

Algoritmos e Estruturas de Dados II

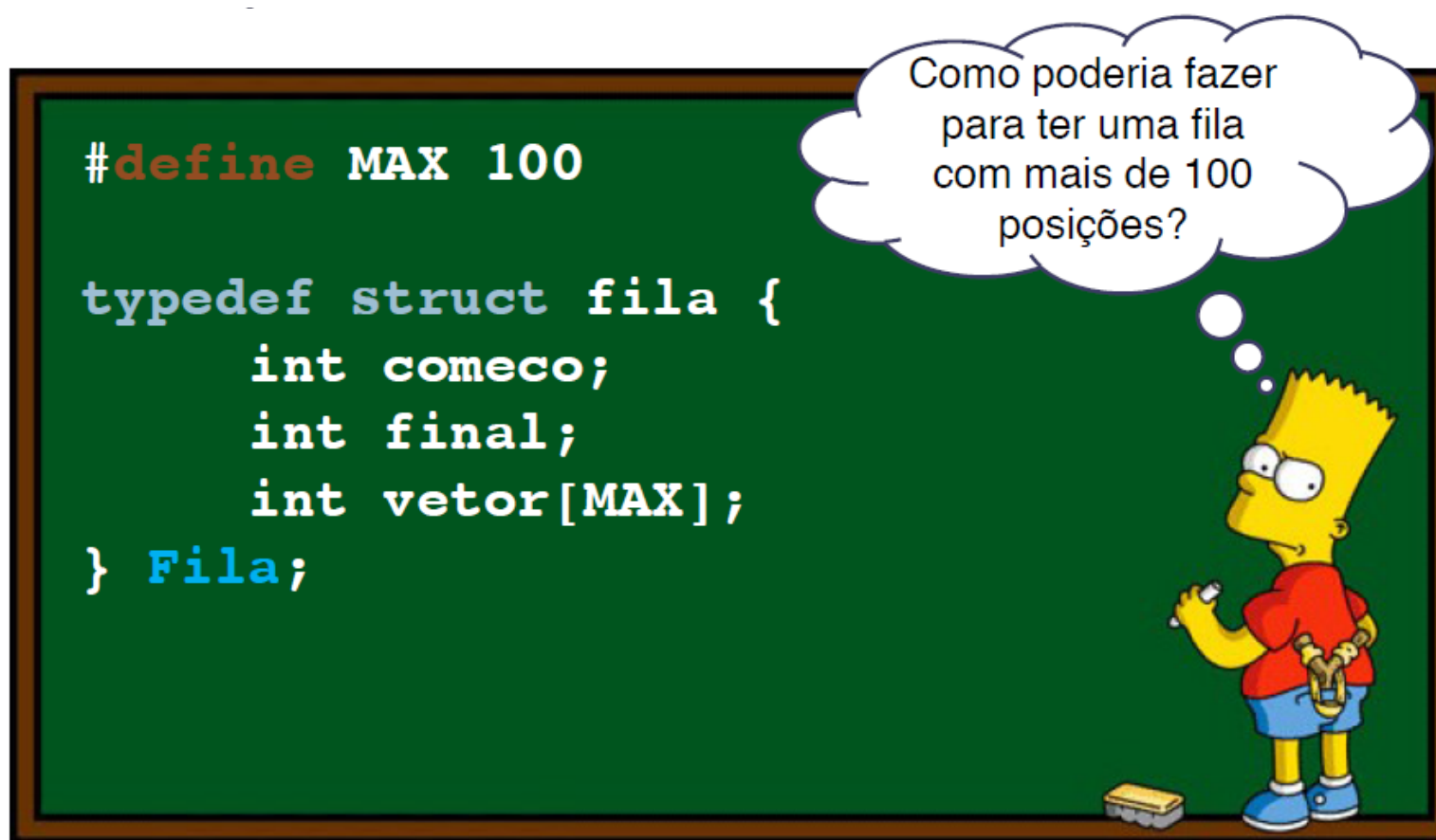
2º Período Engenharia da Computação

Prof. Edwaldo Soares Rodrigues
Email: edwaldo.rodrigues@uemg.br

Estruturas encadeadas – Encadeamento de memória através de ponteiros

Estruturas encadeadas

- Alocação Estática x Dinâmica



Estruturas encadeadas

- Alocação Estática x Dinâmica:
 - Alocação estática: ocorre em tempo de compilação. Quando o programa entra em execução ele “reserva” toda a memória necessária para sua execução;
 - Alocação dinâmica: ocorre em tempo de execução, não há reserva de memória na inicialização do programa;
 - Quando há necessidade, a memória é alocada e posteriormente liberada;

