

Lista duplamente encadeada

Murilo Dantas

Lista duplamente encadeada

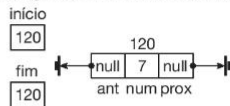
- Definição prática
 - ▶ Contém ponteiros para o “próximo” elemento e para o “anterior”.
 - ▶ Pode tornar o “trânsito” entre os nós mais rápido.

Lista duplamente encadeada

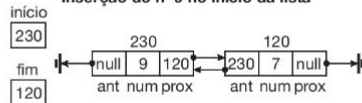
1ª operação
A lista está vazia



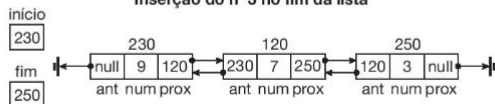
2ª operação
Inserção do nº 7 no início da lista



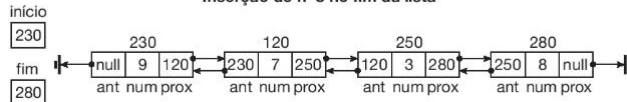
3ª operação
Inserção do nº 9 no início da lista



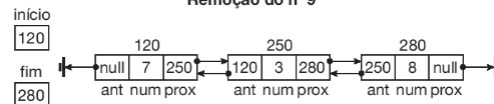
4ª operação
Inserção do nº 3 no fim da lista



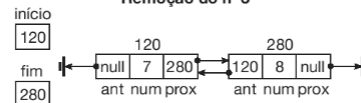
5ª operação
Inserção do nº 8 no fim da lista



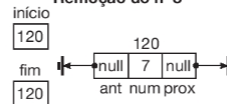
6ª operação
Remoção do nº 9



7ª operação
Remoção do nº 3



8ª operação
Remoção do nº 8




Lista duplamente encadeada

```
#include <iostream.h>
#include <conio.h>

void main()
{
    //Definindo o registro representará
    //cada elemento da lista
    struct LISTA
    {
        int num;
        LISTA *prox;
        LISTA *ant;
    };

    // a lista está vazia, logo,
    // o ponteiro inicio tem o valor NULL
    // o ponteiro inicio conterá o endereço
    // do primeiro elemento da lista
    LISTA *inicio = NULL;
    // o ponteiro fim conterá o endereço
    // do último elemento da lista
    LISTA *fim = NULL;
    // o ponteiro aux é um ponteiro auxiliar
    LISTA *aux;
    // apresentando o menu de opções
    int op, numero, achou;
    do
    {
        clrscr();
        cout<<"\nMENU DE OPÇÕES\n";
        cout<<"\n1 - Inserir no início da
        ↳ lista";
```



```
        cout<<"\n2 - Inserir no fim da lista";
        cout<<"\n3 - Consultar a lista do
        ↳ início ao fim";
        cout<<"\n4 - Consultar a lista do fim
        ↳ ao início";
        cout<<"\n5 - Remover da lista";
        cout<<"\n6 - Esvaziar a lista";
        cout<<"\n7 - Sair";
        cout<<"\nDigite sua opção: ";
        cin>>op;
        if (op < 1 || op > 7)
            cout<<"Opção inválida!!";
        if (op == 1)
        {
            cout<<"Digite o número a ser inserido
            ↳ no início da lista: ";
            LISTA *novo = new LISTA();
            cin>>novo->num;
            if (inicio == NULL)
            {
                // a lista estava vazia
                // e o elemento inserido será
                // o primeiro e o último
                inicio = novo;
                fim = novo;
                novo->prox = NULL;
                novo->ant = NULL;
```

