Filas

Disciplina: Estrutura de Dados

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: lucianobrum18@gmail.com

Tópicos

- O que é uma fila?
- Aplicações.
- Interface do tipo Fila.
- Implementação de Filas com vetor (contiguidade física).
- Implementação de Filas com listas encadeadas (alocação dinâmica).
- Resumo.

O QUE É UMA FILA?

- Fila é uma estrutura de dados em que:
  - O primeiro elemento inserido é o primeiro a ser retirado da fila.
- **Analogia: Fila de atendimento em bancos !**

O QUE É UMA FILA?

O QUE É UMA FILA?

- Ideia fundamental: a inserção de elementos na fila será sempre no final e a retirada dos elementos é do início da mesma.

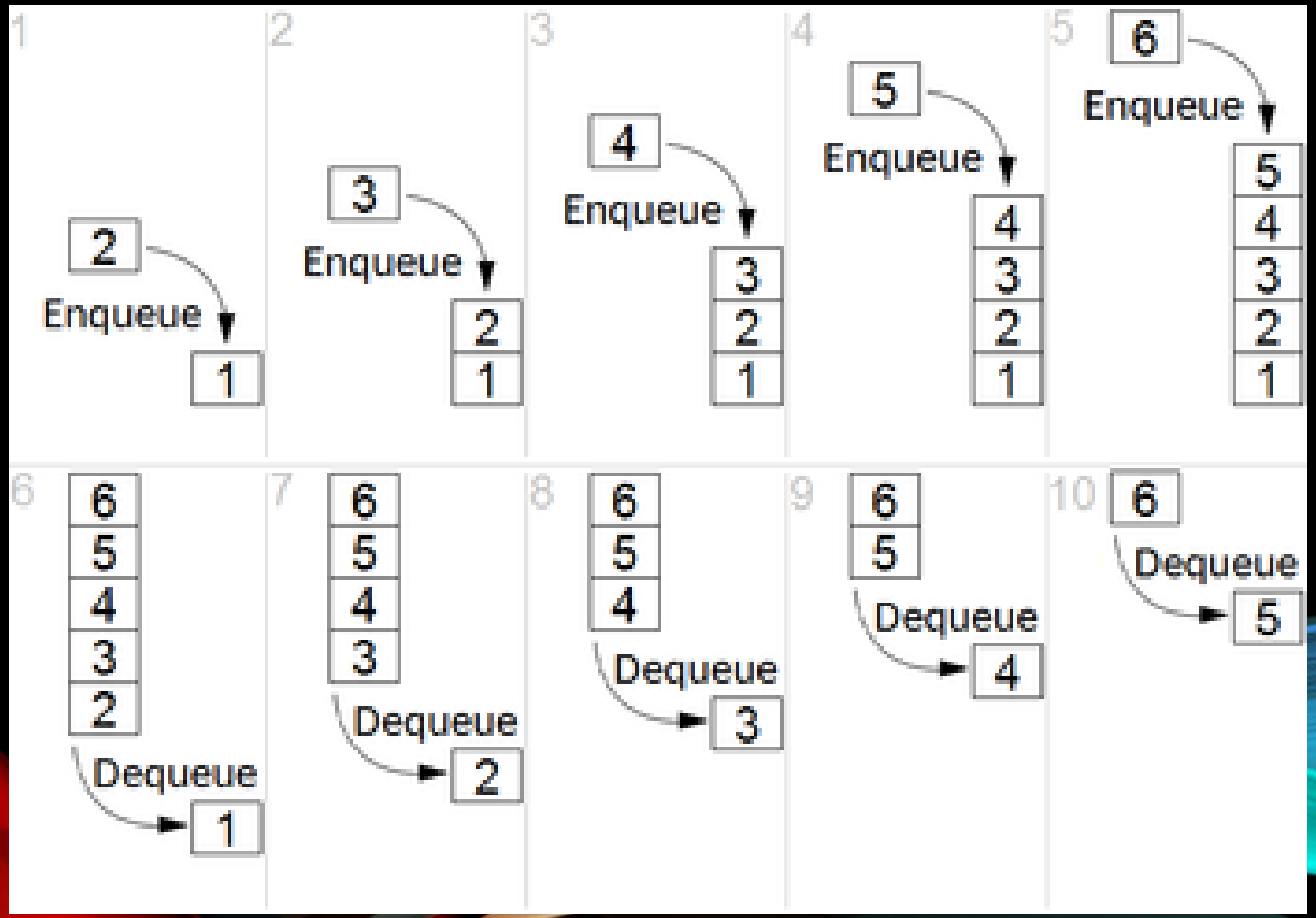
- Quando um elemento é inserido, ele vai para o fim da fila.

