INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CATARINENSE
Campus Videira

Bacharelado em Ciência da Computação

Alocação Encadeada Dupla

Professor: Manassés Ribeiro
manasses.ribeiro@ifc.edu.br

Conceito

Relembrando ...

- As listas são mais comumente implementadas utilizando alocação dinâmica de memória;
- Na alocação encadeada, a ordem lógica das informações pode ser (e geralmente é) diferente da ordem física.

Conceito

- Na alocação encadeada dupla, o **nodo** (elemento) deve conter pelo menos três informações:
  - **INFO;**
  - **ELO Posterior;** e
  - **ELO Anterior**.

Conceito

- **INFO:** campo que contém as informações que são armazenadas na lista;
- **ELO Posterior:** campo que faz o encadeamento das informações com o **próximo elemento** da lista;
- **ELO Anterior:** campo que faz o encadeamento das informações com o **elemento anterior** da lista.

Conceito

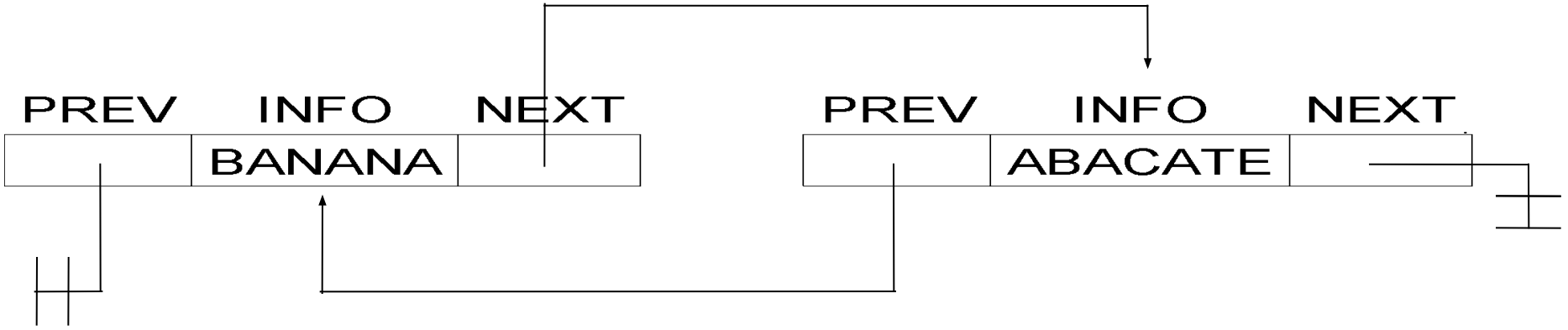
- O campo INFO, por sua vez, pode ser dividido em *n* outros campos (ou pode ser um ponteiro para um TAD (Tipo Abstrato de Dados)
- O campo ELO, anterior e posterior, conterá a referência para os nodos anterior e posterior, respectivamente, ao que o nodo atual está ligado em sua ordem lógica.

Alocação Dinâmica de Memória (ADM)

Relembrando ...

- É o processo que aloca (solicita/**reserva**) memória em tempo de execução;
- É utilizado quando **não se sabe ao certo quanto de memória será necessário** para realizar determinada operação;
- Essencial para **evitar o desperdício** de memória.

Lista Encadeada Dupla com ADM



Lista Encadeada Dupla com ADM

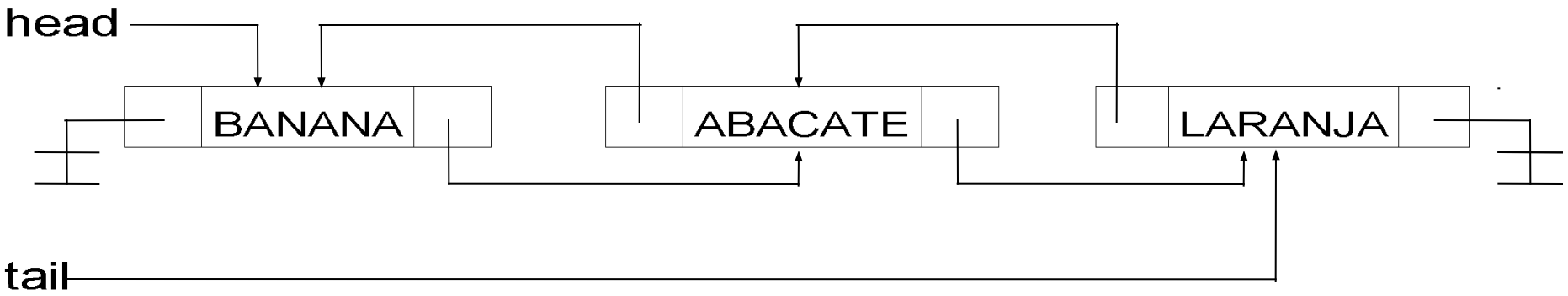
Usando esta estrutura de três membros, uma lista encadeada é formada definindo-se o ponteiro **next** de cada elemento para que ele aponte para o elemento que segue e o ponteiro **prev** para o elemento que o antecede.

Lista Encadeada Dupla com ADM

- O elemento **next** do último elemento é definido para NULL, bem como o elemento **prev** do primeiro elemento também é apontado para NULL;
- Assim como na lista encadeada simples, o elemento no início da lista é a cabeça (**head**) e o elemento no final da lista é a cauda (**tail**).

Lista Encadeada Dupla com ADM

- Exemplo:



Lista Encadeada Dupla com ADM

- Para acessar um elemento em uma lista encadeada dupla começa-se pela cabeça da lista e usando os ponteiros ***next*** de elementos sucessivos para mover de elemento a elemento até que o elemento desejado seja encontrado.
- Da mesma forma é possível retornar utilizando o ponteiro ***prev***.

