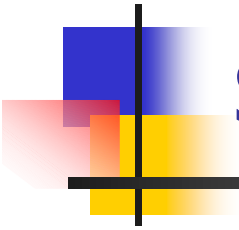


SIN211 Algoritmos e Estruturas de Dados

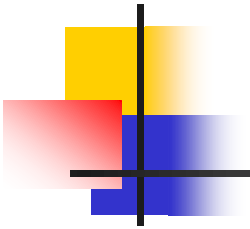
Prof. João Batista Ribeiro

joao42lbatista@gmail.com

UFV

Universidade Federal de Viçosa

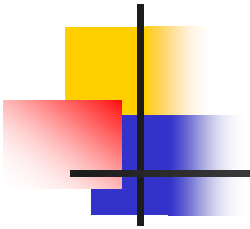
Slides baseados no material da Prof.ª Rachel Reis



Situação-Problema

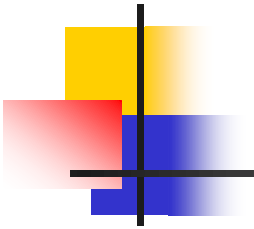
- Usando o conhecimento adquirido até o momento como vocês implementariam uma solução, usando os conceitos de Estrutura de Dados, para controlar o atendimento de clientes para uso do caixa eletrônico de um estabelecimento bancário?

  Que operações vocês sugerem?



Aula de Hoje

- TAD Fila



TAD Fila

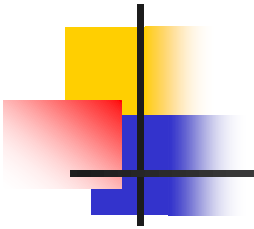
- Sequência de objetos, todos do mesmo tipo, sujeito às seguintes regras de comportamento:

  1) Sempre que solicitamos a remoção de um elemento, o elemento removido é o primeiro da sequência

  2) Sempre que solicitamos a inserção de um novo objeto, o objeto é inserido no fim da sequência.

- Em resumo ...

"O elemento removido é sempre o que está lá a mais tempo"

