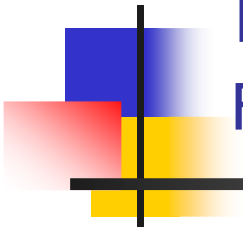
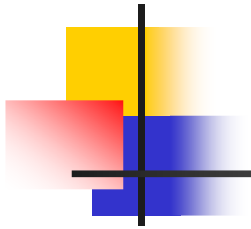


Estruturas de Dados – EDDA3

Fila Estática

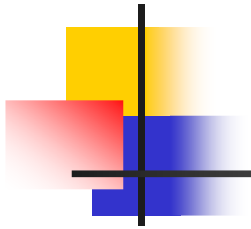
Prof. Marcelo Zorzan

Profa. Melissa Zanatta



Aula de Hoje

- TAD Fila

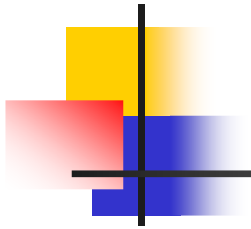


TAD Fila

- 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 há mais tempo".

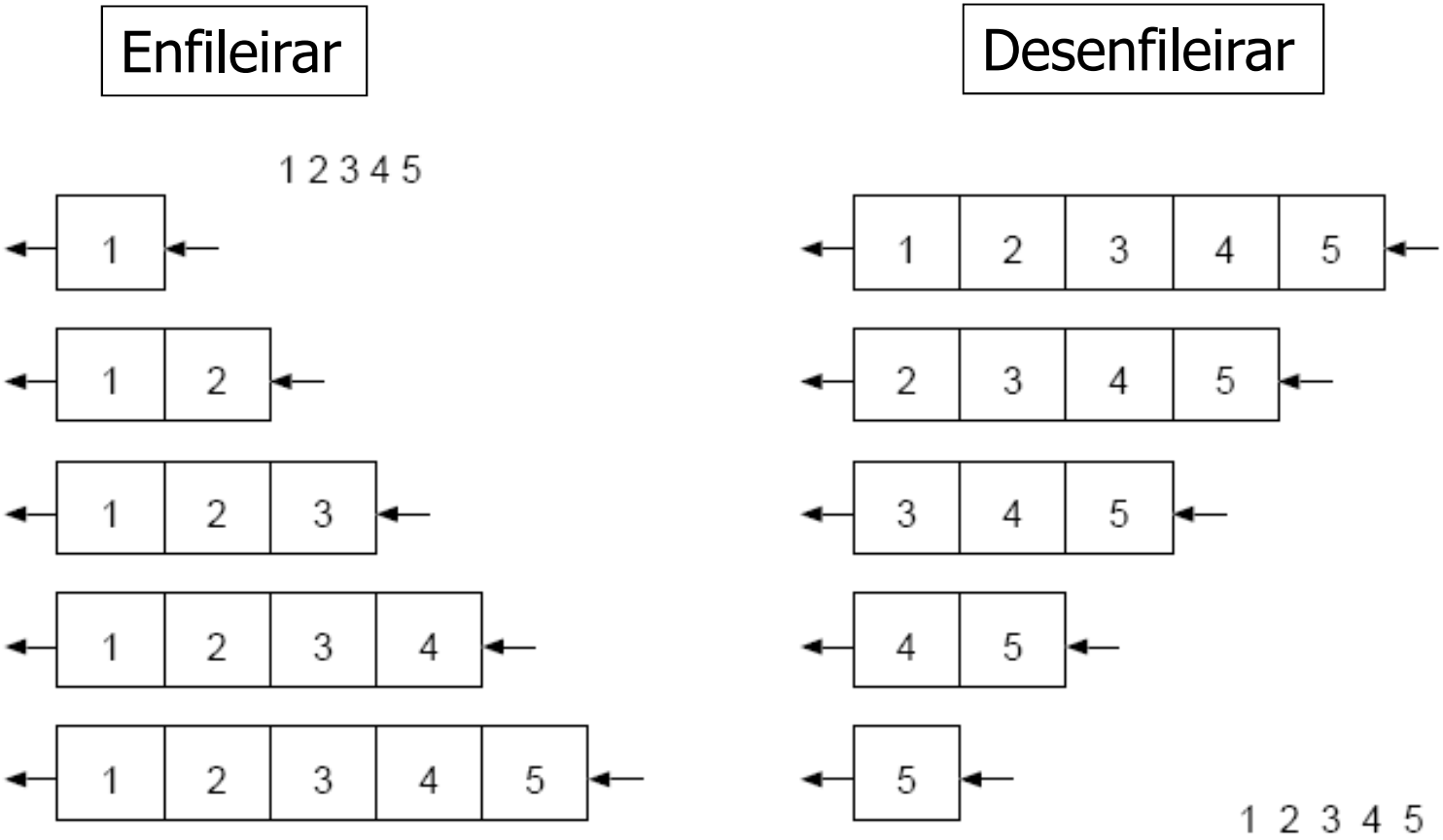


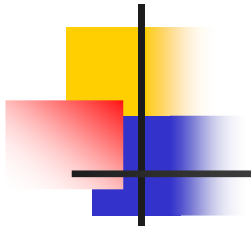
TAD Fila

- As estruturas de dados **fila** adotam 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

