

ESTRUTURA DE DADOS

LISTAS DUPLAMENTE ENCADEADAS

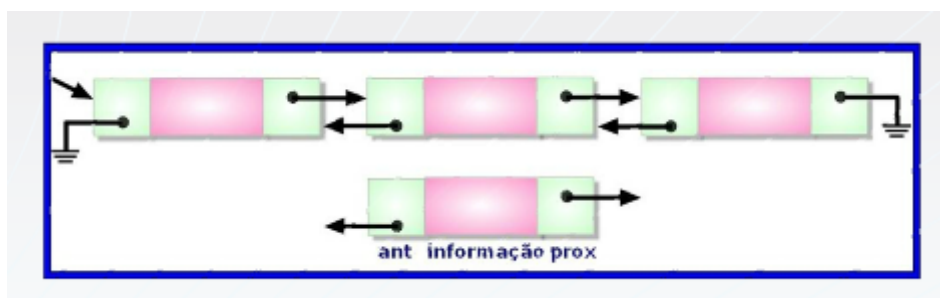
Olá!

Ao final desta aula, você será capaz de:

1. Compreender e implementar operações com Listas Duplamente Encadeadas.

1 Tema: Duplamente encadear para garantir e flexibilizar

As Listas Duplamente Encadeadas são compostas de nós ligados por dois ponteiros.



Pela figura, podemos observar que o ponteiro ant aponta para o nó que o precede. Enquanto que o ponteiro prox, aponta para o nó que o sucede. Sendo assim, se desejarmos varrer a lista de trás para frente, faremos uso do ponteiro ant caso contrário, usaremos o ponteiro prox.

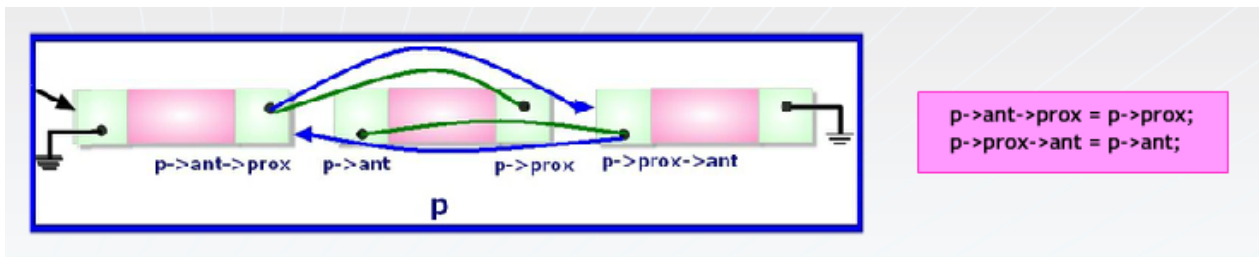
O fato de existir um ponteiro a mais em cada nodo, implicando em um novo membro implica em um uso maior de memória falarão alguns, mas isso é recompensado porque torna o processamento da Lista Duplamente Encadeada mais flexível do que a Lista Simplesmente Encadeada.

Uma das maiores dificuldades na Listas Simplesmente Encadeada era a remoção de um nó porque, como não tínhamos como acessar o nó anterior, era necessário percorrer a lista toda para localizá-lo.

Com dois ponteiros, podemos, facilmente, sabendo a localização do nó que queremos remover, acessar o nó anterior e o posterior e, dessa forma, atualizar a Lista, após a remoção como podemos constatar na função remover do programa que será mostrado.

Mas é bom reforçar que, para sabermos a localização do nó a ser removido, precisaremos fazer uma busca na lista, implicando na criação de uma função de busca sequencial.

Vamos analisar como seria a remoção de um nó que se encontra no meio através da figura abaixo e de dois comandos que considero de maior dificuldade na função remover.



- 1) Através de seu ponteiro **prox**, **p** apontava para o próximo nó cuja representação é: **p->prox**.
 - 2) Esse endereço foi copiado para o ponteiro **prox** do nó anterior acessado por **p->ant->prox**. (linha verde)
 - 3) Sendo assim, após a remoção de **p**, **p->ant->prox** apontará para o nó seguinte ao que foi removido. (seta azul)
- O mesmo procedimento se repetiu para ajustar o outro ponteiro.

Atenção

Não se esqueça que, embora não seja comum usar, ao invés de usarmos a seta, poderemos usar:

(nomeDoPonteiro).membro

Logo: **P->prox** é equivalente à **(*p).prox**.

Todas as operações realizadas com as Listas Simplesmente Encadeadas poderão ser realizadas com as Duplamente Encadeadas e algumas, com mais facilidade.