- **Segue estratégia FIFO (First in – First out).**

O QUE É UMA FILA?

- A fila possui duas operações básicas:
  - **Operação para inserir um elemento no final da fila;**
  - **Operação para retirar um elemento do início da fila;**

O QUE É UMA FILA?



Tópicos

- ~~O que é uma fila?~~
- Aplicações.
- Interface do tipo Fila.
- Implementação de Fila com vetor (contiguidade física).
- Implementação de Fila com listas encadeadas (alocação dinâmica).
- Resumo.

APLICAÇÕES

- **Fila de impressão de documentos da impressora:**
  - Se a impressora é compartilhada por várias máquinas, deve ser utilizada alguma estratégia para decidir qual documento deve ser impresso primeiro.
  - Uma das mais utilizadas estratégias é atribuir a mesma prioridade a todos documentos e imprimi-los na ordem em que foram submetidos.

APLICAÇÕES

- **Processos:**
  - Todos processos do sistema operacional, antes de serem atribuídos para algum processador, eles devem aguardar sua execução numa fila.

Tópicos

- ~~O que é uma fila?~~
- ~~Aplicações.~~
- **Interface do tipo Fila.**
- Implementação de Fila com vetor (contiguidade física).
- Implementação de Fila com listas encadeadas (alocação dinâmica).
- Resumo.

INTERFACE DO TIPO FILA

- **Vamos considerar a implementação de 5 operações:**
  - Criar uma fila vazia;
  - Inserir elemento no fim;
  - Remover elemento do início;
  - Verificar se a fila está vazia;
  - Liberar a estrutura da fila;

INTERFACE DO TIPO FILA

- **Podemos criar o arquivo fila.h, que representa a interface da pilha:**

```
typedef struct fila Fila;
Fila *fila_cria (void);
float fila_retira (Fila *f);
void fila_insere (Fila *f, float v);
int fila_vazia (Fila *f);
void fila_libera (Fila *f);
```

Tópicos

- ~~O que é uma pilha?~~
- ~~Aplicações.~~
- ~~Interface do tipo Fila.~~
- Implementação de Fila com vetor (contiguidade física).
- Implementação de Fila com listas encadeadas (alocação dinâmica).
- Resumo.

