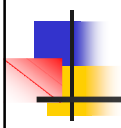
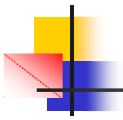


Filas

Prof. Rui Jorge Tramontin Jr.

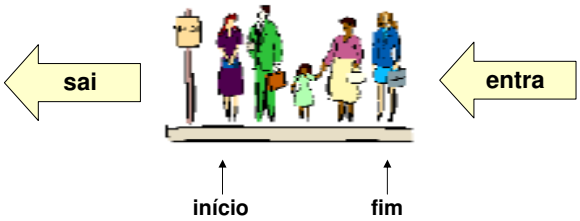


Índice

- Introdução
- Aplicações
- Implementação / Modelagem
- Operações de uma fila
- TAD da Fila em C

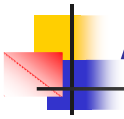
Introdução

- **Fila** é uma lista na qual as inserções são feitas em um lado (no **fim** da fila) e exclusões são feitas lado oposto (no **início** da fila).
- O primeiro elemento inserido será o primeiro a ser removido.
  - São também chamadas de listas **FIFO** (First In First Out).



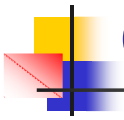
Introdução

- **Características das filas:**
  - Dados são organizados pela ordem de entrada.
- **Tipos de filas:**
  - Filas de espera (***queues***);
  - Filas com duplo fim (**deque** "*double-ended queue*");
  - Filas de espera com prioridades (***priority queues***).



Aplicações

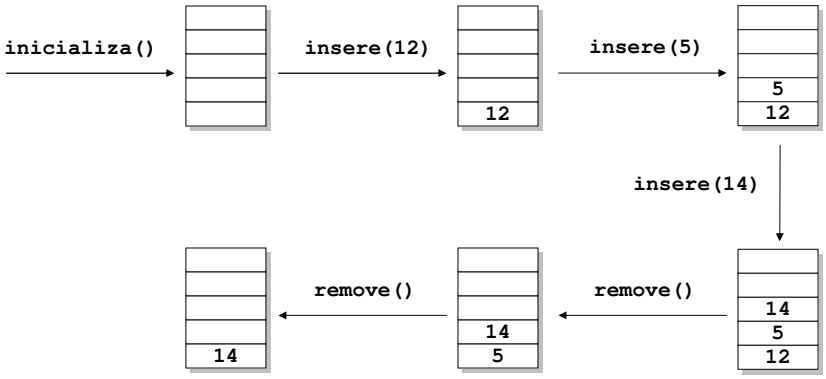
- Filas do sistema operacional
  - Processos, impressão, etc.
- Simulação
  - Filas para recursos quaisquer (ex. fila de banco).



Operações de uma fila

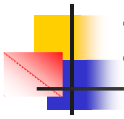
- **`inicializa_fila(F)`** : criar uma fila F vazia.
- **`insere(item, F)`** : insere item no fim de F.
- **`remove(F)`** : elimina o elemento do início de F.
- **`inicio(F)`** : acessa o elemento do início da fila (sem eliminar).
- **`estah_vazia(F)`** : testa se F está vazia.
- **`esta_cheia(F)`** : testa se F está cheia.

Exemplo



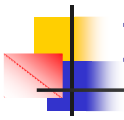
Implementação

- Usando vetores (circulares ou não).
- Utilizando alocação dinâmica (lista encadeada).
- O foco desta aula será na implementação usando **vetores**.



Implementação usando vetores

- Vantagens:
  - Facilidade de implementação.
- Desvantagens:
  - Vetores possuem um espaço limitado para armazenamento de dados.
  - Necessidade de definir um espaço grande o suficiente para a fila.
  - Necessidade de um indicador para o início e para o fim da fila.



Implementação usando vetores

- Método de Implementação:
  - A adição de elementos à fila resulta no incremento do indicador do fim da fila.
  - A remoção de elementos da fila resulta no incremento do indicador do início da fila.
- Nessa abordagem, a fila rapidamente perderá espaço para a inserção de mais elementos!

Como resolver isso?

Implementação usando vetores

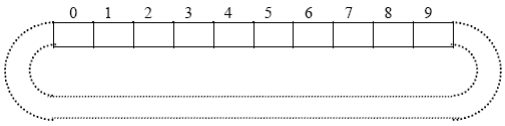
Soluções comuns:

- Juntar os elementos sempre à esquerda, cada vez que se faz uma remoção!

  (Não recomendado, baixo desempenho)

- Imaginar o vetor como sendo circular, isto é, unir o fim do vetor ao seu início.

  (Melhor implementação!!)