Estruturas encadeadas

- São estruturas dinâmicas (a memória necessária é alocada durante a execução do programa);
- Devem ser utilizadas quando a organização física não necessariamente coincide com a organização lógica;
- No estudo das estruturas “pilhas” e “filas” a organização física contínua favorece a organização lógica, mas não é a única alternativa;

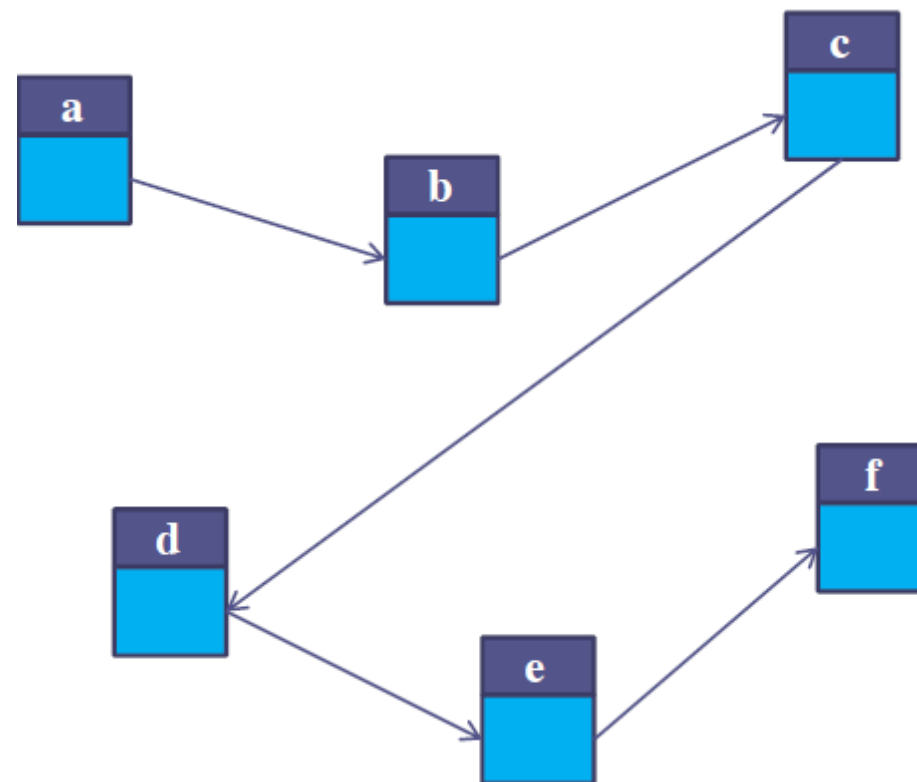
Estruturas encadeadas

- Como funcionam?
- É necessário definir um tipo estruturado com um “campo” ou atributo para fazer a ligação entre elementos de mesmo tipo;