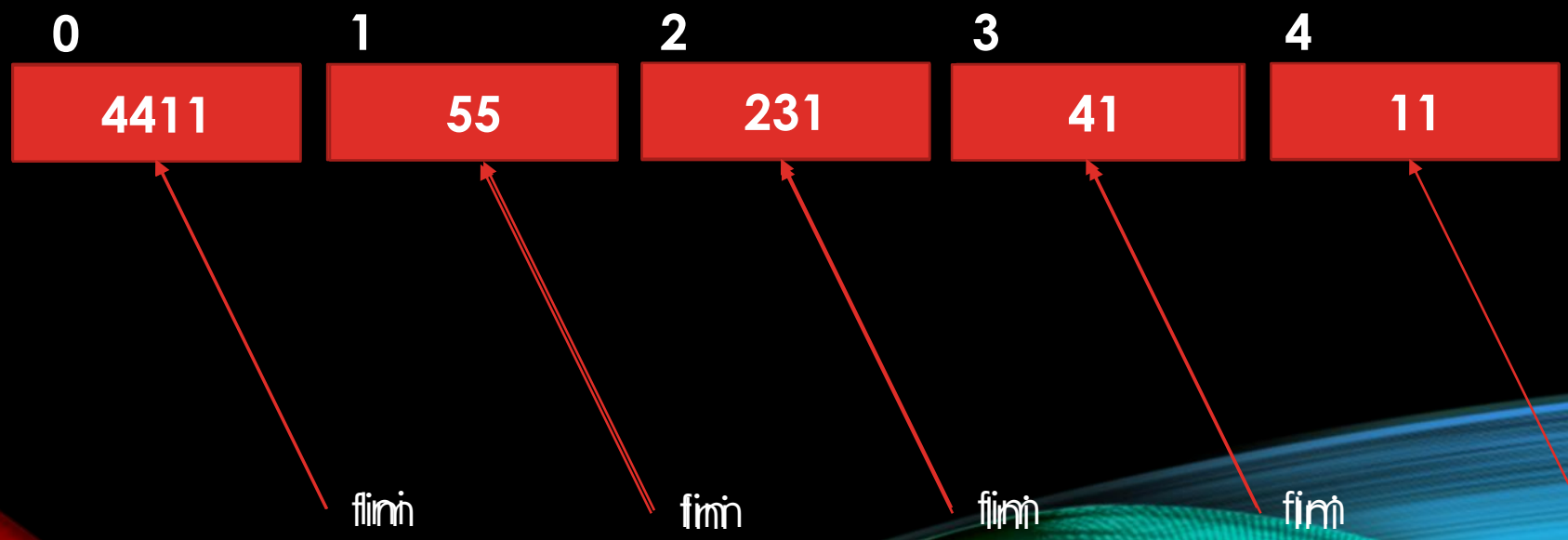
IMPLEMENTAÇÃO DE FILAS COM VETOR

- Se sabemos de antemão o número máximo de elementos possíveis na fila, podemos implementar a fila com **vetores.**

- A inserção e remoção de elementos da fila fará a fila “andar” no vetor.

- A figura mostra a ideia por trás da implementação de fila com vetor.

IMPLEMENTAÇÃO DE FILAS COM VETOR



IMPLEMENTAÇÃO DE FILAS COM VETOR

- Podemos ter a seguinte representação para o elemento fila:

```
#define TAM 500
struct fila{
    int n;
    int ini;
    float vet[TAM];
}
```

Tamanho do vetor

N° de elementos

Posição do início da fila

Elementos

IMPLEMENTAÇÃO DE FILAS COM VETOR

- Devemos ter uma forma de identificar o final da fila (posição onde será inserido o novo elemento), que mudará todo instante, assim como o início.

- Sabendo o número de elementos da fila e o início, podemos usar a seguinte expressão: fim = (ini + n) % N;

- Ex: ini = 3, n = 3 e N = 5, fim = 6 % 5 = 1

IMPLEMENTAÇÃO DE FILAS COM VETOR

- A função para criar a fila:

```
Fila *fila_cria (void){
    Fila *f = (Fila*) malloc(sizeof(Fila));
    f->n = 0;
    f->ini=0;
    return f;
}
```

IMPLEMENTAÇÃO DE FILAS COM VETOR

- A função para inserir um elemento na pilha:

```
void fila_insere (Fila *f, float v){
        if(f->n == TAM){
                printf("Capacidade da fila estourou.\n");
                exit(1);
        }
        fim = (f->ini + f->n)%TAM;
        f->vet[fim]=v;
        f->n++;
}
```

IMPLEMENTAÇÃO DE FILAS COM VETOR

- A função para retirar um elemento da fila:

```
float fila_retira (Fila *f){
        float v;
        if(fila_vazia(f)){
                printf("Fila já vazia.\n");
                exit(1);
        }
        v = f->vet[f->ini];
        f->n--;
        f->ini++;
        return v;
}
```

Realmente funciona pra todos casos?

IMPLEMENTAÇÃO DE FILAS COM VETOR

- A função para retirar um elemento da fila:

```
float fila_retira (Fila *f){
        float v;
        if(fila_vazia(f)){
                printf("Fila já vazia.\n");
                exit(1);
        }
        v = f->vet[f->ini];
        f->n--;
        f->ini=(f->ini+1)%TAM;
        return v;

}
```

IMPLEMENTAÇÃO DE FILAS COM VETOR

- A função para verificar se a fila está vazia:

```
int fila_vazia (Fila *f){
        return (f->n == 0); //1 = verdadeiro e 0 = falso
}
```

- A função para liberar a fila da memória:

```
void fila_libera(Fila *f){
        free(f);
}
```

Tópicos

- ~~O que é uma fila?~~
- ~~Aplicações.~~
- ~~Interface do tipo Fila.~~
- ~~Implementação de Fila com vetor (contiguidade física).~~
- Implementação de Fila com listas encadeadas (alocação dinâmica).
- Resumo.