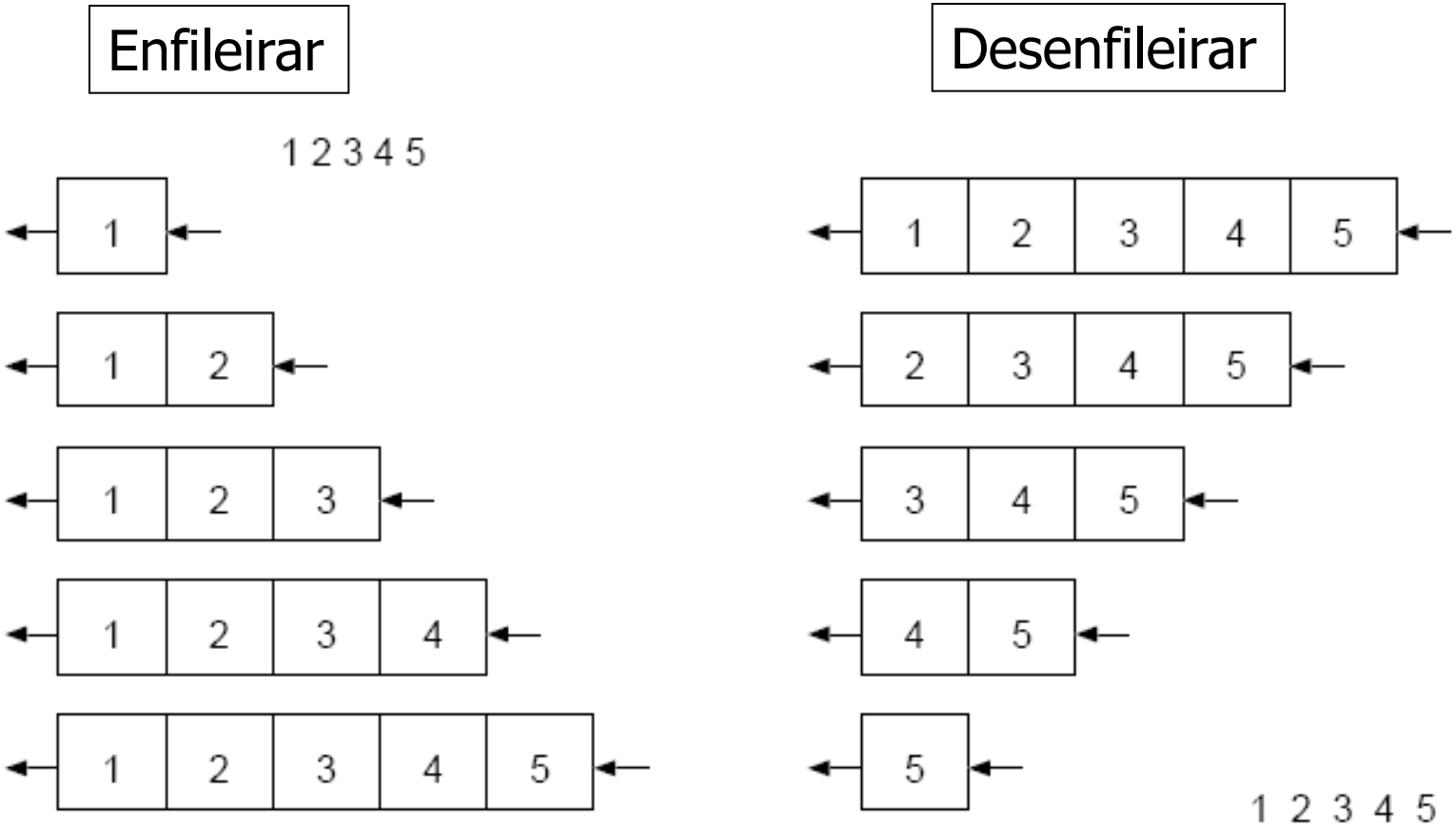


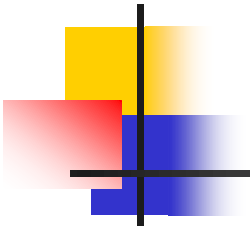
TAD Fila

- A estrutura de dados **fila** adota o critério FIFO (*First In First Out – o primeiro que entra é o primeiro que sai*)
- Operações principais:
  - enfileirar(x, F) // insere o elemento x no final da fila F
  - desenfileirar(F) // remove o elemento no início da fila F
- Outras operações:
  - inicializa(F) // inicializa a fila F no estado "vazia"
  - vazia(F) // indica se a fila F está vazia
  - cheia(F) /* indica se a fila F está cheia (útil para implementação estática)*/

TAD Fila

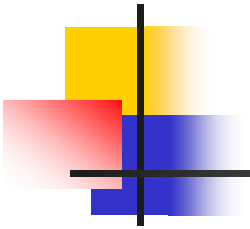
- Operações Enfileirar/Desenfileirar:





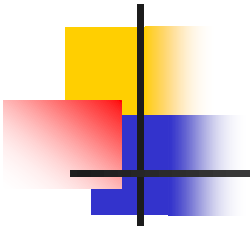
Implementação TAD Fila

- Como uma fila pode ser representada em C?



Implementação TAD Fila

- Como uma fila pode ser representada em C?
  - Uma ideia é usar um vetor para armazenar os elementos da fila e duas variáveis (início e fim) para armazenar o primeiro e o último elemento (implementação estática)
  - Outra ideia é utilizar lista simplesmente encadeada (implementação dinâmica)



Implementação TAD Fila - vetor

- Em uma implementação por meio de vetor os itens são armazenados em posições contíguas de memória
- A operação enfileirar faz a parte de trás da fila expandir-se
- A operação desenfileirar faz a parte da frente da fila contrair-se

Implementação TAD Fila - vetor

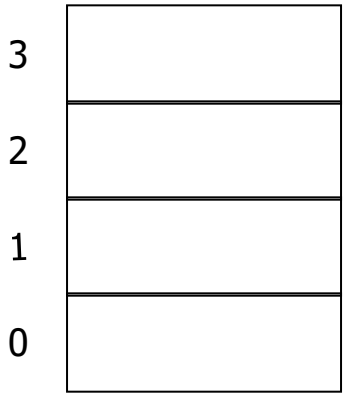
- Definição do tipo Fila

```
#define TAMMAX 10
typedef struct sFila{
    <TIPO> itens[TAMMAX];
    int inicio, fim;
}Fila;
```

→ Usar um vetor para armazenar uma fila introduz a possibilidade de estouro (caso a fila fique maior que o tamanho do vetor)

