

# SCC-223 Estruturas de Dados I

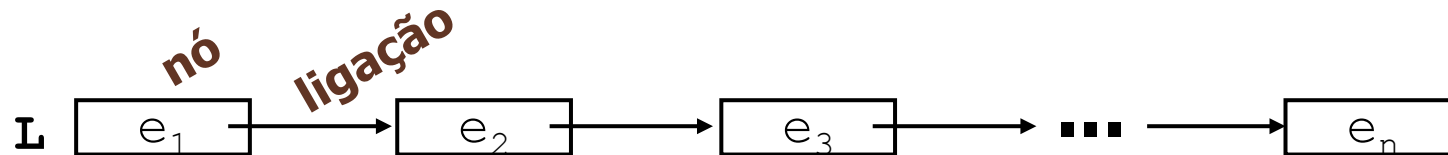
## Lista Encadeada (Circular e Duplamente Encadeada )

Profa. Elaine Parros Machado de Sousa

# Relembrando...

- **Lista Encadeada Dinâmica**  $\Rightarrow$  definida como uma sequência (lógica) de **nós encadeados** e ligados por **ponteiros**

## ORGANIZAÇÃO LÓGICA



```
struct no_{  
    ITEM item;  
    NO *proximo;  
};
```

```
struct lista_{  
    NO *inicio;  
};
```

# Lista Encadeada Dinâmica

- Operações de **Inserção** e **Remoção**
  - **início** e **fim** de lista são tratados como **casos especiais**
  - **testes** para verificar início e fim de lista
- Operações de **Busca** (também utilizada em inserções e remoções)
  - **teste** de fim de lista dentro de *loop*  
ex: `while (p != NULL && p->item != chave)`
- Lista percorrida em uma **única direção**
  - uso de **ponteiros auxiliares** para inserção e remoção no meio da lista

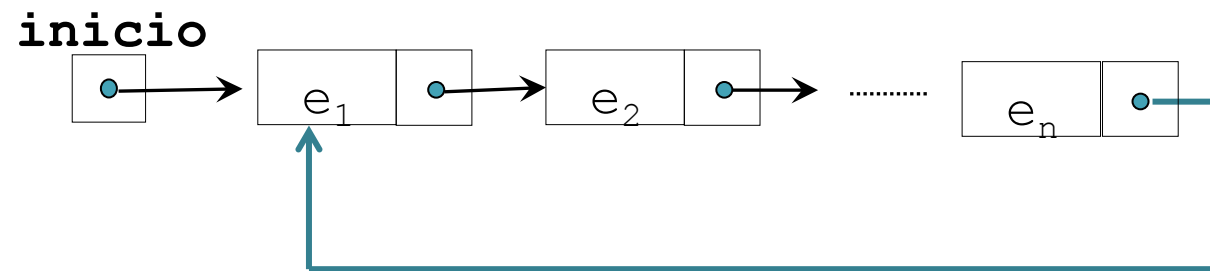
# Lista Encadeada Dinâmica

- Operações de **Inserção e Remoção**
  - **início** e **fim** de lista são tratados como **casos especiais**
  - **testes** para verificar início e fim de lista
- Operações de **Busca** (também utilizada em inserções e remoções)
  - **teste** de fim de lista dentro de *loop*  
`ex: while(p != NULL && p->item != chave)`
- Lista percorrida em uma **única direção**
  - uso de **ponteiros auxiliares** para inserção e remoção no meio da lista

- Alternativas para otimizar esses aspectos:
- ✓ Listas circulares
  - ✓ Uso de nó cabeça
  - ✓ Listas duplamente encadeadas

# Listas Encadeadas Circulares

- **LISTA CIRCULAR**  $\Rightarrow$  *o próximo do último nó é o primeiro nó da lista*



- A partir de um nó da lista pode-se chegar a qualquer outro nó!!!!

