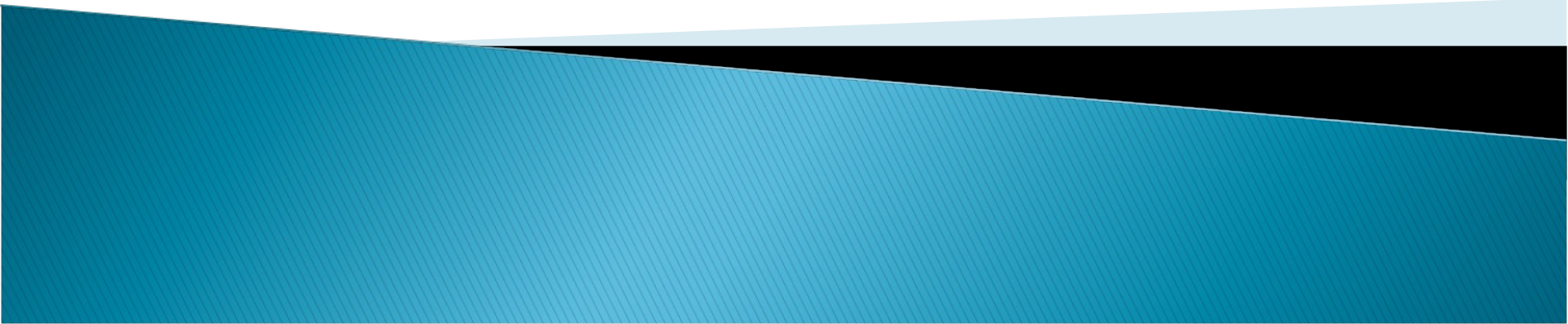


# Estrutura de Dados

## 05

Filas



# Filas

- ▶ A Fila é uma estrutura de dados dinâmica que obedece a uma regra rígida para a inserção e a retirada de elementos, conhecida como **FIFO**.
- ▶ O primeiro elemento a entrar na Fila deverá ser o primeiro elemento a sair da Fila.
- ▶ **First In First Out (FIFO)**.
- ▶ Sua implementação é semelhante a de qualquer lista linear encadeada (simples ou dupla) – exceto pelo FIFO.

# Fila (Parte 1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define tamanho 40

typedef struct tipoElemento {
    char nome[tamanho];
    tipoElemento *prox, *ante;
} TElemento;

typedef struct tipoFila {
    TElemento *inicio;
    TElemento *fim;
} TFila;

void inicializaFila(TFila *f);
int filaVazia(TFila *f);
void insere(TFila *f, char elemNome[tamanho]);
void exhibeFila(TFila f);
TElemento *retiraDaFila(TFila *f);
```

# Fila (Parte 2)

```
int main(){
    TElemento *elem;
    TFila fila;
    inicializaFila(&fila);

    insere(&fila, "Asdrubal Soares");
    insere(&fila, "Sandra Rosa Madalena");
    insere(&fila, "Madalena do Jucu");

    exibeFila(fila);

    elem = retiraDaFila(&fila);
    if (elem != NULL) free(elem);
    exibeFila(fila);

    elem = retiraDaFila(&fila);
    if (elem != NULL) free(elem);
    exibeFila(fila);

    elem = retiraDaFila(&fila);
    if (elem != NULL) free(elem);
    exibeFila(fila);

}
```

# Fila (Parte 3)

```
//=====
void inicializaFila(TFila *f){
    f->inicio = NULL;
    f->fim = NULL;
}
//=====
int filaVazia(TFila *f){
    if (f->inicio == NULL) return 1;
    else return 0;
}
//=====
void insere(TFila *f, char elemNome[tamanho]){
    //insere sempre no final da FILA ...
    TElemento *novo;

    novo = (TElemento *)malloc(sizeof(TElemento));
    strcpy(novo->nome, elemNome);
    novo->prox = NULL;
    novo->ante = NULL;

    if (filaVazia(f)){
        f->inicio = novo;
        f->fim = novo;
    } else {
        f->fim->prox = novo;
        novo->ante = f->fim;
        f->fim = novo;
    } //if
}
//=====
```

# Fila (Parte 4)