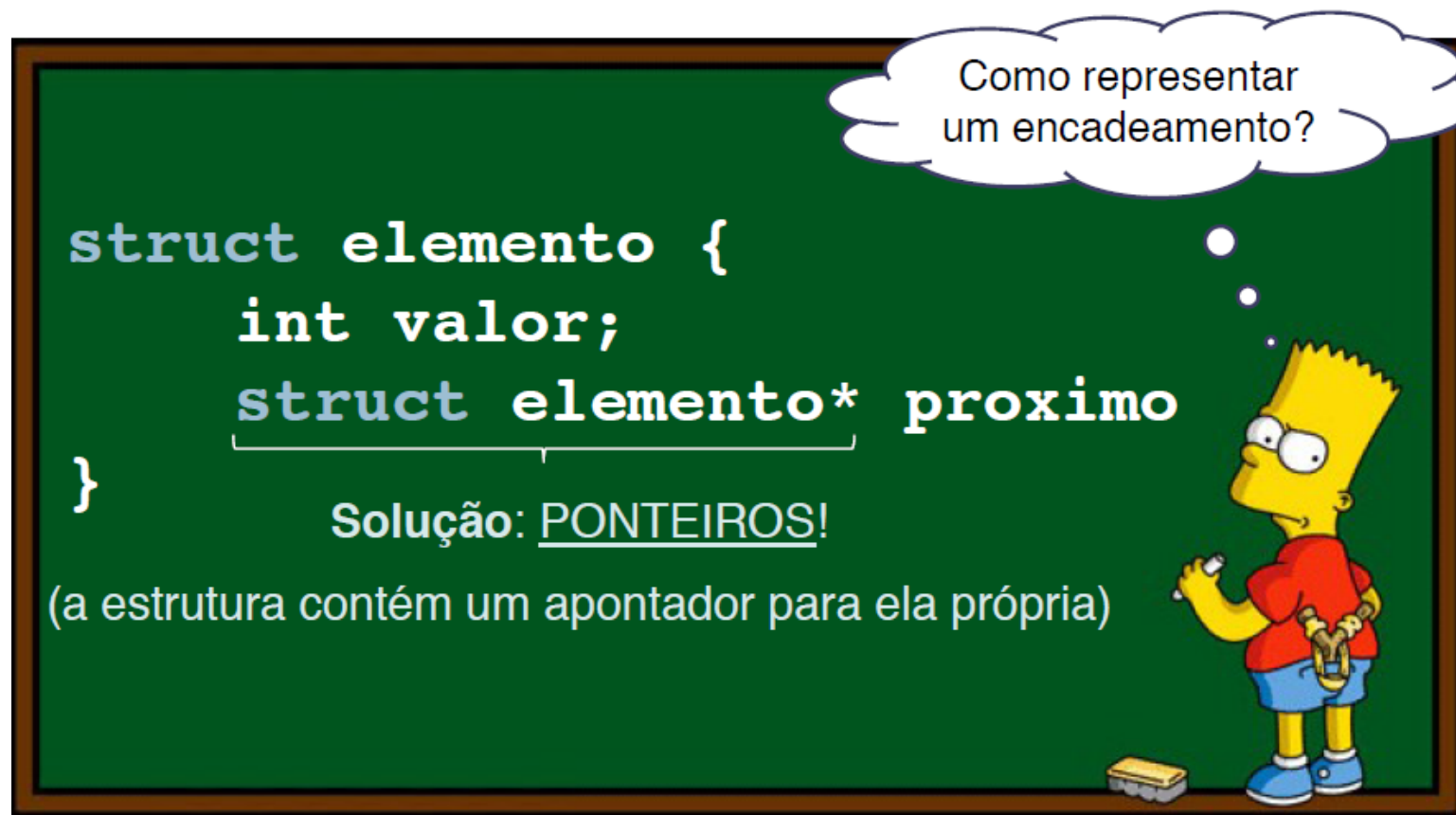


Estruturas encadeadas

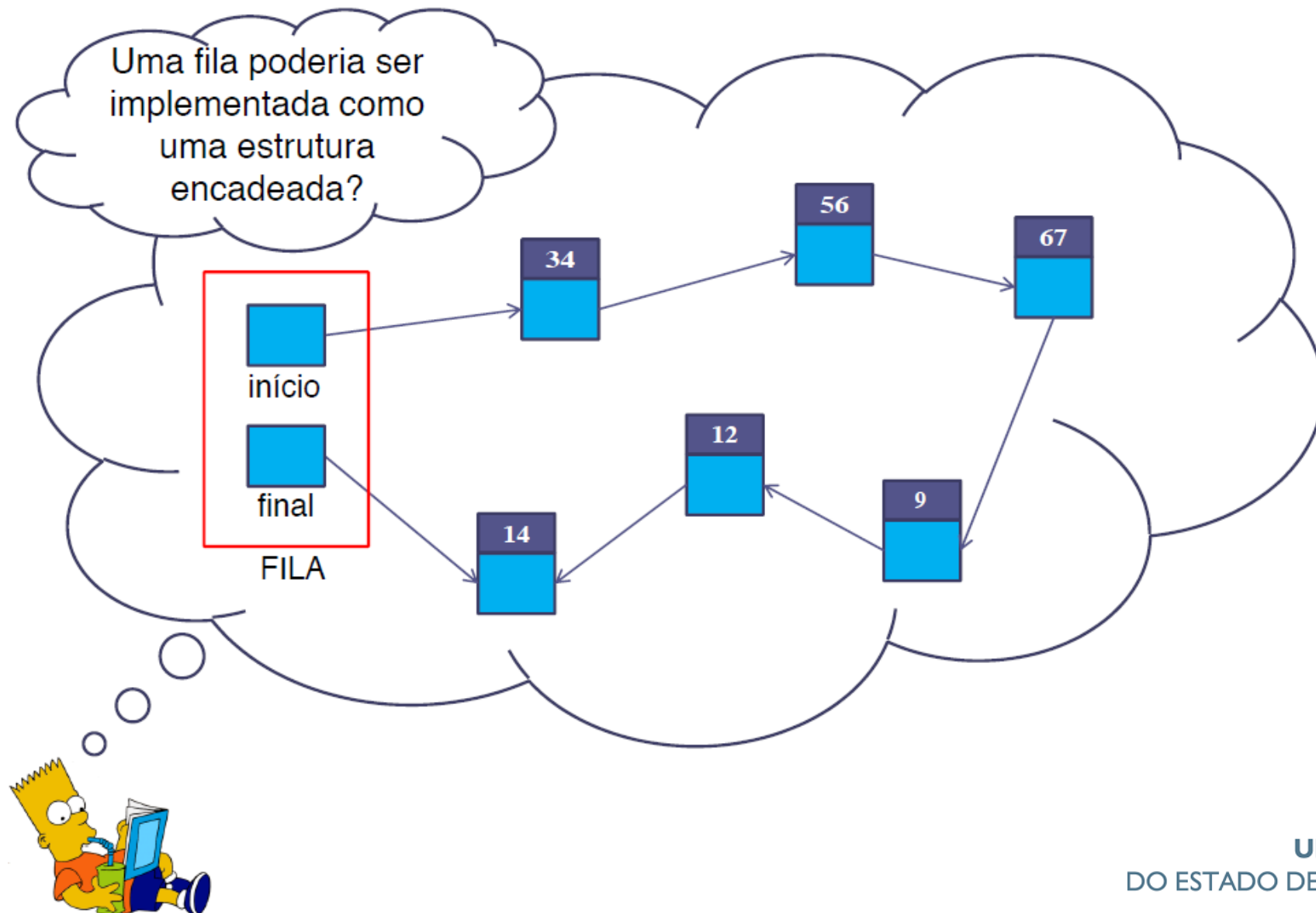
- Cada elemento de uma estrutura encadeada tem dois blocos:



Estruturas encadeadas



Estruturas encadeadas



Pilha Dinâmica

Pilha Dinâmica

- Definição conceitual – TAD Pilha:

```
typedef struct elemento{  
    int valor;  
    struct elemento *anterior;  
}Elemento;
```

```
typedef struct pilha{  
    Elemento *topo;  
}Pilha;
```

Pilha Dinâmica

- Definição das operações do TAD pilha:

```
Pilha* criaPilha();
```

```
void liberaPilha(Pilha* p);
```

```
int empilha(Pilha* p, int v);
```

```
int desempilha(Pilha* p, int* v);
```

```
int estahVazia(Pilha* p);
```

```
int estahCheia(Pilha* p);
```

As operações são
exatamente as
mesmas?



Pilha Dinâmica

- Implementação e utilização das operações:

```
a = criaPilha();


empilha(a, 10);
empilha(a, 20);
empilha(a, 30);

int x;
desempilha(a, &x);
printf("Elemento '%d' retirado", x);

liberaPilha(a);
```

Mas isso é exatamente como já fizemos antes?

A construção de um TAD abstrai do usuário final a forma como o mesmo foi implementado. Basta conhecer a interface do TAD para utilizá-lo.