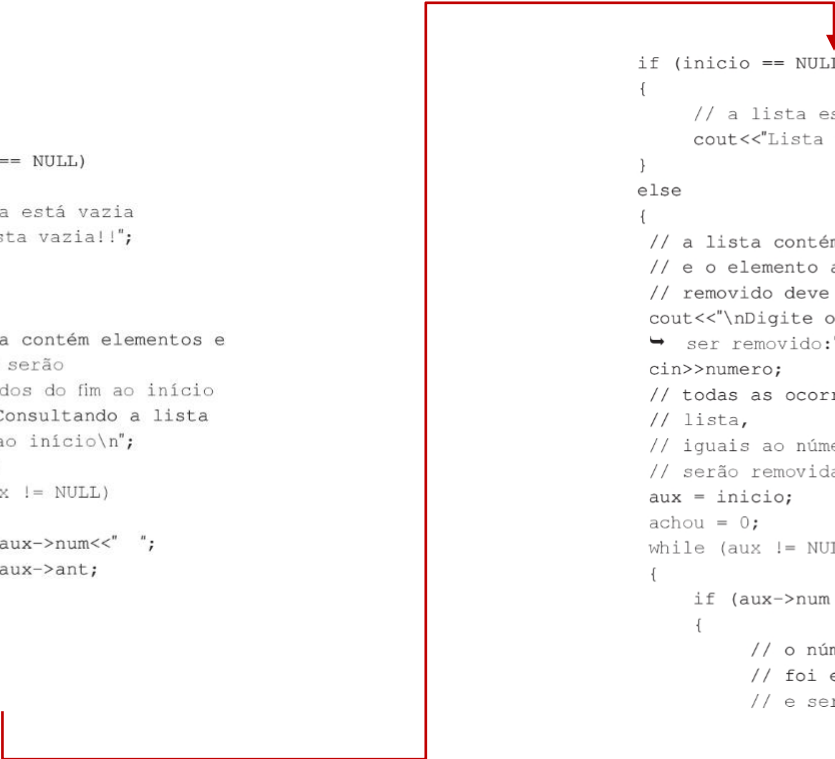
Lista duplamente encadeada

```
}
else
{
    // a lista já contém elementos
    // e o novo elemento
    // será inserido no início da lista
    novo->prox = inicio;
    inicio->ant = novo;
    novo->ant = NULL;
    inicio = novo;
}
cout<<"Número inserido no início da lista!!";
}
if (op == 2)
{
    cout<<"Digite o número a ser
    ↳ inserido no fim da lista: ";
    LISTA *novo = new LISTA();
    cin>>novo->num;
    if (inicio == NULL)
    {
        // a lista estava vazia
        // e o elemento inserido será
        // o primeiro e o último
        inicio = novo;
        fim = novo;
        novo->prox = NULL;
        novo->ant = NULL;
    }
}
```

```
else
{
    // a lista já contém elementos
    // e o novo elemento
    // será inserido no fim da lista
    fim->prox = novo;
    novo->ant = fim;
    novo->prox = NULL;
    fim = novo;
}
cout<<"Número inserido no fim da
↳ lista!!";
}
if (op == 3)
{
    if (inicio == NULL)
    {
        // a lista está vazia
        cout<<"Lista vazia!!";
    }
    else
    {
        // a lista contém elementos e
        // estes serão
        // mostrados do início ao fim
        cout<<"\nConsultando a lista
        ↳ do início ao fim\n";
        aux = inicio;
        while (aux != NULL)
        {
            cout<<aux->num<<" ";
            aux = aux->prox;
        }
    }
}
```

Lista duplamente encadeada

```
if (op == 4)
{
    if (inicio == NULL)
    {
        // a lista está vazia
        cout<<"Lista vazia!!";
    }
    else
    {
        // a lista contém elementos e
        // estes serão
        // mostrados do fim ao início
        cout<<"\nConsultando a lista
        ↳ do fim ao início\n";
        aux = fim;
        while (aux != NULL)
        {
            cout<<aux->num<<" ";
            aux = aux->ant;
        }
    }
}
if (op == 5)
{
```



```
if (inicio == NULL)
{
    // a lista está vazia
    cout<<"Lista vazia!!";
}
else
{
    // a lista contém elementos
    // e o elemento a ser
    // removido deve ser digitado
    cout<<"\nDigite o elemento a
    ↳ ser removido:";
    cin>>numero;
    // todas as ocorrências da
    // lista,
    // iguais ao número digitado,
    // serão removidas
    aux = inicio;
    achou = 0;
    while (aux != NULL)
    {
        if (aux->num == numero)
        {
            // o número digitado
            // foi encontrado na lista
            // e será removido
```

Lista duplamente encadeada

```
achou = achou + 1;
if (aux == inicio)
{
    // o número a
    // ser removido
    // é o primeiro da
    // lista
    inicio = aux->prox;
    if (inicio != NULL)
    {
        inicio->ant = NULL;
    }
    delete (aux);
    aux = inicio;
}
else if (aux == fim)
{
    // o número a ser
    // removido
    // é o último da
    // lista
    fim = fim->ant;
    fim->prox = NULL;
    delete (aux);
    aux = NULL;
}
```

```
    else {
        // o número a ser
        // removido
        // está no meio
        // da lista
        aux->ant->prox =
            ↪ aux->prox;
        aux->prox->ant =
            ↪ aux->ant;
        LISTA *aux2;
        aux2 = aux->prox;
        delete(aux);
        aux = aux2;
    }
    else
    {
        aux = aux->prox;
    }
}
if (achou == 0)
    cout<<"Número não encontrado";
else if (achou == 1)
    cout<<"Número removido 1 vez";
else
    cout<<"Número removido
    ↪ "<<achou<<" vezes";
}
```

```
}
```

Lista duplamente encadeada

```
if (op == 6)
{
    if (inicio == NULL)
    {
        // a lista está vazia
        cout<<"Lista vazia!!";
    }
    else
    {
        // a lista será esvaziada
        aux = inicio;
        while (aux != NULL)
        {
            inicio = inicio->prox;
            delete(aux);
            aux = inicio;
        }
        cout<<"Lista esvaziada";
    }
}
getch();
}
while (op != 7);
}
```


Perguntas?

Bibliografia da aula

- Notas de aula do Prof. Edson L.F. Senne (UNESP/INPE) em 2010.
- ASCENCIO, A. F. G.; ARAÚJO, G. S. Estrutura de dados. Algoritmos, análise da complexidade e implementação em Java e C/C++. Pearson. 2010.