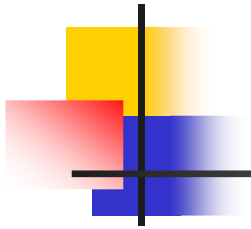
- Operações Enfileirar/Desenfileirar:





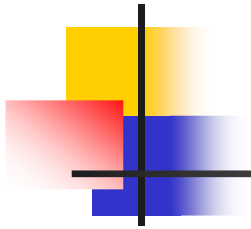
Implementação TAD Fila

- Como uma fila pode ser representada em C?
  - Uma idéia é usar um *array* para armazenar os elementos da fila e duas variáveis início e fim para armazenar o primeiro e o último elemento
  - Outra idéia é usar lista simplesmente encadeada



Implementação TAD Fila - *array*

- Em uma implementação por meio de *array* 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 - *array*

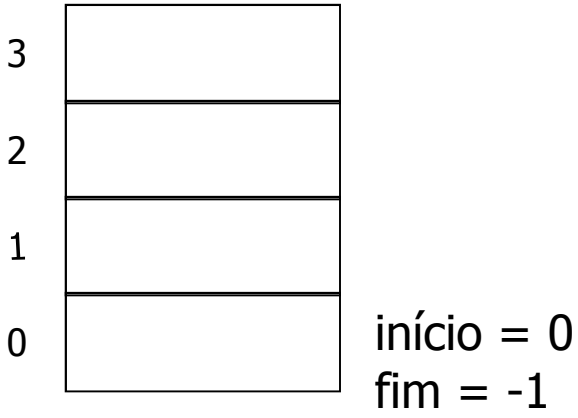
- Definição do tipo Fila

```
#define TAMMAX 10
typedef struct sFila{
      int items[TAMMAX];
      int inicio, fim;
}Fila;
```

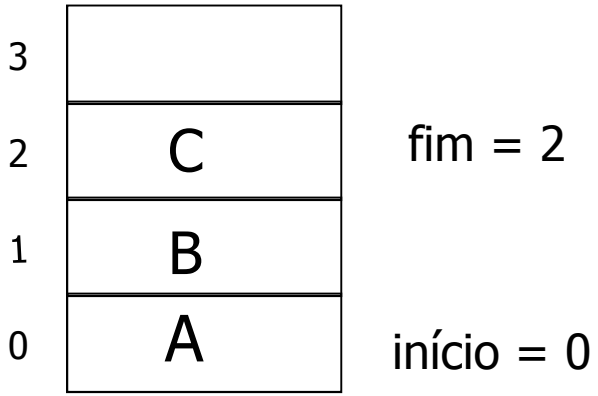
→ Usar um vetor para armazenar uma fila introduz a possibilidade de estouro (caso a fila fique maior que o tamanho do vetor)