Pilha Dinâmica

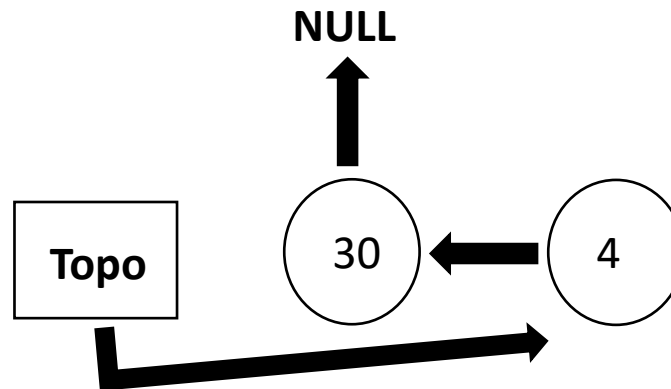
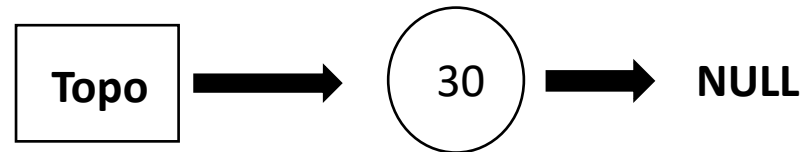
- **Pilha* criaPilha():**
 - Aloca memória para a estrutura física;
 - Inicializa o controle do topo da pilha;
 - Retorna um ponteiro para a estrutura criada;
- **void liberaPilha(Pilha *p):**
 - Recebe um ponteiro para uma estrutura do tipo pilha e libera a memória ocupada por ela (será necessário percorrer todos os elementos da pilha para liberar individualmente seus elementos);

Pilha Dinâmica

- **int estahVazia(Pilha *p):**
 - O que caracteriza uma pilha vazia?
 - O topo não aponta para lugar nenhum;
- **int estahCheia(Pilha *p):**
 - O que caracteriza uma pilha cheia?
 - Com alocação dinâmica a única chance da pilha ficar cheia é se ela ocupar toda a memória do computador;

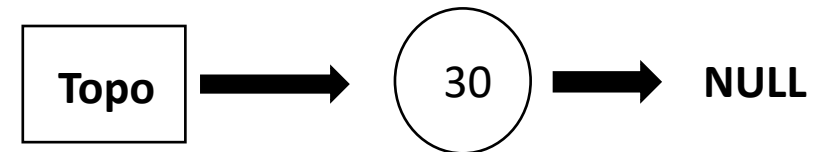
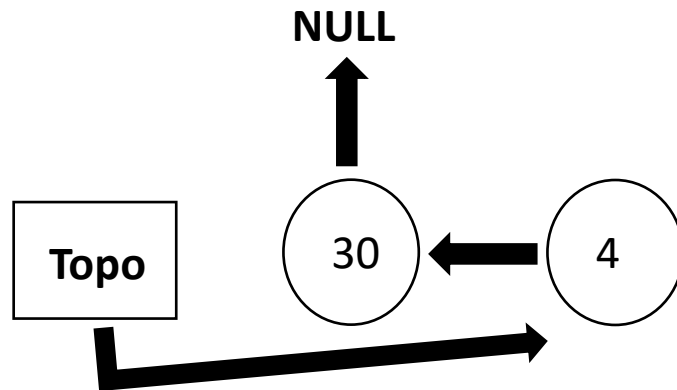
Pilha Dinâmica

- **void empilhar(Pilha *p, int valor):**



Pilha Dinâmica

- **void desempilhar(Pilha *p, int *valor):**



Fila Dinâmica

Fila Dinâmica

- Definição conceitual – TAD Fila:

```
typedef struct elemento{  
    int valor;  
    struct elemento *proximo;  
}Elemento;
```

```
typedef struct fila{  
    Elemento *inicio;  
    Elemento *final;  
}Fila;
```

Fila Dinâmica

- Definição das operações do TAD fila:

```
Fila* criaFila();
```

```
void liberaFila(Fila* p);
```

```
int inserir(Fila* p, int v);
```

```
int retirar(Fila* p, int* v);
```

```
int estahVazia(Fila* p);
```

```
int estahCheia(Fila* p);
```

As operações são
exatamente as
mesmas?



Fila Dinâmica

- Implementação e utilização das operações:

```
a = criaFila();


inserir(a,10);
inserir(a,20);
inserir(a,30);

int x;
retirar(a, &x);
printf("Elemento '%d' retirado",x);

liberaFila(a);
```

Mas isso é exatamente como já fizemos antes?