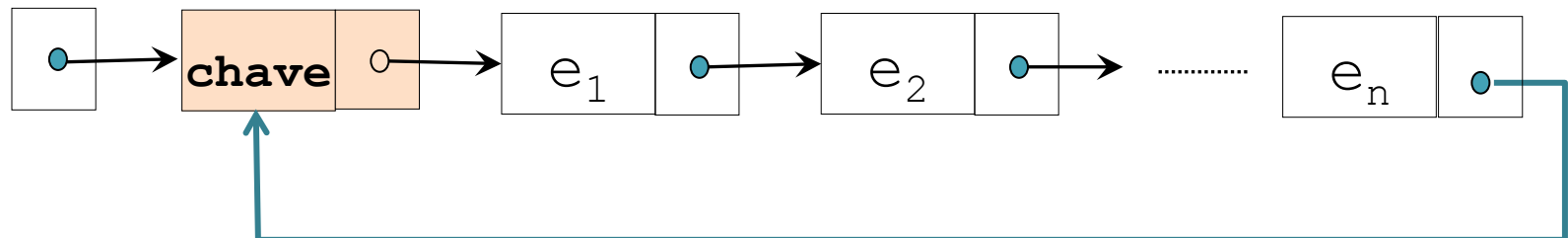
## Listas Encadeadas Circulares – Sentinela

- **Busca** em listas circulares  $\Rightarrow$  o uso de um **nó cabeça** pode reduzir a quantidade de testes necessários
  - **Ex:** `while(p != NULL && p->item != chave)`
  - **não é necessário** testar se a lista acabou
- Nó cabeça em listas circulares é chamado de **SENTINELA**

# Listas Encadeadas Circulares – Sentinela

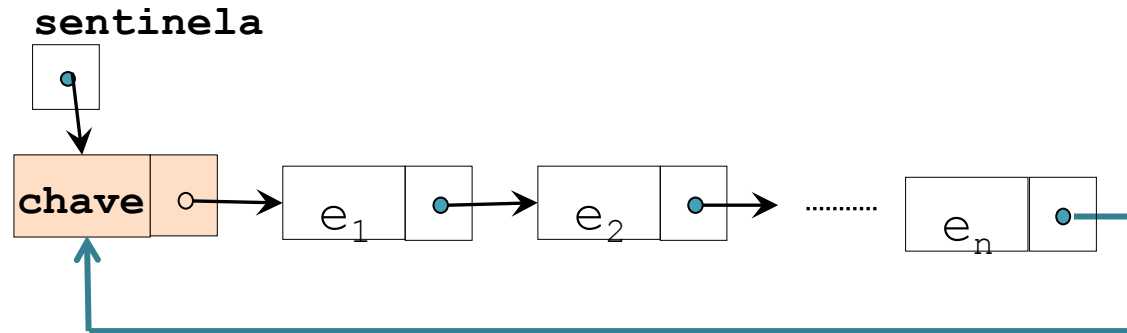
- Ideia principal:
  - chave de busca é colocada no nó cabeça
  - busca começa no próximo nó
  - elemento será sempre encontrado
    - se o nó encontrado for a cabeça, a busca não teve sucesso.

**sentinela**



# Listas Encadeadas Circulares - Sentinela

```
struct lista_{  
    NO *sentinela;  
};
```




```
int lista_busca(int chave, LISTA *lista); {  
    lista->sentinela->item = chave;  
    NO *p = lista->sentinela;  
  
    do {  
        p = p->proximo;  
    } while (p->item) != chave);  
  
    return ((p != lista->sentinela) ? p->item : ERRO);  
}
```



# Listas Encadeadas Circulares - Sentinela

- Qual o ganho em eficiência, em relação à Busca em Lista Linear Encadeada?
- A complexidade (O) da Busca muda?

```
int lista_busca(int chave, LISTA *lista); {  
    lista->sentinela->item = chave;  
    NO *p = lista->sentinela;  
  
    do {  
        p = p->proximo;  
    } while (p->item) != chave);   
  
    return ((p != lista->sentinela) ? p->item : ERRO);  
}
```



# Listas Encadeadas Circulares – Sentinela

- Principais alterações na implementação das funções do TAD Lista
  - Criação da lista => alocar nó cabeça/sentinela
  - Inserção e remoção => não é necessário testes específicos para primeiro e último elemento
  - Busca
- Nó cabeça pode ser usado na lista linear (não circular)
  - Inserção/remoção é “igual” para nós em qq posição da lista