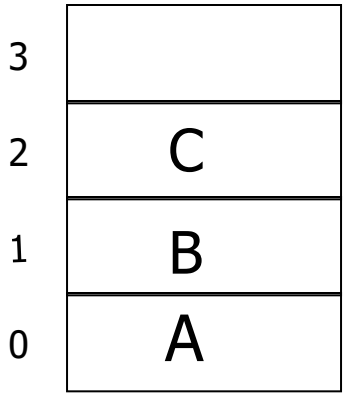
Implementação TAD Fila - vetor

1) Fila vazia



início = 0
fim = -1

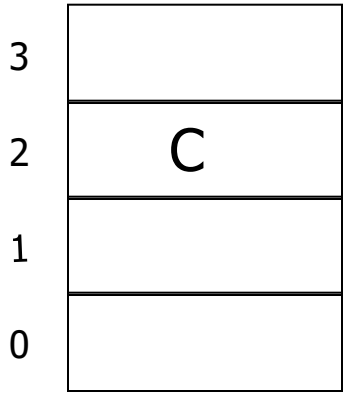
2) Insere A, B e C



fim = 2

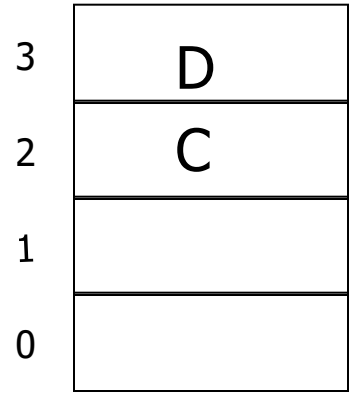
início = 0

3) Elimina dois itens



início = fim = 2

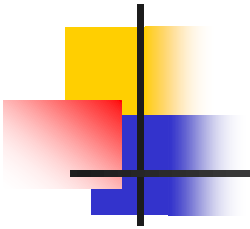
4) Insere novo item D



fim = 3

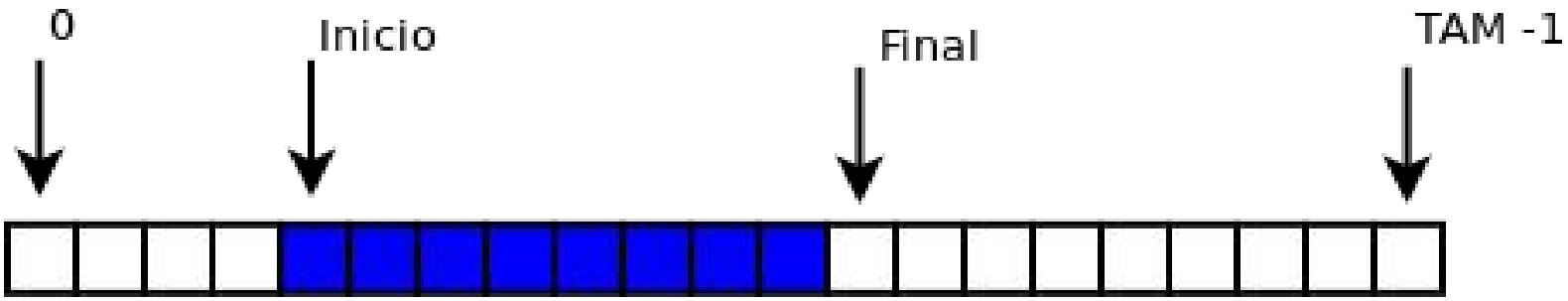
início = 2

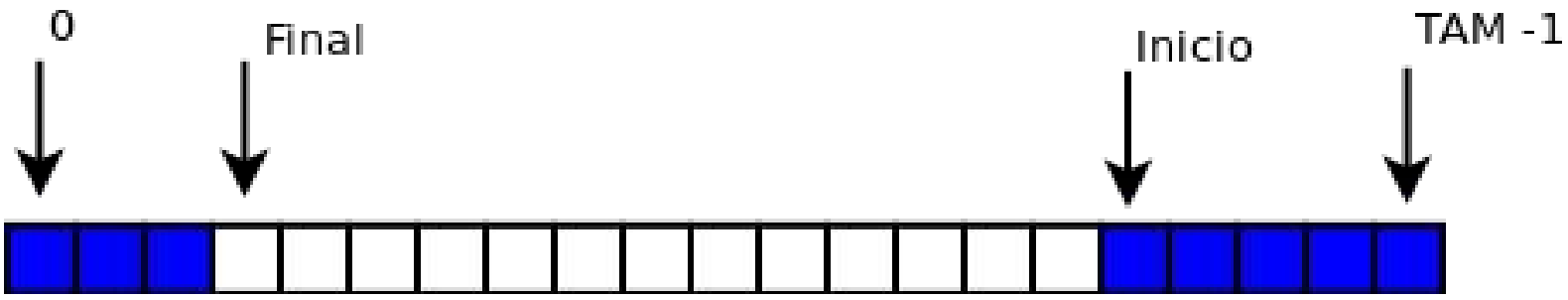