IMPLEMENTAÇÃO DE FILAS COM LISTAS

- Se não sabemos de antemão o número máximo de elementos a serem inseridos, podemos implementar a fila com uma **estrutura de dados dinâmica**, por exemplo, uma **lista.**

- Os elementos são armazenados na lista e a fila agora possui 2 ponteiros: um para o início da lista e outro para o final da lista.

IMPLEMENTAÇÃO DE FILAS COM LISTAS

- Podemos ter a seguinte representação para os elementos lista e fila:

```
struct lista{
      float info;
      struct lista *prox;
};
typedef struct lista Lista;

struct fila{
      Lista *ini;
      Lista *fim;
};
```

IMPLEMENTAÇÃO DE FILAS COM LISTAS

- A função para criar a fila:

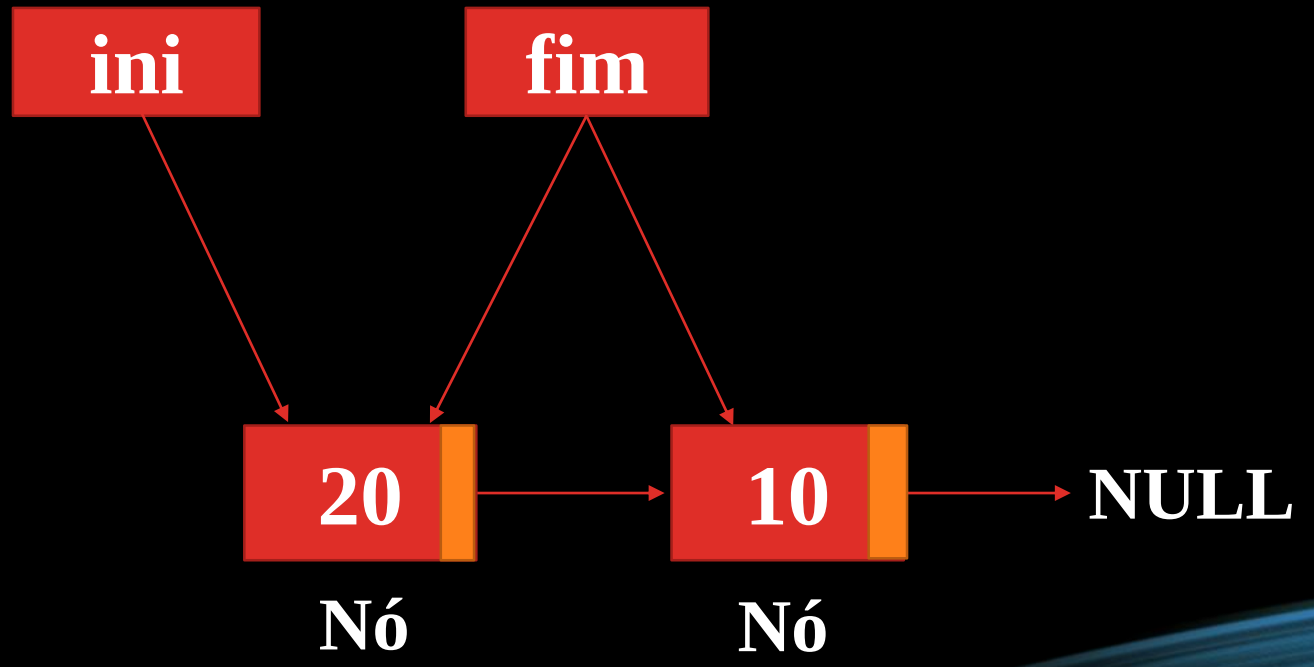
```
Fila* fila_cria (void){
    Fila *f = (Fila*) malloc(sizeof(Fila));
    f->ini = f->fim = NULL;
    return f;
}
```

IMPLEMENTAÇÃO DE FILAS COM LISTAS

- A função para inserir um elemento na fila:

```
void fila_insere (Fila *f, float v){
    Lista *n= (Lista*) malloc(sizeof(Lista));
    n->info = v;
    n->prox = NULL;
    if(f->fim!=NULL){ //fila não estava vazia?
            f->fim->prox = n;
    }
    else{
            f->ini = n;
    }
    f->fim = n;
}
```