A construção de um TAD abstrai do usuário final a forma como o mesmo foi implementado. Basta conhecer a interface do TAD para utilizá-lo.



Fila Dinâmica

- **Fila* criaFila():**

- Aloca memória para a estrutura física;
- Inicializa os controles de início e de fim da fila;
- Retorna um ponteiro para a estrutura criada;

- **void liberaFila(Fila *p):**

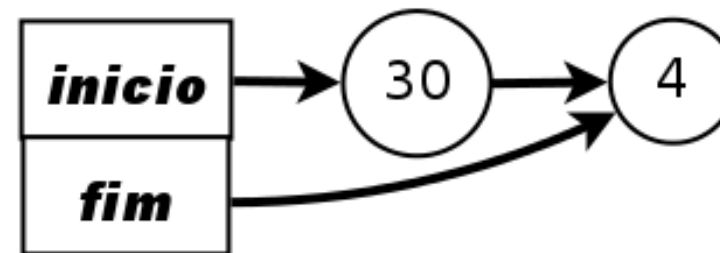
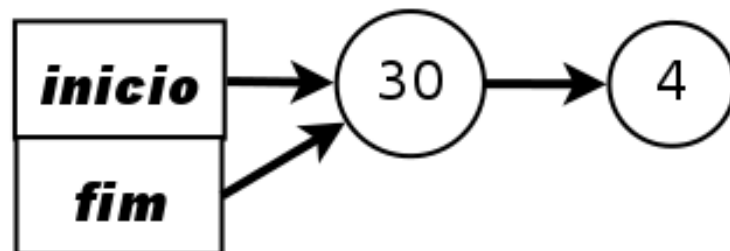
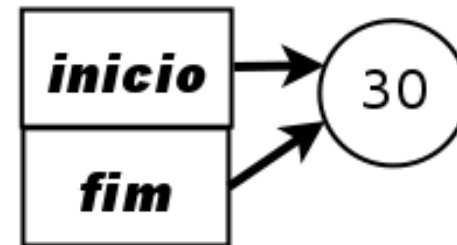
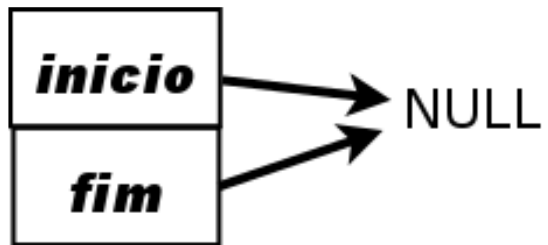
- Recebe um ponteiro para uma estrutura do tipo fila e libera a memória ocupada por ela (será necessário percorrer todos os elementos da fila para liberar individualmente seus elementos);

Fila Dinâmica

- **int estahVazia(Fila *p):**
 - O que caracteriza uma fila vazia?
 - O início e o fim da fila não apontam para nenhum elemento;
- **int estahCheia(Fila *p):**
 - O que caracteriza uma fila cheia?
 - Com alocação dinâmica a única chance da fila ficar cheia é se ela ocupar toda a memória do computador;