Implementação TAD Fila - vetor



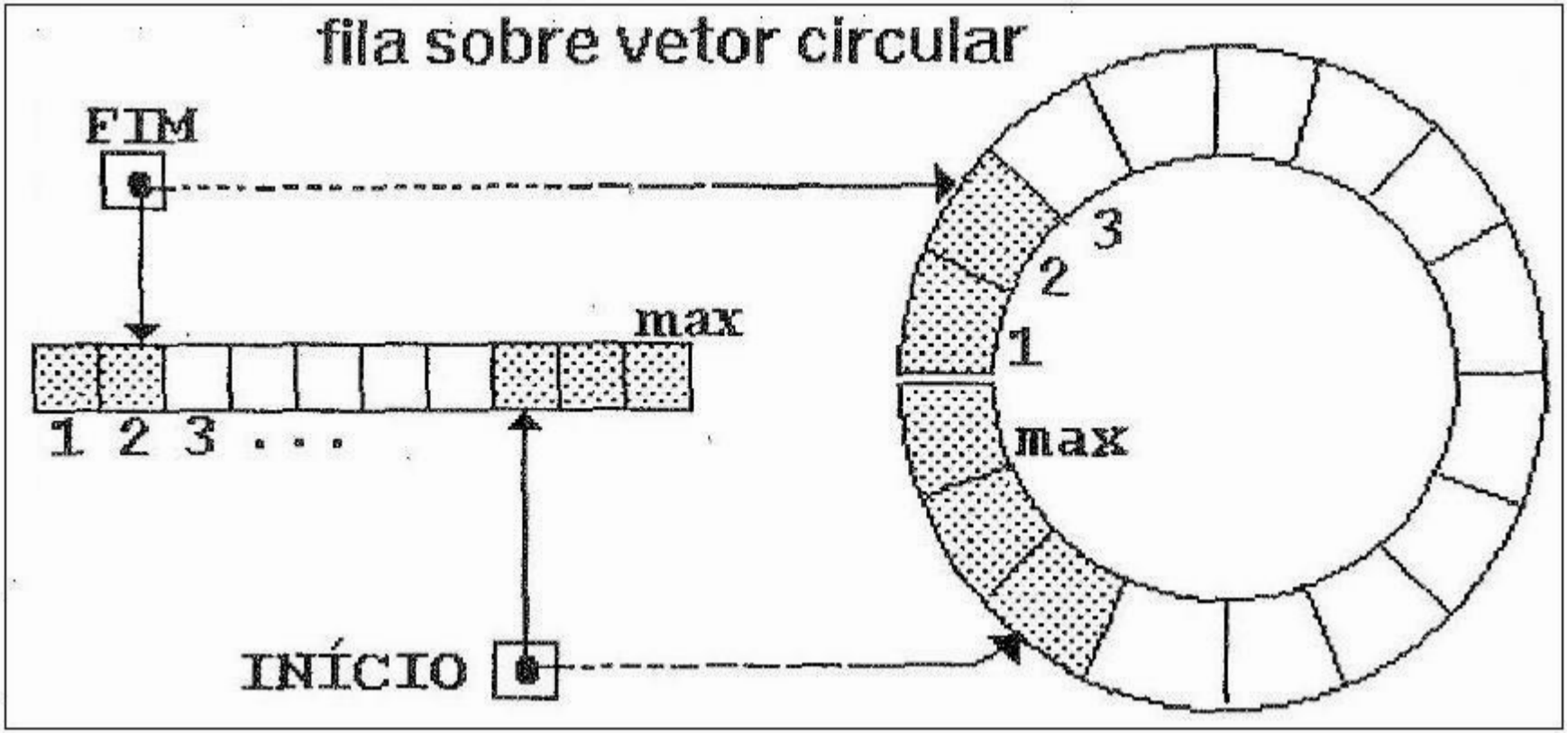
- Teste de fila vazia: fim < início
- Problema com o método anterior:
  - Chegar a situação absurda em que a fila está vazia, mas nenhum elemento novo pode ser inserido
- Como resolver?
  1) Modificar a operação *desenfileirar* de modo que, quando um item for removido, a fila inteira seja deslocada no sentido do início do vetor.
  2) Visualizar o vetor que armazena a fila como um círculo.