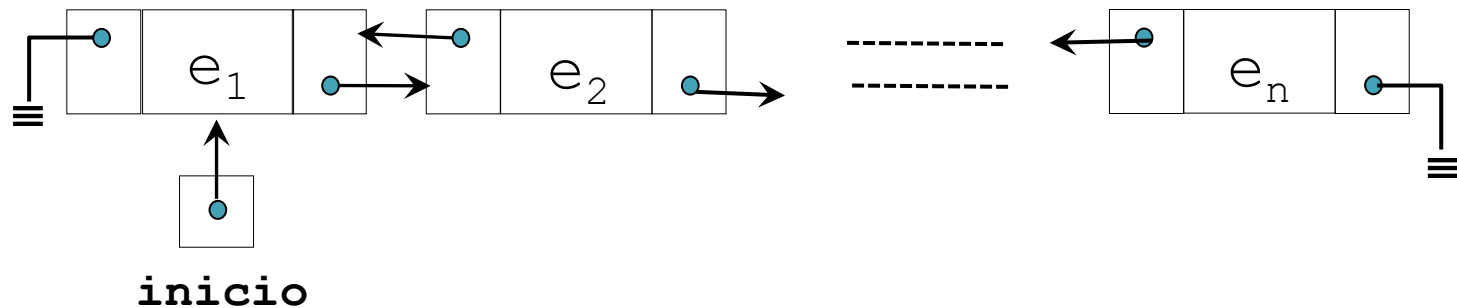


# Listas Encadeadas Circulares – Sentinela

- Exemplos de aplicação?

# Listas Duplamente Encadeadas

- **Encadeamento duplo**  $\Rightarrow$  cada nó mantém um ponteiro para o nó anterior e um ponteiro para o nó posterior





# Listas Duplamente Encadeadas

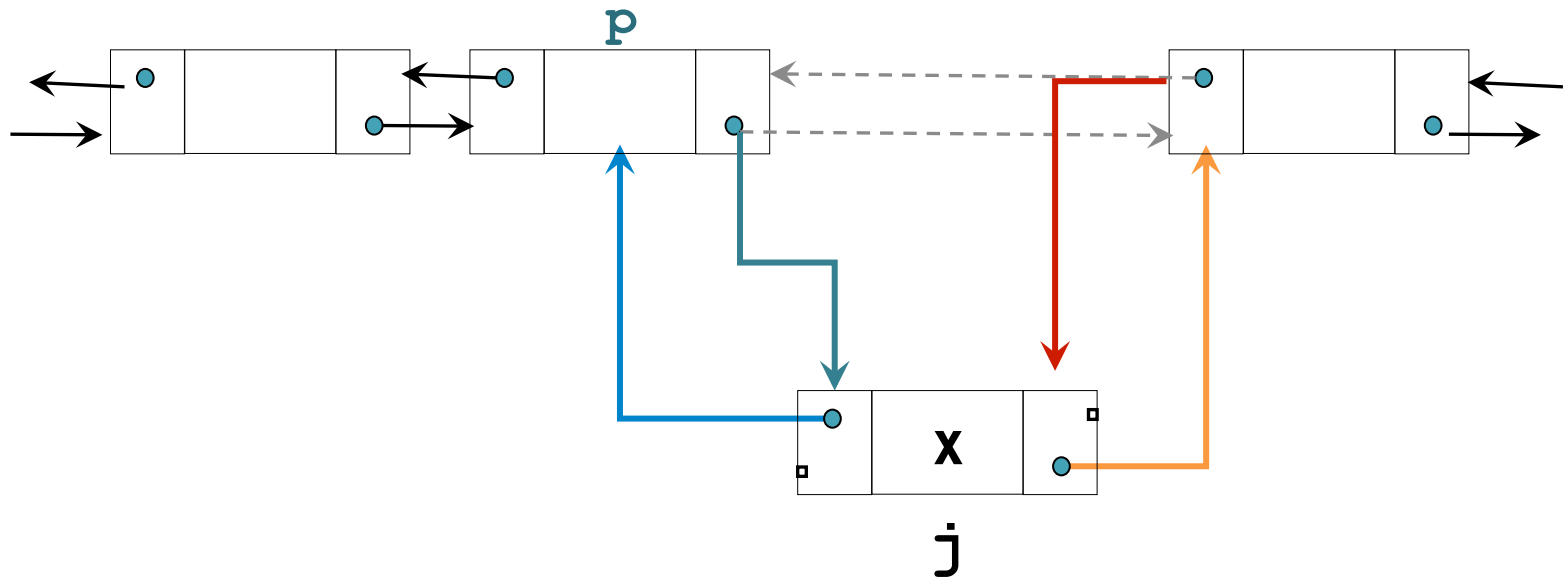
- A manipulação da lista é mais complexa, porém algumas operações são diretamente beneficiadas:
  - operações de inserção e remoção no interior da lista sem necessidade de ponteiro auxiliar
  - percurso em qualquer direção
- Encadeamento duplo pode ser implementado para variações de lista: com nó cabeça, sem nó cabeça, circular com nó cabeça, circular sem nó cabeça, etc.