Modelagem da Fila

- Aspecto Estrutural:
  - Vetor para armazenar as informações.
  - Indicadores das posições de início e fim da fila.
  - Constantes que indiquem o tamanho da fila e para codificar situações de erros.
- Pseudo-código:

```
constantes MaxFila         100;
           ErroFilaVazia   -1;    // Underflow
           ErroFilaCheia   -2;    // Overflow

tipo Fila {
   inteiro   dados[MaxFila];
   inteiro   inicio;
   inteiro   fim;
};
```

Inicialização da Fila

```
procedimento inicializa_fila(var f: Fila)

início
    f.inicio <- 0;
    f.fim <- 0;
fim
```


Insere na Fila

```
função insere(var f: Fila; item: inteiro): inteiro

início
   se estah_cheia(f) então
      retorne ErroFilaCheia
   senão
      f.dados[f.fim] <- item
      se f.fim = MaxFila-1 então
         f.fim <- 0;                  // vetor circular!
      senão
         f.fim <- f.fim + 1;
      fimse
      retorne 1
   fimse
fim
```

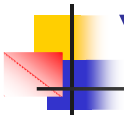
Remove da Fila

```
função remove(var f: Fila): inteiro
var
   removido: inteiro
início
   se estah_vazia(f) então
      retorne ErroFilaVazia
   senão
      removido <- f.dados[f.incio]
      se f.inicio = MaxFila-1 então
         f.inicio <- 0;                // vetor circular!
      senão
         f.inicio <- f.inicio + 1;
      fimse
      retorne removido
   fimse
fim
```

Retorna elemento no início

```
função início(f: Fila): inteiro

início
   se estah_vazia(f) então
      retorne ErroFilaVazia
   senão
      retorne f.dados[f.inicio]
   fimse
fim
```

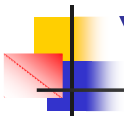


"Está Cheia" e "Está Vazia"

- Quando da inicialização da pilha, os campos *inicio* e *fim* têm valor 0.
  (está poderia ser a condição de fila vazia?)

- Como a fila é circular, inserções e remoções alteram a posição de início e fim:
  - Se a fila esvaziar, início e fim podem não ser 0.
  - No entanto, serão iguais!

  (E agora? fila está vazia se início == fim?)

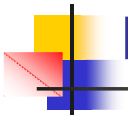


"Está Cheia" e "Está Vazia"

- No entanto, quando a fila está cheia, início também será igual a fim.

  PROBLEMA: Como distinguir entre *fila cheia* e *fila vazia*?

- Soluções comuns:
  - Manter sempre uma posição vazia no vetor entre o indicador de início e o indicador de fim.
  - Definir um indicador do número de elementos que a fila contém. → Por exemplo, vamos mostrar esta solução...

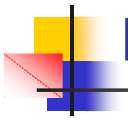


Re-Modelagem da Fila

- Adicionando o campo que indica o número de elementos.

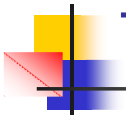
```
constantes MaxFila         100;
           ErroFilaVazia   -1;   // Underflow
           ErroFilaCheia   -2;   // Overflow

tipo Fila {
   inteiro   dados[MaxFila];
   inteiro   inicio;
   inteiro   fim;
   inteiro   num_elementos;
};
```

Re-Modelagem da Fila

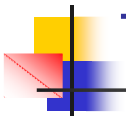
- Além da estrutura de dados, as seguintes funções devem ser adaptadas:
  - **inicializa_fila**: *num_elementos* deve ser inicializado com 0;
  - **insere**: *num_elementos* deve ser incrementado (opcionalmente, *num_elementos* pode ser retornado, ao invés de 1);
  - **remove**: *num_elementos* deve ser decrementado;
- Usando esse indicador, as funções *estah_cheia* e *estah_vazia* podem ser facilmente implementadas...



Testa se Fila está cheia

```
função estah_cheia(f: Fila): booleano

início
   se f.num_elementos = MaxFila então
      retorne VERDADEIRO
   senão
      retorne FALSO
   fimse
fim
```

Testa se Fila está vazia

```
função estah_vazia(f: Fila): booleano

início
   se f.num_elementos = 0 então
      retorne VERDADEIRO
   senão
      retorne FALSO
   fimse
fim
```

TAD da Fila em C (Fila.h)

```
// definição de constantes
#define MAX_FILA        50
#define ERRO_FILA_VAZIA -1
#define ERRO_FILA_CHEIA -2

// define estrutura e tipo Pilha
typedef struct {
        int dados[MAX_FILA];
        int inicio;
        int fim;
        int num_elementos;
} Fila;

// cabeçalho das funções
void inicializa_fila(Fila *fila);
int estah_vazia(Fila fila);
int estah_cheia(Fila fila);
int insere(Fila *fila, int valor);
int remove(Fila *fila);
int inicio(Fila fila);
```


Exercício

- Implementar as operações do TAD:
  - Fila.c;
- Escrever um programa em C que utilize o TAD Fila.