Implementação TAD Fila - vetor





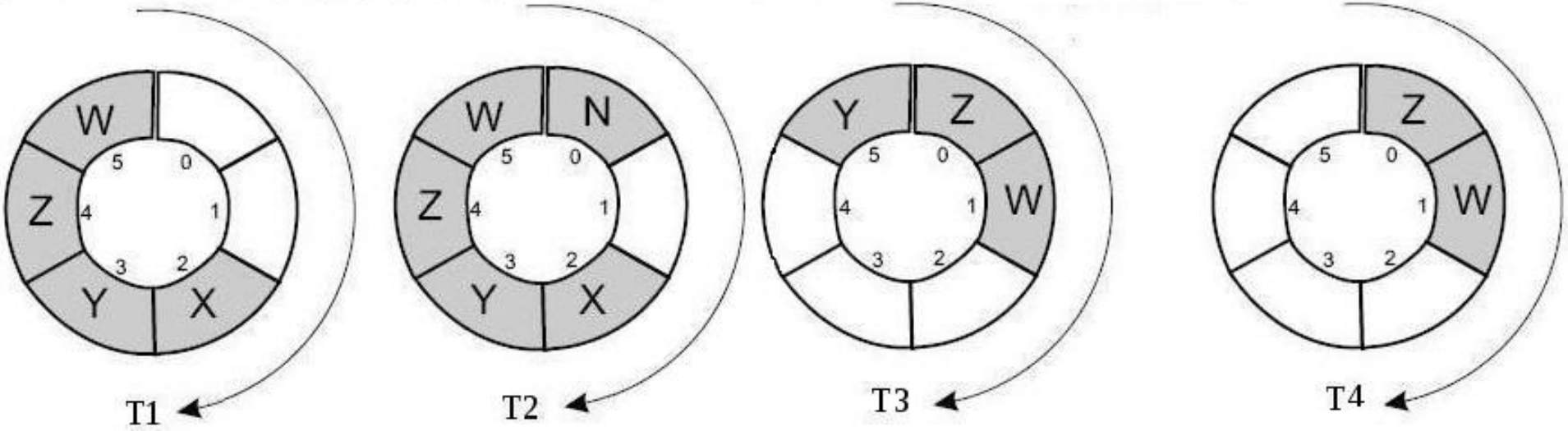
Implementação TAD Fila - vetor

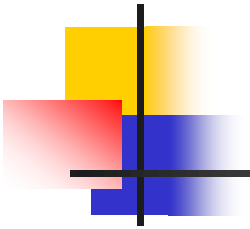


Implementação TAD Fila - vetor

	Início	Fim	Vetor 0	1	2	3	4	5	Operação
T1	2	5			X	Y	Z	W	Insere N
T2	2	0	N		X	Y	Z	W	

	Início	Fim	Vetor 0	1	2	3	4	5	Operação
T3	5	1	Z	W				Y	Retira da Fila
T4	0	1	Z	W					



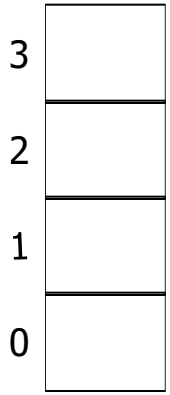


Implementação TAD Fila - vetor

- Solução para gerenciar início/fim na fila circular:
  - Abrir mão de um espaço na fila fazendo **início** sempre referenciar uma posição anterior ao início real da fila.

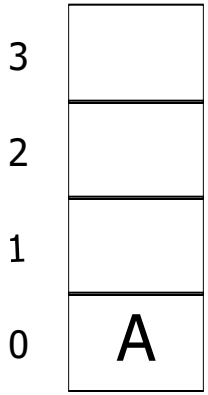
Implementação TAD Fila - vetor

1) Fila vazia



início = fim = 3

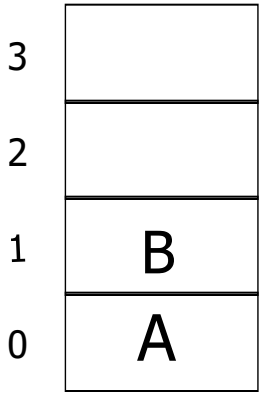
2) Insere A



início = 3

fim = 0

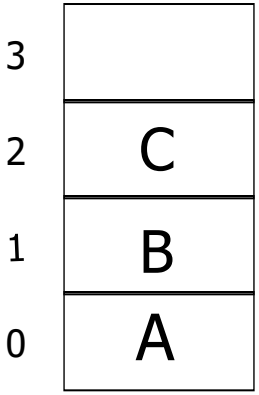
3) Insere B



início = 3

fim = 1

4) Insere C



início = 3

fim = 2

5) Insere D

Erro: fila cheia!

Implementação TAD Fila - *array*

- Inicializar a fila no estado "vazia"

```
void inicializa(Fila *f){
    f->inicio = TAMMAX -1;
    f->fim = TAMMAX - 1;
}
```

Implementação TAD Fila - *array*

- Verificar se a fila está vazia

```
int vazia (Fila *f) {
    if(f->fim == f->inicio)
        return 1;
    return 0;
}
```

Como seria a função para verificar se a fila está cheia?