-

# Listas Duplamente Encadeadas

## ➤ Inserção após o nó **p**

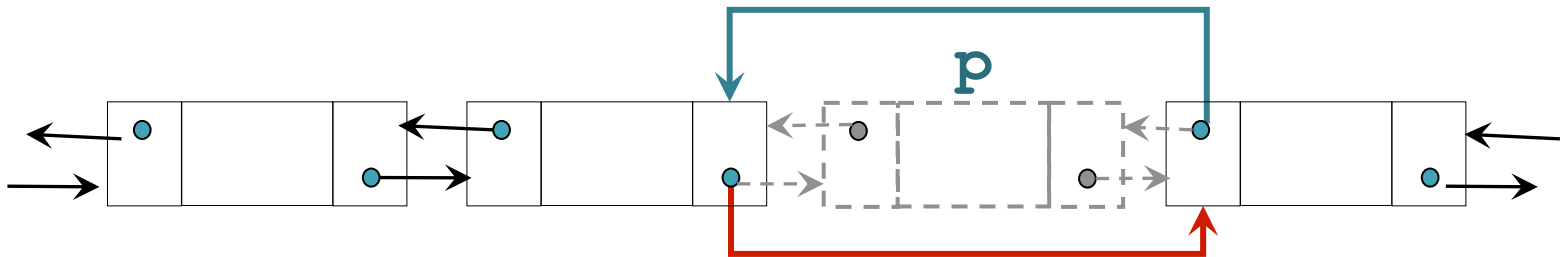
```
j->item = x;  
j->anterior = p;  
j->proximo = p->proximo;  
p->proximo->anterior = j;  
p->proximo = j;
```



# Listas Duplamente Encadeadas

## ➤ Eliminação do nó **p**

```
p->proximo->anterior = p->anterior;  
p->anterior->proximo = p->proximo;  
delete (p) ;
```



# Listas Duplamente Encadeadas - TAD

```
1  /*lista duplamente encadeada*/
2
3  struct no_{
4      ITEM item;
5      NO *anterior;
6      NO *proximo;
7  };
8
9  struct lista_{
10     NO *inicio;
11 } ;  /*tamanho da lista*
...

```



# Listas Duplamente Encadeadas - TAD

```
/*Insere um novo nó no início da lista. LISTAS NÃO ORDENADAS*/
1 boolean lista_inserir(LISTA *lista, ITEM i){
2     if (lista != NULL) {
3         NO *pnovo = (NO *) malloc(sizeof (NO));
4         pnovo->item = i;
5         if (lista->inicio == NULL){
6             lista->inicio = pnovo;
7             pnovo->proximo = NULL;
8     }
9         else {
10            lista->inicio->anterior = pnovo;
11            pnovo->proximo = lista->inicio;
12        }
13        pnovo->anterior = NULL;
14        lista->inicio = pnovo;
15
16        return (TRUE);
17    } else
18    return (FALSE);
19 }
```

# Listas Duplamente Encadeadas - TAD

```
1 boolean lista_remove(LISTA *lista, int chave){
2     NO *p=NULL;
3     if ( (lista != NULL) && (!lista_vazia(lista)) ){
4         p = lista->inicio;
5         while(p != NULL && p->item != chave) )
6             p = p->proximo;
7         if(p != NULL){ /*Lista não acabou -> encontrou a chave*/
8             if(p == lista->inicio)
9                 lista->inicio = p->proximo;
10            else
11                p->anterior->proximo = p->proximo;
12            p->proximo->anterior = p->anterior;
13            p->proximo = NULL;
14            p->anterior = NULL;
15            free(p);
16            return(TRUE);
17 }
18 }
19 return(FALSE); /*chave não está na lista ou lista vazia*/
20 }
```



# Listas Duplamente Encadeadas

➤ Exemplos de aplicação?



## Exercício – não é preciso entregar

- Implementar o TAD Lista Não Ordenada com:
  - encadeamento duplo
  - circular com sentinela