```
//=====
void exhibeFila(TFila f){
    TElemento *atual;
    int contador = 0;

    printf("\n\n\n\t\t\t====| Fila |====\n\n");

    atual = f.inicio;
    while (atual != NULL){
        printf("\t( %d ) - %s\n",++contador,atual->nome);
        atual = atual->prox; //move atual para o elemento seguinte
    }//while
    printf("\n");
    system("PAUSE");
}
//=====
```

# Fila (Parte 5)

```
//=====
TElemento *retiraDaFila(TFila *f){
    //sempre retira do início da FILA ...
    TElemento *retirado;

    retirado = NULL;

    if (!filaVazia(f)){
        retirado = f->inicio;

        f->inicio = f->inicio->prox;

        if (f->inicio == NULL) f->fim = NULL;
        else f->inicio->ante = NULL;

        retirado->prox = NULL;
    }//if
    printf("\n\nRETIRADO da FILA: %s\n\n", retirado->nome);
    system("PAUSE");
    return retirado;
}
//=====
```

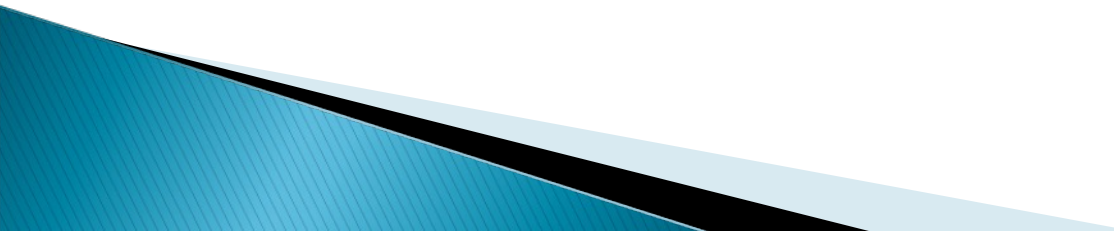
# Exercício de Fixação – 1

- ▶ Suponha a existência de uma pilha (stack) de inteiros  $s$  e uma fila (queue) de inteiros  $q$ . Desenhe a ilustração de  $s$  e  $q$  depois das seguintes operações:
- ▶ `Push(s, 3);`
- ▶ `Push(s, 12);`
- ▶ `insereFila(q, 5);`
- ▶ `insereFila(q, 8);`
- ▶ `x = Pop(s);`
- ▶ `Push(s, 2);`



# Exercício de Fixação – 1

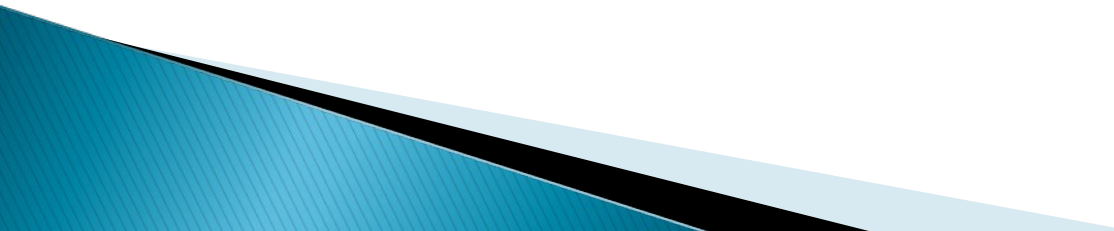
## (continuação)

- ▶ `insereFila(q, x);`
  - ▶ `Push(s,x);`
  - ▶ `y = elemTopo(s);`
  - ▶ `Push(s, y);`
- 


# Utilização de Filas

- ▶ O exemplo mais usual e simples de utilização de uma Fila é a chamada **Fila de Impressão**.
- ▶ Cada documento submetido para impressão é armazenado na **Fila de Impressão** antes de ser enviado para a impressora.
- ▶ Logicamente, os primeiros documentos enviados para essa fila serão impressos antes dos últimos documentos.

# Fila de Prioridades

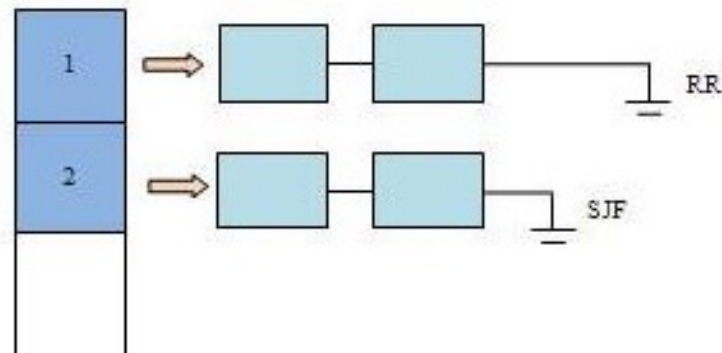
- ▶ Cada elemento tem uma prioridade.
