove o início
- Dequeue(f)
- Dequeue(f)
- Dequeue(f)



# Operações sobre a fila

- A operação de **remover elemento** da fila (**DEQUEUE**):
  - Não permite escolha do elemento, pois sempre remove o início
- Dequeue(f)
- Dequeue(f)
- Dequeue(f)
- **Dequeue(f)**
- Tomar cuidado com *UNDERFLOW*!
  - Verificar se fila está vazia...



# Política da Fila

As filas obedecem a uma política chamada FIFO (*First In, First Out*), ou seja:

- O primeiro elemento inserido é o primeiro a sair
- Insere no fim da fila
- Remove do início da fila



# Exemplo de implementação do TAD fila

Estrutura “item” com atributos “valor” e “próximo item”:

```
struct item {  
    int valor;  
    struct item *prox;  
};
```

```
struct item *c1;  
c1 = malloc(sizeof(struct item));  
c1->valor = 1;
```

# Exemplo de implementação do TAD fila

Estrutura “fila” com atributos “início” e “fim”

```
struct fila {  
    struct item *inicio;  
    struct item *fim;  
};
```

```
struct fila *f1;
```

# Exemplo de implementação do TAD fila

Operações sobre o TAD fila:

- Inicializar

```
struct fila *inicializaFila(){
    struct fila *f1;
    f1 = malloc(sizeof(struct fila));
    f1->inicio = NULL;
    f1->fim = NULL;
    return f1;
}
```

```
struct fila *f1 = inicializaFila();
```

# Exemplo de implementação do TAD fila

Operações sobre o TAD fila:

- Verificar se está vazia:

```
int filaVazia(struct fila *f1){  
    if (f1->inicio != NULL && f1->fim != NULL)  
        return 0;  
    return 1;  
}
```

# Exemplo de implementação do TAD fila

Operações sobre o TAD fila:

- Criar elemento (**função reaproveitada do TAD pilha!**):

```
struct item criaItem(int valor){
    struct item *tmp;
    tmp = malloc(sizeof(struct item));
    tmp->valor = valor;
    tmp->prox = NULL;
    return tmp;
}
```

# Exercício

Implementar o TAD fila em forma de biblioteca (.h e .c), com todas as operações dadas, inclusive:

- ENQUEUE
- DEQUEUE (com retorno de item e liberação de memória na função principal)
- Mostra início e fim da fila
- Conta quantos elementos tem na fila

**ATENÇÃO:** verificar por *overflow* e *underflow*!

**Organize seu código:** separe as funções de pilha e fila, e as funções/estruturas “genéricas” que podem ser reusadas em outros programas no futuro.