Lista Encadeada Dupla com ADM

- Outra característica importante da lista encadeada dupla é a possibilidade de percorrê-la tanto da cabeça para a cauda, como da cauda para a cabeça.
- Esta característica possibilita a implementação de estruturas do tipo Pilha, por exemplo (**veremos nas próximas aulas**).

Lista Encadeada Dupla com ADM

Relembrando ...

- As principais funções em listas são:
  - Cria lista;
  - Cria elemento;
  - Insere elemento na lista;
  - Remove elemento da lista;
  - Percorre a lista (busca elementos).

Estrutura da Lista

Relembrando ...

- A estrutura principal da lista encadeada, utilizando ADM, é composta por:
  - Elemento head;
  - Elemento tail;
  - Tamanho da lista.

Algoritmo da função insere na LED

1. Cria novo elemento
2. Aloca memória para novo elemento
3. Atribui valor ao novo elemento
4. Verifica se elemento pivô é NULL E se lista não vazia, caso verdadeiro
    4.1 Retorna um sinal indicando que só aceita elemento pivô como NULL na inserção do primeiro elemento da lista e encerra o algoritmo.
5. Verifica se a lista está vazia, caso verdadeiro
    5.1 Insere o novo elemento como primeiro elemento da lista
6. Caso o item 5 seja falso, ou seja caso a lista não vazia
    6.1 Seta os ponteiros *next* e *prev* do novo elemento no meio da lista logo após o elemento pivô
    6.2 Verifica se o novo elemento inserido é o último elemento da lista, caso verdadeiro
        6.2.1 Seta o ponteiro *tail* para o novo elemento inserido
    6.3 Caso o item 6.2 seja falso
        6.3.1 Seta o ponteiro *prev* do elemento posterior ao elemento pivô para o novo elemento.
    6.4 Seta o ponteiro *next* do elemento pivô para o novo elemento.
7. Atualiza o tamanho da lista.

Algoritmo da função insere na LED

```
Insercao (lista, pivo, dado)
inicio
   Cria (novo_elemento)
   Aloca_memória (novo_elemento)
   novo_elemento->dado = dado

   se ((pivo==NULL) E (lista não vazia))
        retorna erro indicando que só
        aceita pivo NULL na inserção do
        primeiro elemento
   fimse

   se (lista vazia)
   |      lista->head  = novo_elemento
   |      lista->tail = novo_elemento
   |
   |
   |
   senao
         novo_elemento->next = pivo->next
         novo_elemento->prev = pivo

         se (pivo->next == NULL)
               lista->tail = novo_elemento
         senao
               pivo->next->prev = novo_elemento
         fimse

         pivo->next = novo_elemento
   fimse

   atualiza_tamanho_lista
fim
```

Algoritmo da função remove na LED

1. Verifica se o elemento a ser excluído é diferente de NULL e se a lista não esta vazia, caso verdadeiro
    1.1 Verifica se o elemento a ser excluído é o *head* da lista, caso verdadeiro
        1.1.1 Atribui o próximo elemento após o elemento a ser excluído como *head* da lista
        1.1.2 Verifica se o head da lista é NULL, caso verdadeiro
            1.1.2.1 Seta a lista como vazia
        1.1.3 Caso o item 1.1.2 seja falso
            1.1.3.1 Seta o ponteiro *prev* do *head* da lista com NULL
    1.2 Caso o item 1.1 seja falso
        1.2.1 Seta o ponteiro *next* do elemento anterior ao elemento a ser excluído para o elemento posterior ao ser excluído
        1.2.2 Verifica se o elemento a ser excluído está no fim da lista, caso verdadeiro
            1.2.2.1 Seta o ponteiro *tail* da lista para o elemento anterior ao elemento a ser excluído
        1.2.3 Caso o item 1.2.2 seja falso
            1.2.3.1 Seta o ponteiro *prev* do elemento posterior ao elemento a ser excluído para o anterior
    1.3 Destrói o elemento a ser excluído
    1.4 Atualiza o tamanho da lista

Algoritmo da função remove

```
remocao (elemento, lista)
inicio
   se ((elemento != NULL) e (lista não vazia))
        se (elemento == lista->head)
              lista->head = elemento->next
              se (lista->head == NULL)
                    lista->tail = NULL
              senao
                    elemento->next->prev = NULL
              fimse
        senao
              elemento->prev->next = elemento->next
              se (elemento->next == NULL)
                    lista->tail = elemento->prev
              senao
                    elemento->next->prev = elemento->prev
              fimse
        fimse
              destroi(elemento)
              atualiza_tamanho_lista
    fimse
fim
```

Principais funções

- **Cria lista:** função que cria a lista e aloca a memória necessária para a lista;
- **Cria elemento:** função que cria um elemento (nodo) novo, aloca memória necessária e atribui um valor para este novo elemento;
- **Insere elemento na lista:** função que insere este novo elemento em uma lista seguindo o algoritmo apresentado nos slides 15 e 16;
- **Remove elemento da lista:** função que insere este novo elemento em uma lista seguindo o algoritmo apresentado nos slides 17 e 18;
- **Percorre a lista para encontrar elementos (pesquisa):** função que percorre a lista, a partir da cabeça (head) até encontrar o elemento que está sendo buscado ou atingir o fim da lista, e também no sentido oposto (sentido cauda-cabeça), uma vez que a lista é dupla.

Representação em C

Representação de Elementos em LED usando C

```
typedef struct sElemento{
    struct sElemento *next;
    struct sElemento *prev;
    int dado; //Depende do tipo de dados que se deseja trabalhar
} Elemento;
```

Representação em C

Representação da LED em C

```
typedef struct sLista{
      struct sElemento *head;
      struct sElemento *tail;
      int size;
} Lista;
```