Implementação TAD Fila - *array*

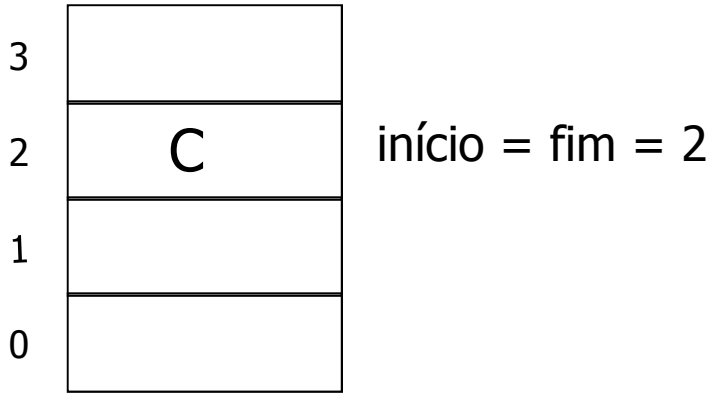
1) Fila vazia



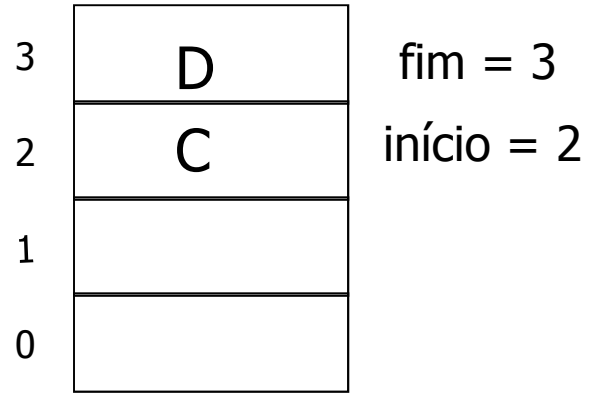
2) Insere A, B e C

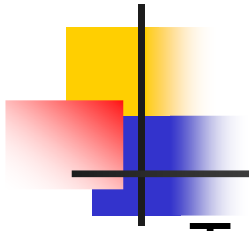


3) Elimina dois itens



4) Insere novo item

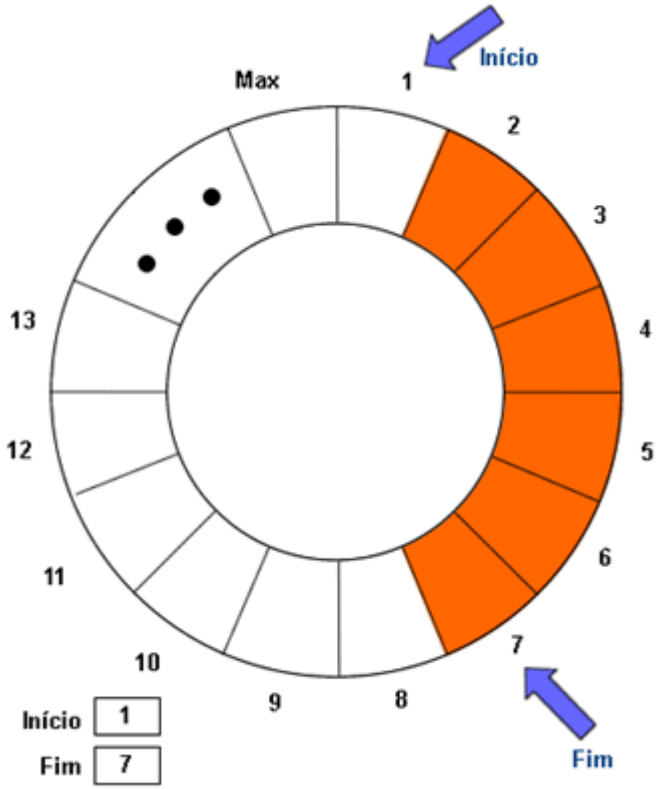




Implementação TAD Fila - *array*

- Teste de fila vazia: fim < inicio
- 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.