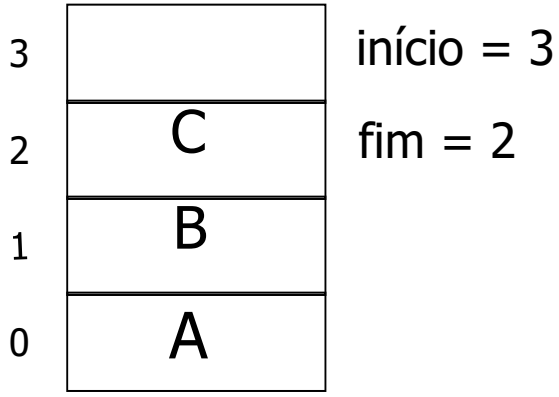
Implementação TAD Fila - *array*

```
void enfileirar (Fila* f, int x) {
     if(f->fim == (TAMMAX-1)){
        f->fim = 0;
     }else{
        (f->fim)++;
     }
     if (cheia(f)){ //(f->fim == f->inicio)
         printf("\nERRO: fila cheia.\n");
         (f->fim)--;
         if (f->fim == -1)
             f->fim = TAMMAX - 1;
         return;
     }
     f->itens[f->fim] = x;
}
```

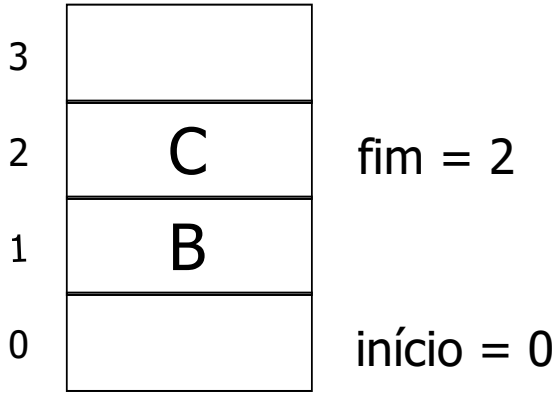
- Insere item na fila

Implementação TAD Fila - *array*

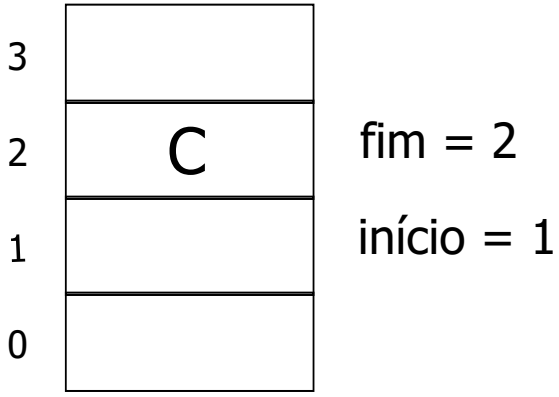
1) Fila Cheia



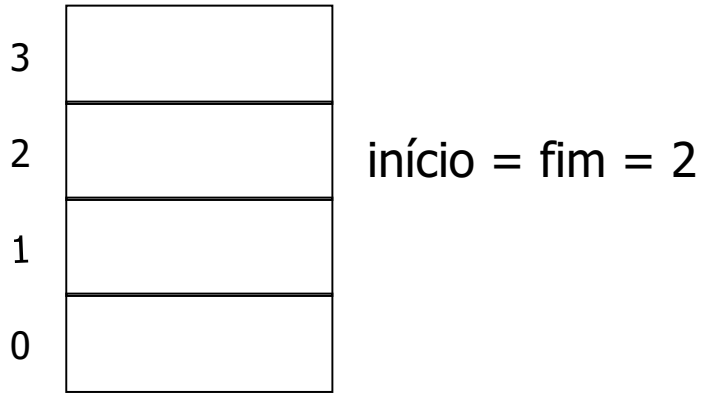
2) Remove



3) Remove



4) Remove



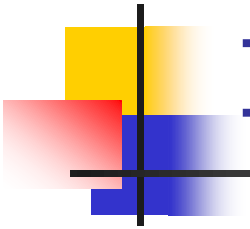
5) Remove

Erro: fila vazia!

Implementação TAD Fila - *array*

- Desenfileirar

```
int desenfileirar (Fila *f){
    int aux = 0;
    if (!vazia(f)) {
        if(f->inicio == TAMMAX-1){
            f->inicio = 0;
        } else {
            f->inicio++;
        }
        aux = f->itens[f->inicio];
    }else{
        printf ("\nERRO: fila vazia.\n");
    }
    return aux;
}
```



Implementação TAD Fila - *array*

- Impressão de elementos de uma fila:
  - A estrutura de dados "FILA clássica" não suporta a impressão de todos os elementos sem a remoção;
    - Entretanto, nada impede que você faça uma modificação na implementação de impressão apenas para leitura dos elementos (à título de verificação dos elementos).