  - ▶ Não necessariamente o primeiro inserido na fila será o primeiro a ser removido.
  - ▶ Deve ser removido da fila o elemento de maior prioridade. Havendo mais de um elemento com a mesma prioridade, deve ser retirado o primeiro inserido.
  - ▶ Muito utilizados em escalonadores de processos.
- 

# Fila de Prioridades

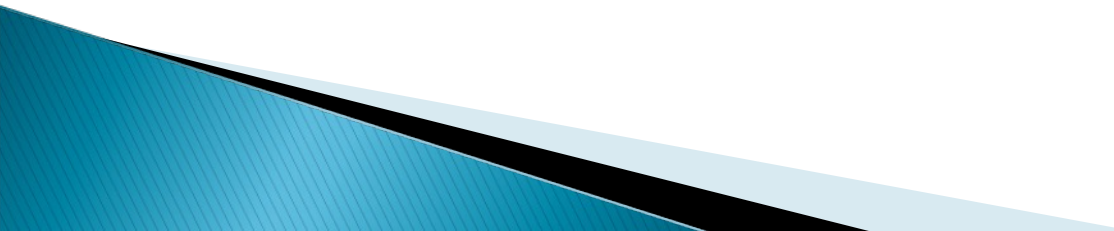
- ▶ Pode ser implementado utilizando listas encadeadas:
    - Cada elemento da lista teria um valor, um ponteiro para o próximo elemento, um ponteiro para o elemento anterior (se for duplamente encadeada) e uma prioridade;
    - Ao inserir na lista um elemento deve ser inserido após todos os elementos com prioridades maiores ou menores que ele;
    - A remoção se dá sempre no início da lista, ou seja, será removido sempre o elemento de maior prioridade.
- 

# Fila de Prioridades

- ▶ Pode ser implementado utilizando várias filas:
  - Cada fila teria uma prioridade;
  - Um elemento é sempre inserido no final da fila de sua prioridade;
  - A remoção se dá sempre no início da fila de maior prioridade que não esteja vazia.
  - Uma possibilidade de implementação seria utilizar um vetor de filas, sendo o índice do vetor a prioridade da fila.



# Starvation

- ▶ Um problema clássico em escalonadores de processos utilizando prioridades ocorre quando um processo nunca é executado pois sempre aparece um processo com prioridade maior. A esse fenômeno dá-se o nome de starvation.
  - ▶ Para resolver o problema utiliza-se políticas de redefinição de prioridade.
- 

# Starvation

- ▶ Aging: política em que, com o passar do tempo um processo vai tendo sua prioridade aumentada a fim de evitar starvation.
  - ▶ Por exemplo, a cada 10 segundos a prioridade do processo aumenta.
  - ▶ Um outro exemplo seria aumentar a prioridade de um processo após ele esperar pelo atendimento de  $n$  processos de prioridade mais alta