IMPLEMENTAÇÃO DE FILAS COM LISTAS

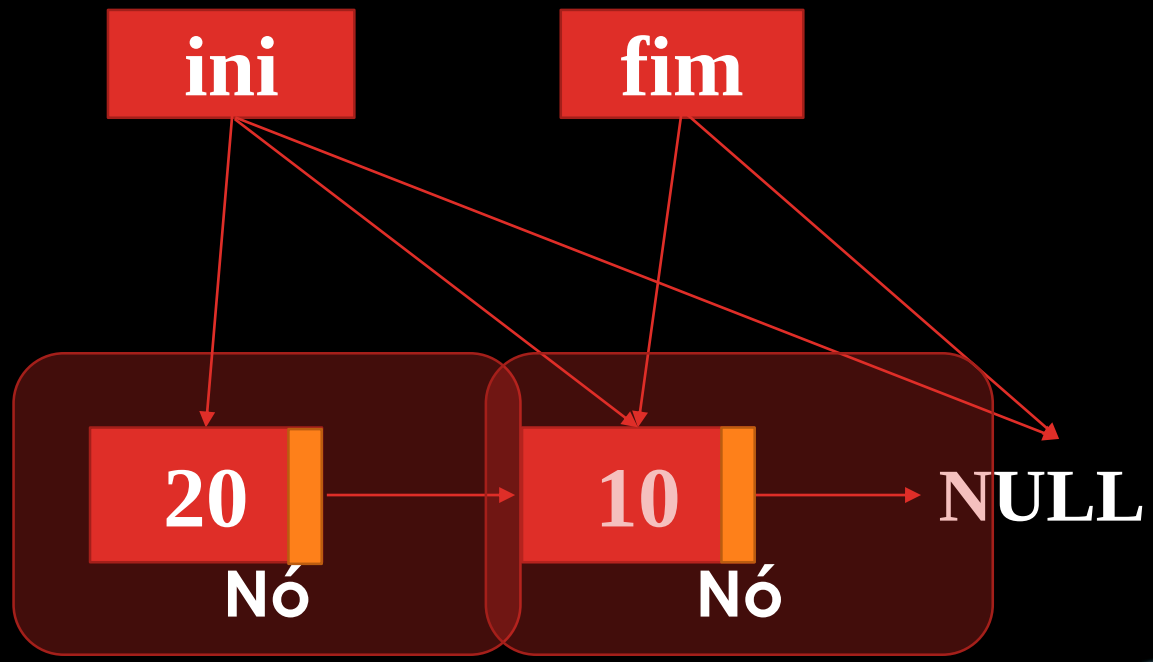


IMPLEMENTAÇÃO DE FILAS COM LISTAS

- A função para retirar um elemento da fila:

```
float fila_retira(Fila *f){
    if(fila_vazia(f)){ printf("Fila Vazia!.\n"); exit(1);}
    Lista *no = f->ini;
    float v = no->info;
    f->ini = f->ini->prox; //ou f->ini = no->prox;
    if(f->ini == NULL){
        f->fim = NULL;
    }
    free(no);
    return v;
}
```

IMPLEMENTAÇÃO DE FILAS COM LISTAS



IMPLEMENTAÇÃO DE FILAS COM LISTAS

- A função para verificar se a fila está vazia:

```
int fila_vazia (Fila *f){
        return (f->ini == NULL); //1 = verdadeiro e 0 = falso
}
```

- A função para liberar a fila deve também liberar a lista:

```
void fila_libera(Fila *f){
        Lista *q = f->ini;
        while(q!=NULL){
                Lista *t = q->prox;
                free(q);
                q = t;
        }
        free(f);
}
```

IMPLEMENTAÇÃO DE FILAS COM LISTAS

- Podemos ter uma função que mostre os elementos da fila, para fins de teste:

```
void fila_imprime(Fila *f){
      int i;
      for(i = 0, i < f->n; i--){
            printf(“%f\n”, f->vet[(f->ini+i)%TAM]);
      }
}
void fila_imprime(Fila *f){
      Lista *q;
      for(q = f->ini; q!=NULL; q=q->prox){
            printf(“%f\n”, q->info);
      }
}
```