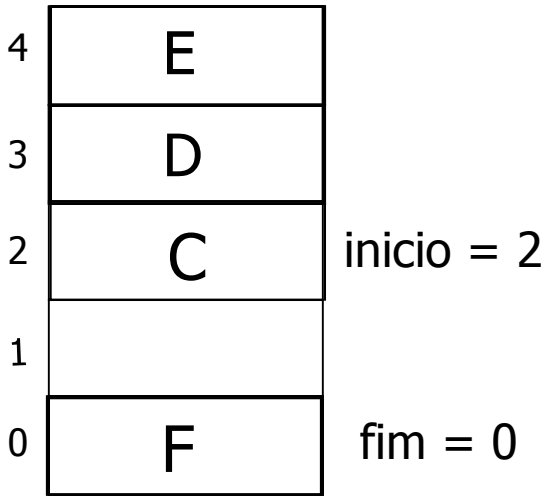
Implementação TAD Fila - *array*

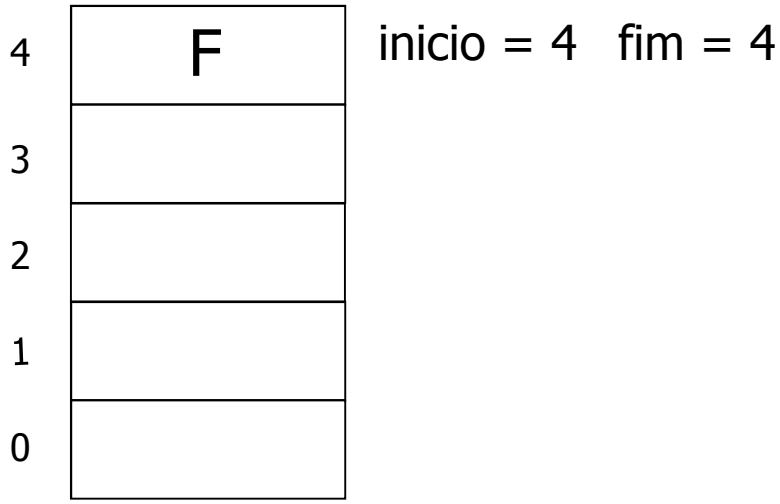


Fonte: http://www.lazilha.com/estrutura/filacircular.htm

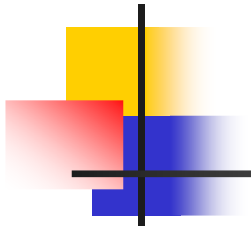
Implementação TAD Fila - *array*

- Como fazer a distinção entre fila cheia e vazia usando a representação circular?





Teste de fila vazia continua: fim < inicio???



