

# Fila Estática

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)  
Engenharia de Computação  
Departamento Acadêmico de Informática (Dainf)  
Universidade Tecnológica Federal do Paraná (UTFPR)  
Campus Pato Branco



# Sumário

- Fila
- Filas Estáticas
- TAD Fila Estática Circular

## Fila

- É uma lista em todas as inserções são realizadas em um extremo e a remoção em outro extremo
- *First-in, first-out*
- A inserção é realizada no final da lista e a remoção, no início



- Aplicações

- Gerenciamento de arquivos para impressão
- *Buffer* para a gravação de dados em mídia física
- Comunicação de redes de computadores
- Execução de processos no sistema operacional
- *Call centers*
- Etc

- Principais operações
  - Criar
  - Verificar se a fila está cheia
  - Verificar se a fila está vazia
  - Enfileirar
  - Desenfileirar
  - Verificar qual item está no início da fila
  - Verificar qual item está no final da fila
  - Liberar

- Representação
  - Alocação contígua (estática)



- Alocação encadeada



## Filas Estáticas

- Implementação semelhante ao da lista e da pilha estáticas
  - Uso de arranjo (vetor)
  - Inserção e remoção nas extremidades (dependendo do tipo de fila)
- Tipos de fila:
  - Simples
  - Circular
  - Prioridade
  - Deque

# Filas Estáticas

## Fila simples

- Pode ser utilizada uma variável para indicar o tamanho da fila

**tamanho = n**

$x_1$	$x_2$	...	$x_n$		
-------	-------	-----	-------	--	--

- Para a inserção, basta acrescentar o tamanho e inserir o novo elemento no final da fila

**tamanho++**

$x_1$	$x_2$	...	$x_n$	$x_{n+1}$	
-------	-------	-----	-------	-----------	--

- Em cada remoção,  $n - 1$  elementos devem ser deslocados, onde  $n$  é a quantidade de elementos existentes na fila

**tamanho--**

$$x_1 = x_2 \quad x_2 = x_3 \quad x_3 = x_4 \quad x_4 = x_5$$

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
-------	-------	-------	-------	-------	--

# Filas Estáticas

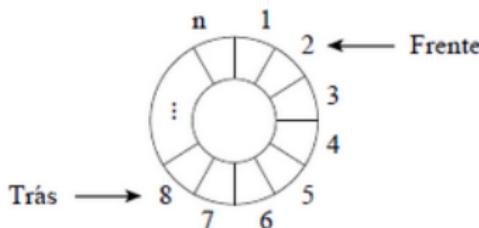
## Fila simples

- Principal problema da fila estática simples
- Solução do problema: fila circular

# Filas Estáticas

## Fila circular

- Nessa implementação, a fila se comporta de forma "circular"
- Na estrutura de dados, são necessárias as variáveis que indicam o início (*start*) e o fim (*end*) da fila
  - Imaginar a fila como um círculo



# Filas Estáticas

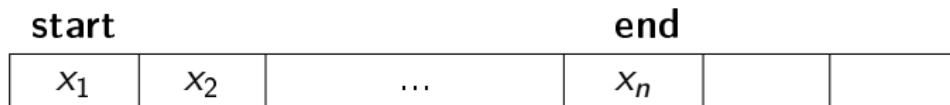
## Fila circular

- Quando um item é desenfileirado, a variável  $start$  é atualizada em uma das seguintes condições:
  - $start = -1$ : se a fila ficar vazia
  - $start = start + 1$ : se essa variável for menor que a última posição da fila
  - $start = 0$ : se essa variável for igual à última posição da fila

# Filas Estáticas

## Fila circular

- Quando um item é enfileirado (se houver espaço), a variável  $end$  é atualizada em uma das seguintes condições:
  - $end = end + 1$ : se a última posição fila ainda não foi alcançada
  - $end = 0$ : se essa variável for igual à última posição da fila



# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 10, 15, 20, 25, 5, 18, 9, 30

**start = end = -1**



# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 10, 15, 20, 25, 5, 18, 9, 30

**start = end = 0**

10							
----	--	--	--	--	--	--	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 15, 20, 25, 5, 18, 9, 30

**start = 0, end = 1**

10	15						
----	----	--	--	--	--	--	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 20, 25, 5, 18, 9, 30