Implementação TAD Fila - *array*

- Impressão (não clássica) de todos elementos de uma fila:

```
void imprimir(Fila *f){
    int i = (f->inicio + 1) %TAMMAX;
    if(!vazia(f)){
        printf("\nFila: ");
        while(i != ((f->fim + 1)%TAMMAX)){
            printf("%d ", f->itens[i]);
            i =(i + 1) % TAMMAX;
        }
    }else
        printf("\nFila vazia");
}
```

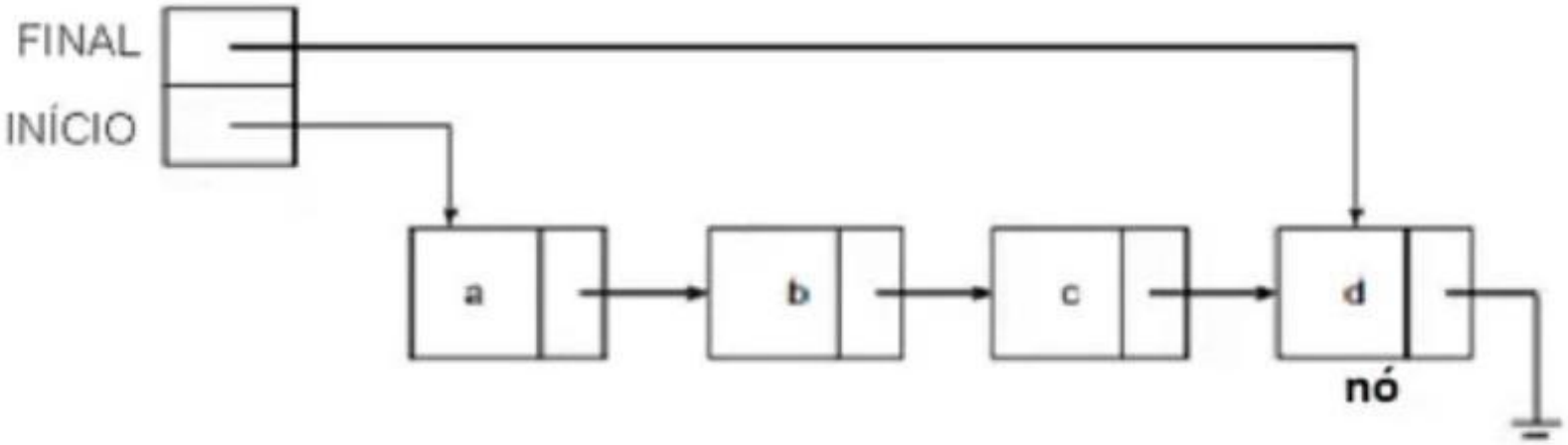
Implementação TAD Fila - *array*

Programa Principal

```
int main(){
    Fila ptrFila;

    inicializa(&ptrFila);

    enfileirar(&ptrFila, 2);
    imprimir(&ptrFila);

    enfileirar(&ptrFila, 3);
    imprimir(&ptrFila);

    desenfileirar(&ptrFila);
    imprimir(&ptrFila);

    enfileirar(&ptrFila, 4);
    imprimir(&ptrFila);
}
```

Implementação – lista encadeada

- Para implementar uma fila como uma lista encadeada, a fila precisa ter acesso às suas duas extremidades
  - Para isso será necessário armazenar dois ponteiros

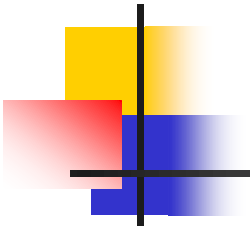


Implementação – lista encadeada

- Definição do tipo Fila

```
typedef struct cell{
    <TIPO> info;
    struct cell *prox;
}CELULA;
```

```
typedef struct sFila{
    CELULA* inicio;
    CELULA* fim;
}Fila;
```



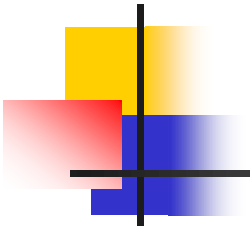
Implementação – lista encadeada

- Inicializa
  - Nome da função: inicializa
  - Tipo de retorno: void
  - Descrição: Inicializa o ponteiro “início” e “fim” com valor NULL
- Verifica se a fila está vazia
  - Nome da função: vazia
  - Tipo de retorno: int
  - Retornar 1 se a fila estiver vazia; 0 caso contrário.