Como na aula 8, já falamos sobre alocação dinâmica de Memória, sobre Lista Encadeada, explicamos da necessidade do ponteiro, constato que não temos muito que acrescentar de teoria nesta aula. Sendo assim, vou apresentar nosso último programa, evidentemente com menu como gosto e com algumas funções que considero básicas e que já fizemos com Listas simplesmente Encadeadas.

Como o código é muito grande, clique no link a seguir para visualizá-lo:

http://estaciODOcente.webaula.com.br/cursos/gon119/docs/10ED_doc01.pdf

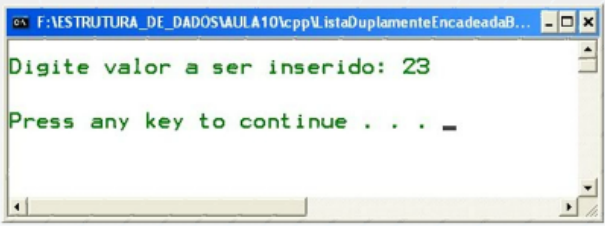
```

F:\ESTRUTURA_DE_DADOS\AULA10\cpp\ListaDuplamenteEncadeadaB...
( ( ) ) Alocação Dinâmica ( ( ) )
( 1- Insere na 1ª posição )
( 2- Remove da Lista DE )
( 3- Exibe a Lista DE )
( 4- Conta Nos da Lista DE )
( 5- Libera a Lista DE )
( 6- Sai )
( Opção: )
( ( ) ) ( ( ) ) ( ( ) ) ( ( ) )

```

Agora clique nos elos abaixo para visualizar os 5 casos do menu abaixo:

CASO 1 DO MENU

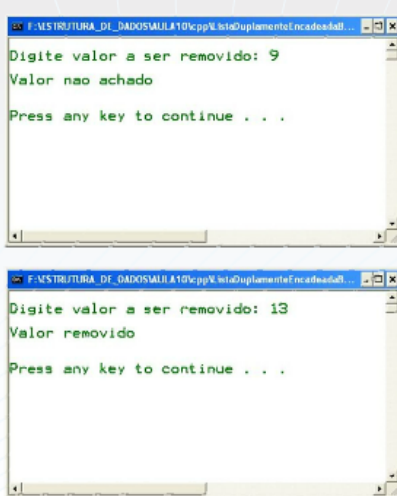


```
cout<<"\nDigite valor a ser inserido: ";
cin>>valor;
lista = insere(lista, valor);
```

```
// insere no início
listaDE *insere(listaDE *LISTA, int valor)
{
    listaDE* novo = new listaDE;
    novo->info = valor;
    novo->prox = LISTA;
    novo->ant = NULL;

    if (LISTA)
    {
        LISTA->ant = novo;
    }
    return novo;
}
```

CASO 2 DO MENU



```
if(!lista)
    cout << "\n\nNada a remover. Lista vazia\n";
else
{
    cout<<"\nDigite valor a ser removido: ";
    cin>>valor;
    lista=remove(lista, valor);
}
```

```
// busca valor na lista
listaDE *busca (listaDE *LISTA, int valor)
{
    listaDE *ptr;
    for (ptr=LISTA; ptr != NULL; ptr=ptr->prox)
        if (ptr->info == valor)
            return ptr;
    return NULL; // nao achou o elemento
}
```

```
// remove um elemento da lista
listaDE *remove(listaDE * LISTA, int valor)
{
    listaDE *p = busca(LISTA, valor) ;

    if (!p)
    {
        cout<< "\nValor nao achado\n";
        return LISTA;
    } // nao achou o elemento

    // retira elemento do encadeamento
    if (LISTA == p)
        LISTA = p->prox;
    else
        p->ant->prox = p->prox;

    if (p->prox )
        p->prox->ant = p->ant;
    cout<<"\nValor removido\n";
    delete p;
    return LISTA;
}
```

Atenção

Esta foi a única função diferente, visto que foram introduzidos dois ponteiros. Entretanto sua análise já foi feita anteriormente.