**start = 0, end = 2**

10	15	20					
----	----	----	--	--	--	--	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 25, 5, 18, 9, 30

**start = 0, end = 3**

10	15	20	25				
----	----	----	----	--	--	--	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 5, 18, 9, 30

**start = 0, end = 4**

10	15	20	25	5			
----	----	----	----	---	--	--	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 18, 9, 30

**start = 0, end = 5**

10	15	20	25	5	18		
----	----	----	----	---	----	--	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar os seguintes números em uma fila de tamanho 8
  - 9, 30

**start = 0, end = 6**

10	15	20	25	5	18	9	
----	----	----	----	---	----	---	--

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar o seguinte item em uma fila de tamanho 8
  - 30

**start = 0, end = 7**

10	15	20	25	5	18	9	30
----	----	----	----	---	----	---	----

# Filas Estáticas

## Fila circular

- Exemplo: desenfileirar

**start = 1, end = 7**

	15	20	25	5	18	9	30
--	----	----	----	---	----	---	----

# Filas Estáticas

## Fila circular

- Exemplo: desenfileirar

**start = 2, end = 7**

		20	25	5	18	9	30
--	--	----	----	---	----	---	----

# Filas Estáticas

## Fila circular

- Exemplo: desenfileirar

**start = 3, end = 7**

			25	5	18	9	30
--	--	--	----	---	----	---	----

# Filas Estáticas

## Fila circular

- Exemplo: enfileirar o item 19

**start = 3, end = 0**

19			25	5	18	9	30
----	--	--	----	---	----	---	----

# Filas Estáticas

## Fila de prioridade

- Cada elemento tem um valor que representa a sua prioridade
- Nesse tipo de fila, os elementos podem ser organizados de acordo com a prioridade
  - Elemento com maior prioridade é posicionado no início da fila
  - Elemento com menor prioridade é posicionado no final da fila
- Operação enfileirar
  - Implementação 1: o elemento é inserido de forma ordenada na fila
  - Implementação 2: o elemento é colocado no final da fila (como na fila simples ou circular)

# Filas Estáticas

## Fila de prioridade

- Operação desenfileirar
  - Implementação 1: o elemento a ser removido é primeiro fila (caso esteja ordenada por prioridade)
  - Implementação 2: primeiramente é procurado o elemento de maior prioridade para sua respectiva remoção na fila fila
- Implementação de fila de prioridades:
  - Estática
  - Encadeada
  - *Heap\**
  - Árvores binárias de busca\*\*

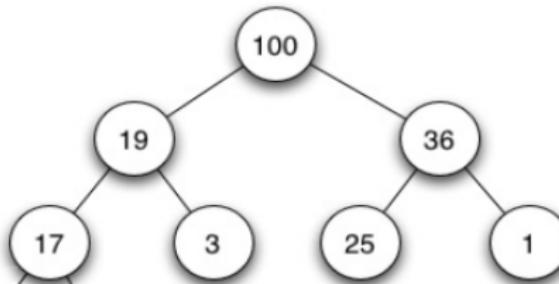
\*Não confundir com o espaço reservado para a alocação dinâmica. Também, essa estrutura será apresentada em uma das aulas sobre algoritmos de ordenação.

\*\*Serão apresentadas na disciplina "Algoritmos e Estrutura de Dados II".

# Filas Estáticas

## Fila de prioridade

- Ilustração de exemplo de *heap*
  - *Heap* é uma estrutura organizada em forma de árvore binária, onde cada nó tem prioridade maior ou igual em relação aos seus nós filhos/descendentes



# Filas Estáticas

## Deque

- *Double Ended QUEue*: "fila dupla" ou "fila com duas saídas"
- Tipo especial de fila e pilha
- Nesse tipo de fila, as operações de inserção e de remoção podem ser realizadas em ambas extremidades
  - Enfileirar no início
  - Enfileirar no final
  - Desenfileirar no início
  - Desenfileirar no final

- Exemplos de aplicações
  - Escalonamento de processos
  - Verificação de palíndromo
  - etc.