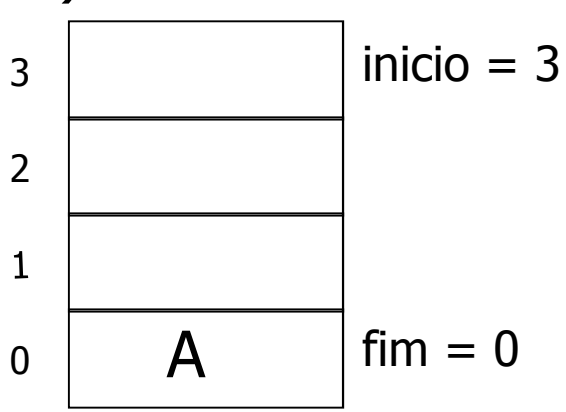
Implementação TAD Fila - *array*

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

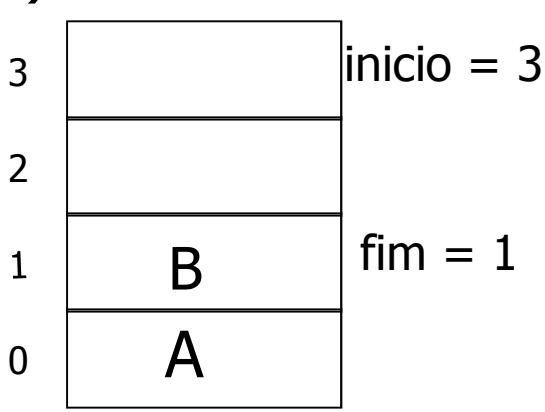
Implementação TAD Fila - *array*

1) Fila vazia

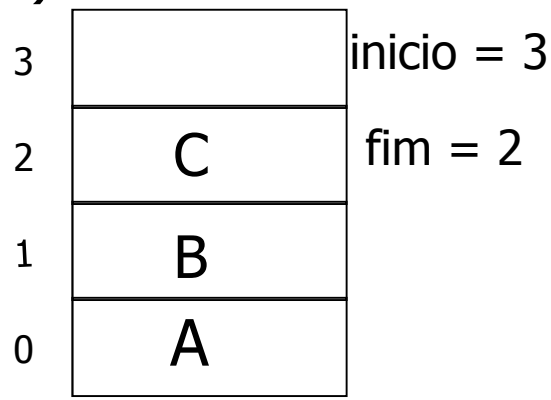
2) Insere A



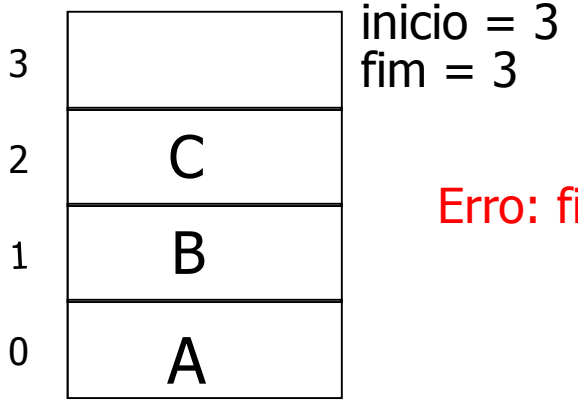
3) Inserindo B



4) Insere C



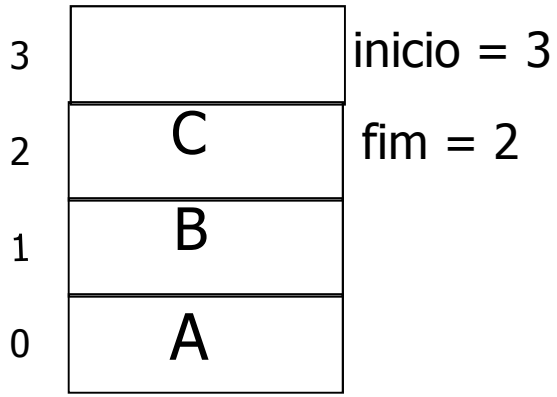
5) Insere D



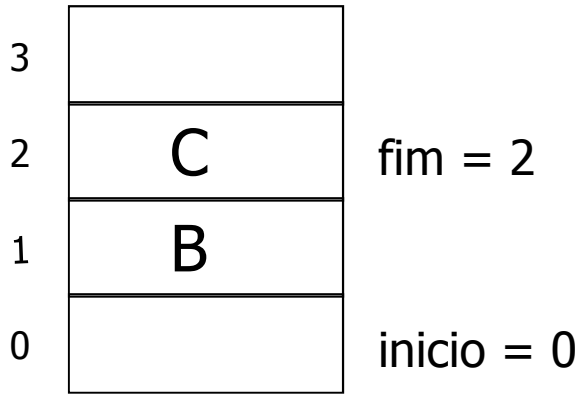
Erro: fila cheia!

Implementação TAD Fila - *array*

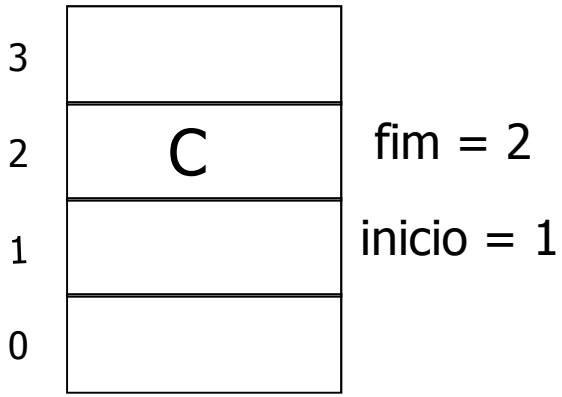
1) Fila Cheia



2) Remove

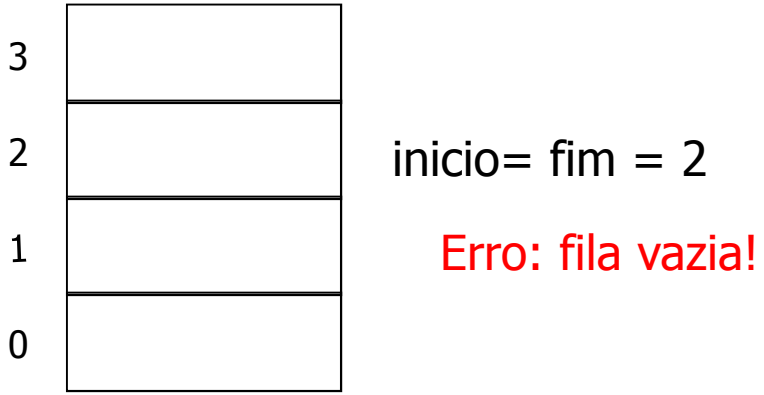


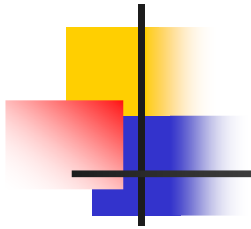
3) Remove



4) Remove

5) Remove

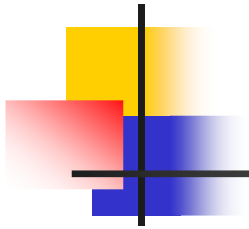




Implementação TAD Fila - *array*

- 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*

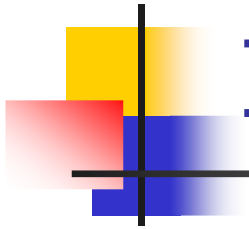
```
void enfileirar (Fila* f, int x) {
      if(f->fim == (TAMMAX-1)){
         f->fim = 0;
      }
      else{
           (f->fim)++;
      }
      if (cheia(f)){  //(f->fim == f->inicio)
          printf("\nERRO: fila cheia.\n");
          (f->fim)--;
          if (f->fim == -1)
            f->fim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

- Insere item na fila

Implementação TAD Fila - *array*

- 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:

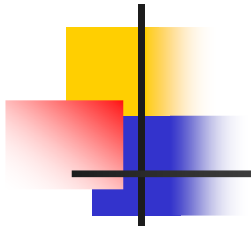
```
void imprimir(Fila *f){

     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);
}
```



Leituras Recomendadas

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

→Pág 130 (Fila)

- 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)