CASO 3 DO MENU

```
F:\ESTRUTURA_DE_DADOS\ALIA10\cpp\ListaDuplamenteEncadeadaB...
89
13
23
Press any key to continue . . .
```

```
// exibe lista
void exibe(listaDE *LISTA)
{
    listaDE* ptr;
    for (ptr=LISTA; ptr != NULL; ptr=ptr->prox)
        cout<<"\n"<<ptr->info;
}
```

```
if(!lista)
    cout << "\n\nLista vazia\n";
else
    exibe(lista);
```

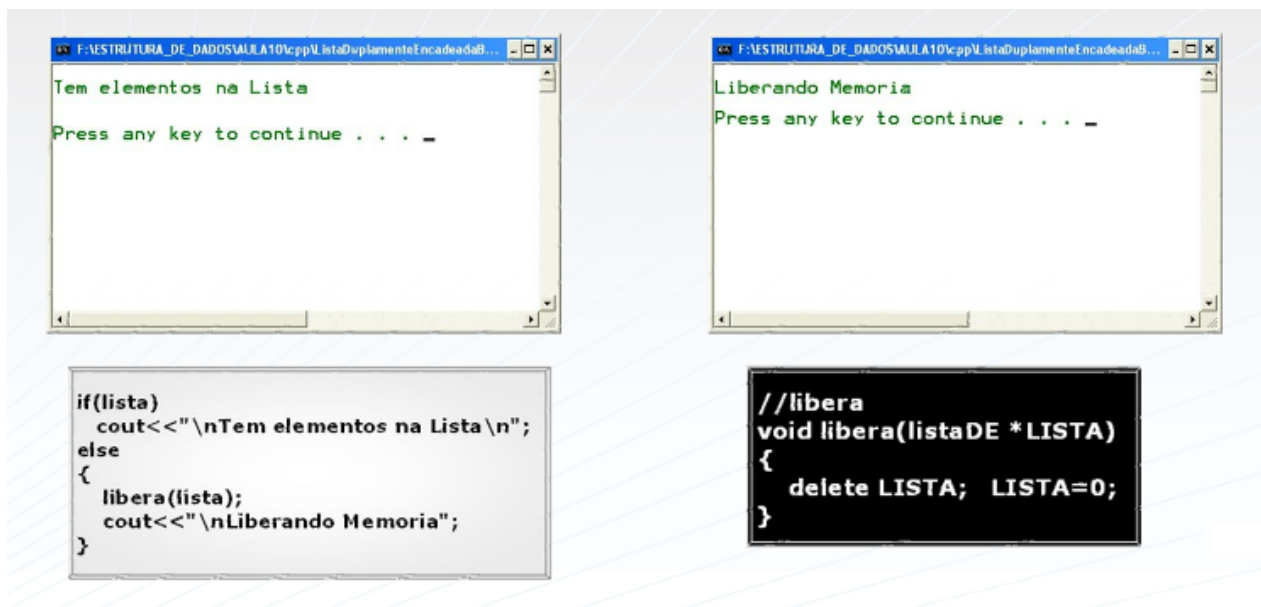
CASO 4 DO MENU

```
F:\ESTRUTURA_DE_DADOS\ALIA10\cpp\ListaDuplamenteEncadeadaB...
Total de nos: 2
Press any key to continue . . .
```

```
// conta nós da Lista
int contaNos(listaDE *LISTA)
{
    int conta = 0;
    while (LISTA != NULL)
    {
        conta++;
        LISTA = LISTA->prox;
    }
    return conta;
}
```

```
if(!lista)
    cout<<"\n\nLista vazia\n";
else
    cout<<"\nTotal de nos:"<< contaNOS(lista);
```

CASO 5 DO MENU



Struct

```
struct listaDE
{
int info;
struct listaDE *ant;
struct listaDE *prox;
};
```

Gostaria de deixar para você um presente. Então pensei: Professor deixa que tipo de presente? Um exercido que dê muito trabalho, é claro, pois o resultado é gratificante.

Deixo então, uma função que estava originalmente neste programa. Gostaria que você analisasse antes de inserir no programa e descobrisse o que ela faz.

Depois, pesquisasse na Internet e pensasse em algumas alterações que poderia fazer nesta função para que ela pudesse ter uma função mais ampla.

Gostaria de deixar para você um presente. Então pensei: Professor deixa que tipo de presente? Um exercido que dê muito trabalho, é claro, pois o resultado é gratificante.

Deixo então, uma função que estava originalmente neste programa. Gostaria que você analisasse antes de inserir no programa e descobrisse o que ela faz.

Depois, pesquisasse na Internet e pensasse em algumas alterações que poderia fazer nesta função para que ela pudesse ter uma função mais ampla.

```
void insere (listaDE *LISTA, int valor)
{
    listaDE *novo= new listaDE;
    novo->info=valor;
    novo->prox=LISTA->prox;
    if (LISTA->prox != NULL)
        novo->prox->ant=novo;
    LISTA->prox = novo;
    novo->ant=LISTA;
}
```

CONCLUSÃO

Nesta aula, você:

- Compreendeu e usou Lista Duplamente Encadeada.