

Lista circular

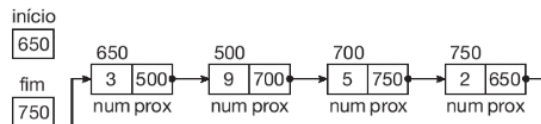
Murilo Dantas

Lista circular

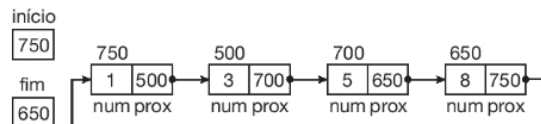
- Caso especial de lista dupla
 - ▶ O ponteiro **prox** do “último” elemento aponta para o “primeiro” elemento da lista.
 - ▶ O ponteiro **prev** do “primeiro” elemento aponta para o “último” elemento da lista.

Lista circular

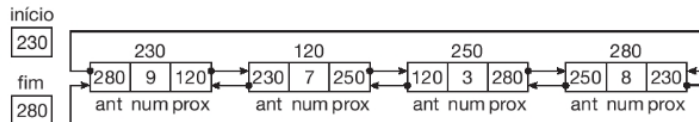
Lista circular simplesmente encadeada e não ordenada



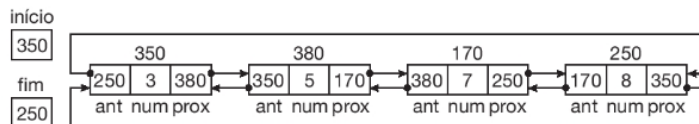
Lista circular simplesmente encadeada e ordenada



Lista circular duplamente encadeada e não ordenada



Lista circular duplamente encadeada e ordenada

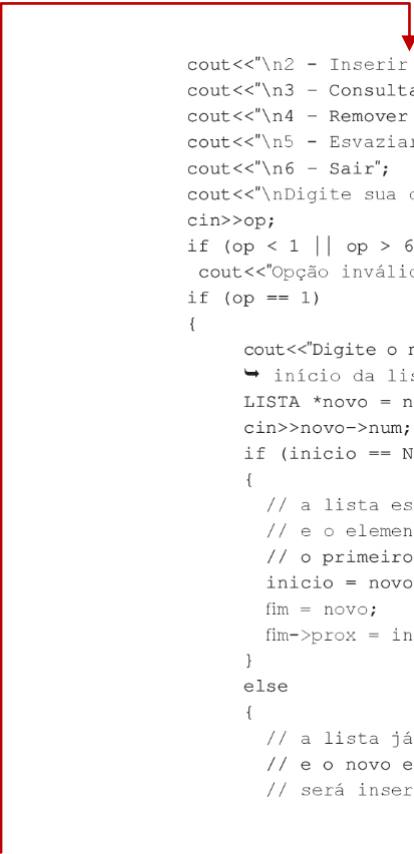


Lista circular simp. enc. não ord.

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

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

    // a lista está vazia, logo,
    // o ponteiro inicio têm 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;
    // o ponteiro anterior é um ponteiro auxiliar
    LISTA *anterior;
    // 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 toda a lista";
        cout<<"\n4 - Remover da lista";
        cout<<"\n5 - Esvaziar a lista";
        cout<<"\n6 - Sair";
        cout<<"\nDigite sua opção: ";
        cin>>op;
        if (op < 1 || op > 6)
            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;
                fim->prox = inicio;
            }
            else
            {
                // a lista já contém elementos
                // e o novo elemento
                // será inserido no início da lista
```

Lista circular simp. enc. não ord.

```
    novo->prox = inicio;
    inicio = novo;
    fim->prox = inicio;
}
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;
        fim->prox = inicio;
    }
    else
    {
        // a lista já contém elementos
        // e o novo elemento
        // será inserido no fim da lista
        fim->prox = novo;
        fim = novo;
        fim->prox = inicio;
    }
}
```

```
    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 toda a
↳ lista\n";
        aux = inicio;
        do
        {
            cout<<aux->num<<" ";
            aux = aux->prox;
        }
        while (aux != inicio);
    }
}
if (op == 4)
{
    if (inicio == NULL)
    {
        // a lista está vazia
        cout<<"Lista vazia!!";
    }
}
```

Lista circular simp. enc. não ord.

```
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;
    anterior = NULL;
    achou = 0;
    // descobrindo a quantidade de
    // elementos da lista
    int quantidade = 0;
    aux = inicio;
    do
    {
        quantidade = quantidade + 1;
        aux = aux->prox;
    }
    while (aux != inicio);
```

```
int elemento = 1;
do
{
    // se a lista possui apenas
    // um elemento
    if (inicio == fim &&
        inicio->num == numero)
    {
        delete(inicio);
        inicio = NULL;
        achou = achou + 1;
    }
    else
    {
        if (aux->num == numero)
        {
            // o número digitado
            // foi encontrado na lista
            // e será removido
            achou = achou + 1;
            if (aux == inicio)
            {
                // o número a ser
                // removido
                // é o primeiro da lista
                inicio = aux->prox;
                fim->prox = inicio;
                delete(aux);
                aux = inicio;
            }
        }
    }
}
```

Lista circular simp. enc. não ord.

```
else if (aux == fim)
{
    // o número a ser
    // removido
    // é o último da
    // lista
    fim = anterior;
    fim->prox = inicio;
    delete(aux);
    aux = NULL;
}
else
{
    // o número a ser
    // removido
    // está no meio da
    // lista
    anterior->prox =
        aux->prox;
    delete(aux);
    aux = anterior-
        >prox;
}
}
else
{
    anterior = aux;
    aux = aux->prox;
}
}
```

```
elemento = elemento + 1;
}
while (elemento <= quantidade);
if (achou == 0)
    cout<<"Número não encontrado";
else if (achou == 1)
    cout<<"Número removido 1 vez";
else
    cout<<"Número removido "<<achou<<"
        vezes";
}
}
if (op == 5)
{
    if (inicio == NULL)
    {
        // a lista está vazia
        cout<<"Lista vazia!!";
    }
    else
    {
        // a lista será esvaziada
        aux = inicio;
        do
        {
            inicio = inicio ->prox;
            delete(aux);
        }
    }
}
```

Lista circular simp. enc. não ord.

```
        aux = inicio;
    }
    while (aux != fim);
    delete(fim);
    inicio=NULL;
    cout<<"Lista esvaziada";
}
}
getch();
}
while (op != 6);
}
```


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.