Implementação – lista encadeada

- Inicializar

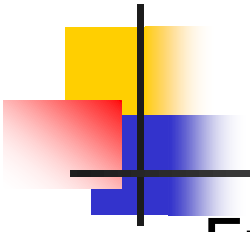
```
void inicializa(Fila *f){
   f->inicio = NULL;
   f->fim = NULL;
}
```



Implementação – lista encadeada

- Verifica se a lista está vazia

```
int vazia (Fila *f){
    if(f->inicio == NULL)
        return 1;
    return 0;
}
```



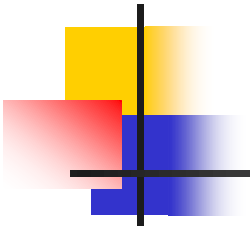
Implementação – lista encadeada

- Enfileirar
  - Nome da função: enfileirar
  - Tipo de retorno: void
  - Descrição: função responsável por criar, inicializar, inserir um elemento no final da fila
  - Levar em consideração os seguintes casos:
    1) Não foi possível alocar memória para o novo elemento
    2) A fila está vazia
    3) A fila não está vazia

Implementação – lista encadeada

- Enfileirar

```
void enfileirar(Fila *f, int x){
    CELULA *q;
    q = getnode();
    if(q! = NULL){
        q->info = x;
        q->next = NULL;
        if (vazia(f)) {
            f->inicio = q;
            f->fim = q;
        }else{
            (f->fim)->next = q;
            f->fim = q;
        }
    }else{
        printf ("\nERRO: falha na alocacao do. \n");
    }
}
```



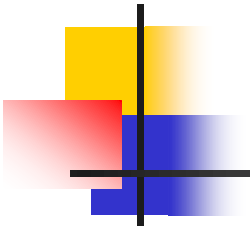
Implementação – lista encadeada

- Desenfileirar
  - Nome da função: desenfileirar
  - Tipo de retorno: <tipo_da_informação>
  - Descrição: função responsável por remover um elemento no início da fila.
  - Considerar as seguintes situações:
    - A fila está vazia (exibir mensagem de erro)
    - Depois de remover o elemento a fila ficou vazia
    - Depois de remover o elemento a fila não ficou vazia

Implementação – lista encadeada

- Desenfileirar

```
int desenfileirar(Fila *f){
    CELULA *q;
    int x;
    if (vazia(f)){
        printf("\nERRO: lista vazia");
        return -1;
    }else{
        q = (f->inicio)->next;
        x = (f->inicio)->info;
        free(f->inicio);
        f->inicio = q;
        if (f->inicio == NULL)
            f->fim = NULL;
        return x;
    }
}
```

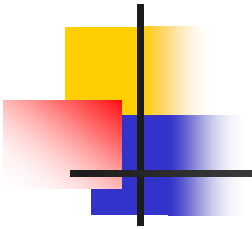


Filas com prioridades

- Definição: fila comum que permite que elementos sejam adicionados com uma prioridade.

- Cada elemento da fila deve possuir um dado adicional que represente sua prioridade de atendimento.

- Regra: o elemento de maior prioridade deve ser o primeiro a ser removido da fila quando uma remoção é requerida.



Filas com prioridades

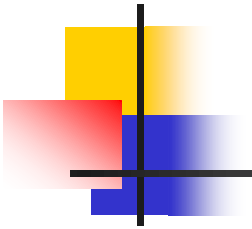
- Problema: encontrar uma implementação eficiente que permita uma rápida colocação e uma rápida retirada da fila.
  - Fila de banco: um cliente preferencial, idoso ou gestante, tem prioridades sobre outras no atendimento
  - Cabines de pedágio: veículos como “carros de polícia”, “ambulância”, “carros de bombeiro” podem passar imediatamente na frente de outros veículos, mesmo sem pagamento.

Filas com prioridades

- Implementação:

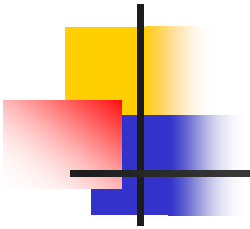
```
typedef struct sFila{
    <TIPO> info;
    int prioridade;
    struct sFila *prox;
}Fila;
```

```
Fila *ptrFila;
```



Filas com prioridades

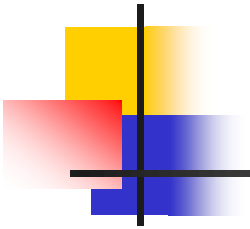
- Operações (extensão das operações básicas de fila comum)
  - Inserir com prioridade
  - Remover elemento de mais alta prioridade



Filas com prioridades

- Possíveis soluções:
  - Elementos inseridos ordenadamente em uma fila encadeada
  - Elementos são inseridos na posição apropriada, de acordo com sua prioridade, em uma fila encadeada

→ Qual possui melhor desempenho?
→ Qual apresenta melhor resposta?



Leituras Recomendadas

- DROZDEK, Adam. Estrutura de Dados e Algoritmos em C++. Editora Pioneira Thomson Learning, 2005.

→Pág. 130 (Fila)

→Pág. 138 (Filas com Prioridades)

- TENENBAUM A., LANGSAM Y. e AUGENSTEIN M. J. Estrutura de Dados usando C. Editora Makron, 1995.

→ Pág. 209 (Fila)

- FEOFILOFF, Paulo. Algoritmos em Linguagem C. Editora Campus, 2009.

→ Pág. 31 (Fila)