## **TAD Fila Estática Circular**

- Operações básicas para uma fila
  - Criar uma fila
  - Verificar se a fila está vazia
  - Verificar se a fila está cheia
  - Enfileirar (*enqueue*)
  - Desenfileirar (*dequeue*)
  - Verificar o elemento no início da fila
  - Verificar o elemento no final da fila
  - Imprimir fila
  - Liberar a fila

- Primeiro passo: definir arquivo .h

```
// Queue.h
#define TAM_MAX 100 // tamanho máximo da fila

typedef struct{
    int item[TAM_MAX];
    int ini;
    int fim;
    int tam;
}Fila;
```

# TAD Fila Estática Circular

```
Fila* criar_fila();  
  
int fila_vazia(Fila *f);  
  
int fila_cheia(Fila *f);  
  
int enfileirar(int chave, Fila *f);  
  
int desenfileirar(Fila *f);  
  
int verificar_inicio(Fila *f);  
  
int verificar_fim(Fila *f);  
  
void imprimir_fila(Fila *f);  
  
int liberar_fila(Fila *f);
```

- Segundo passo: definir arquivo .c

```
#include "fila.h"

Fila* criar_fila(){
    Fila *f = (Fila*) malloc(sizeof(Fila));
    f->ini = -1;
    f->fim = -1;
    f->tam = 0;

    return f;
}
```

## TAD Fila Estática Circular

```
int fila_vazia(Fila *f) {
    return (f != NULL) && (f->tam == 0);
}

int fila_cheia(Fila *f) {
    return (f != NULL) && (f->tam == TAM_MAX);
}
```

# TAD Fila Estática Circular

```
int enfileirar(int chave, Fila *f) {
    if (f != NULL)
        f = criar_fila();
    if (!fila_cheia(f)){
        if ((f->ini < 0) && (f->fim < 0))
            f->ini = f->fim = 0;

        if (f->fim < TAM_MAX - 1)
            f->fim++;
        else
            f->fim = 0;

        f->item[f->fim] = chave;
        f->tam++;

        return 1;
    }

    return 0;
}
```

# TAD Fila Estática Circular

```
int desenfileirar(Fila *f) {
    int item = INT_MIN;

    if (!fila_vazia(f)){
        item = f->item[f->ini];

        if (f->ini == f->fim) {
            f->ini = -1;
            f->fim = -1;
            f->tam = 0;
        }else if (f->ini < TAM_MAX - 1)
            f->ini++;
        else
            f->ini = 0;

        f->tam--;
    }

    return item;
}
```

# TAD Fila Estática Circular

```
int verificar_inicio(Fila *f) {
    if (!fila_vazia(f))
        return f->item[f->ini];
    else
        return INT_MIN;
}

int verificar_fim(Fila *f) {
    if (!fila_vazia(f))
        return f->item[f->fim];
    else
        return INT_MIN;
}
```

# TAD Fila Estática Circular

```
void imprimir_fila(Fila *f) {
    Fila aux;
    int item;

    if (f != NULL) {
        aux = *f;

        while (!fila_vazia(&aux)) {
            item = desenfileirar(&aux);

            printf("%d\n", item);
        }
    }
}

int liberar_fila(Fila *f) {
    if (f != NULL) {
        free(f);

        return 1;
    }

    return 0;
}
```

- Exercício 1: considerando o TAD anterior, implementar as funcionalidades para o TAD de fila para a seguinte *struct* (fila simples)

```
#define TAM_MAX 100 // tamanho máximo da fila

typedef struct{
    int item[TAM_MAX];
    int pos_end;
}Queue;
```

- Exercício 2: implemente um TAD para fila de prioridade
- Exercício 3: implemente um TAD para deque

# Referências |

-  Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Clifford, S.  
*Algoritmos: teoria e prática.*  
Elsevier, 2012.
-  Pereira, S. L.  
*Estrutura de Dados e em C: uma abordagem didática.*  
Saraiva, 2016.
-  Szwarcfiter, J.; Markenzon, L.  
*Estruturas de Dados e Seus Algoritmos.*  
LTC, 2010.
-  Tenenbaum, A.; Langsam, Y.  
*Estruturas de Dados usando C.*  
Pearson, 1995.
-  Ziviani, M.  
*Projetos de Algoritmos: com implementações em Pascal e C.*  
Thomson, 2004.