Tópicos

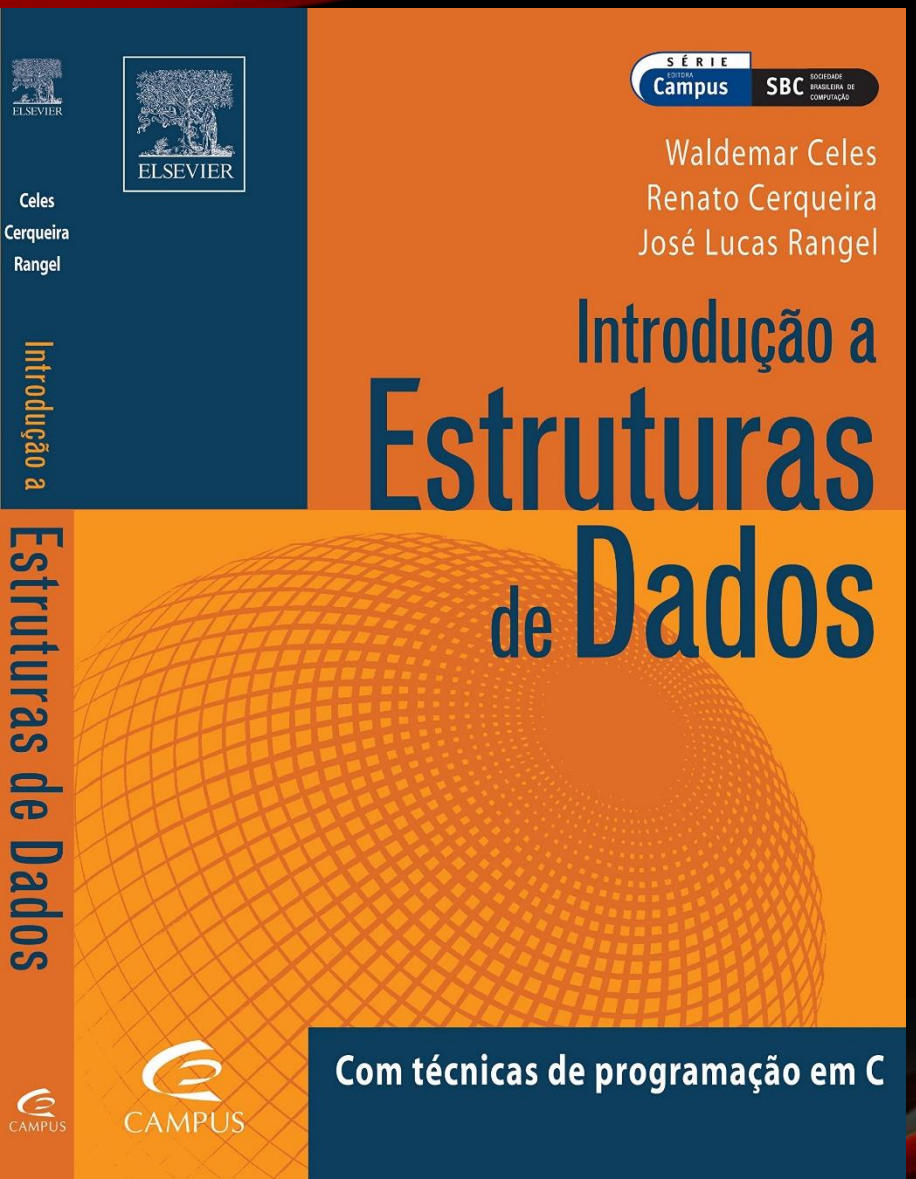
- ~~O que é uma pilha?~~
- ~~Aplicações.~~
- ~~Interface do tipo Fila.~~
- ~~Implementação de Fila com vetor (contiguidade física).~~
- ~~Implementação de Fila com listas encadeadas (alocação dinâmica).~~
- **Resumo.**

RESUMO

- Foi demonstrado:
  - Funcionamento de uma fila com vetor e lista;
  - Aplicações;
  - Operações básicas e interface;
  - Exemplos em C;

DÚVIDAS ?

REFERÊNCIAS



- CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. **Introdução a Estruturas de Dados com técnicas de programação em C**. Rio de Janeiro: Elsevier (Campus), 2004. 4ª Reimpressão. 294 p.