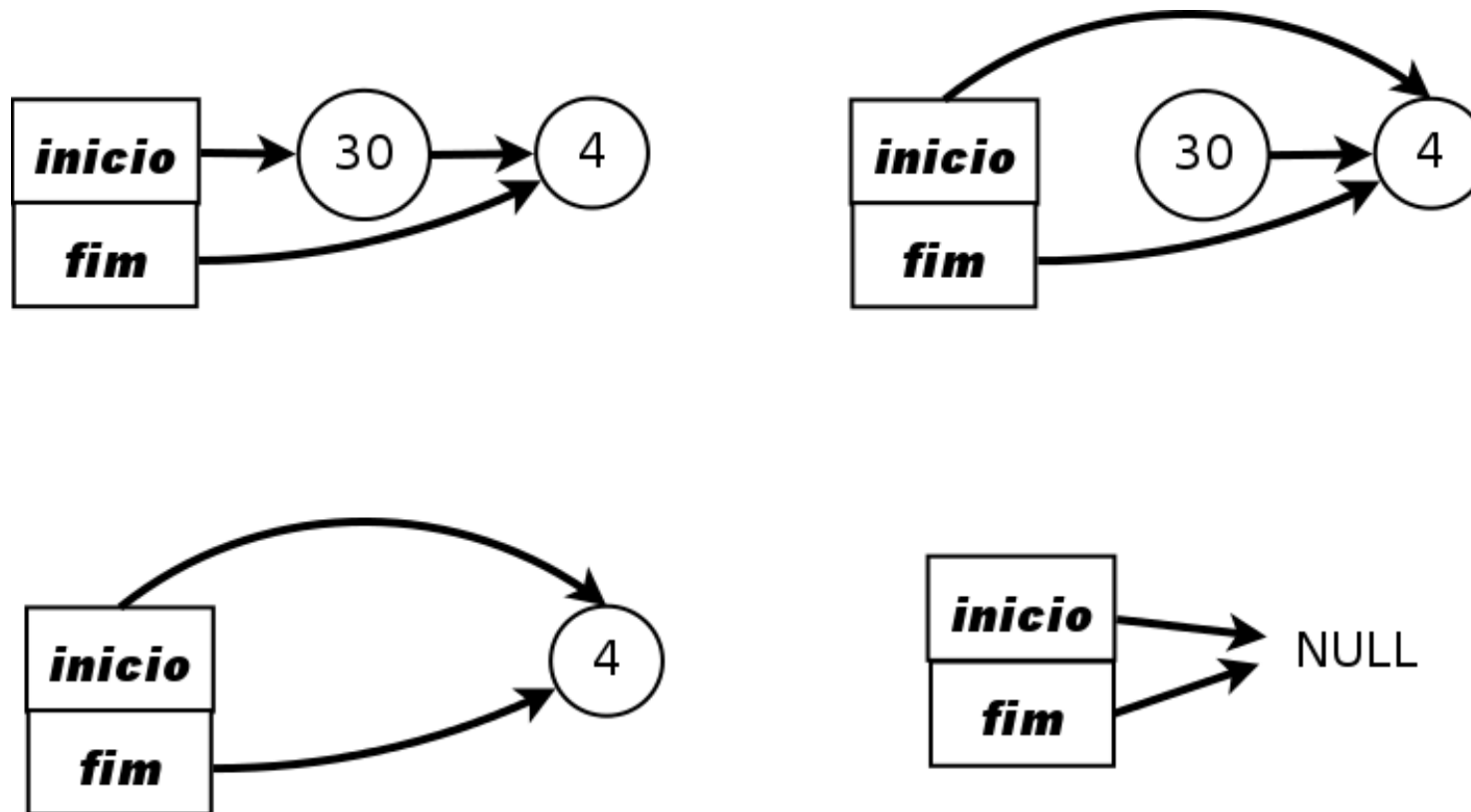
Fila Dinâmica

- `void enfileirar(Fila *p, int valor):`



Fila Dinâmica

- `void desenfileirar(Fila *p, int *valor):`



Algoritmos e Estruturas de Dados II

- Bibliografia:

- Básica:

- CORMEN, Thomas, RIVEST, Ronald, STEIN, Clifford, LEISERSON, Charles. Algoritmos. Rio de Janeiro: Elsevier, 2002.
 - EDELWEISS, Nina, GALANTE, Renata. Estruturas de dados. Porto Alegre: Bookman. 2009. (Série livros didáticos informática UFRGS,18).
 - ZIVIANI, Nívio. Projeto de algoritmos com implementação em Pascal e C. São Paulo: Cengage Learning, 2010.

- Complementar:

- ASCENCIO, Ana C. G. Estrutura de dados. São Paulo: Pearson, 2011. ISBN: 9788576058816.
 - PINTO, W.S. Introdução ao desenvolvimento de algoritmos e estrutura de dados. São Paulo: Érica, 1990.
 - PREISS, Bruno. Estruturas de dados e algoritmos. Rio de Janeiro: Campus, 2000.
 - TENEMBAUM. Aaron M. Estruturas de dados usando C. São Paulo: Makron Books. 1995. 884 p. ISBN: 8534603480.
 - VELOSO, Paulo A. S. Complexidade de algoritmos: análise, projeto e métodos. Porto Alegre, RS: Sagra Luzzatto, 2001

Algoritmos e Estruturas de Dados II

