

Tecnologia Em Analise e Desenv. de Sistemas

Estrutura de Dados

Aula 7 – Lista Simplesmente Encadeada

Prof. Dr. Ricardo Luis Balieiro

Novo **0x00**

0x00
Novo **0x00**

0x00
Novo

0x00
0x00 **Novo** **0x00**

Dica Importante

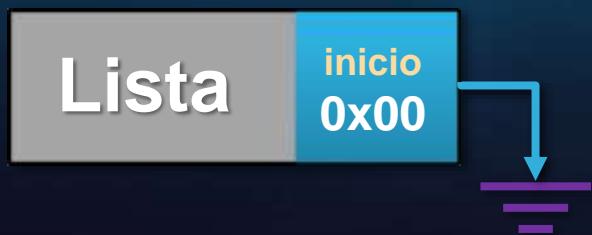
DICA IMPORTANTE

- O link abaixo possui muitas informações a respeito de Estrutura de dados.
- <https://www.ime.usp.br/~pf/algoritmos/idx.html>

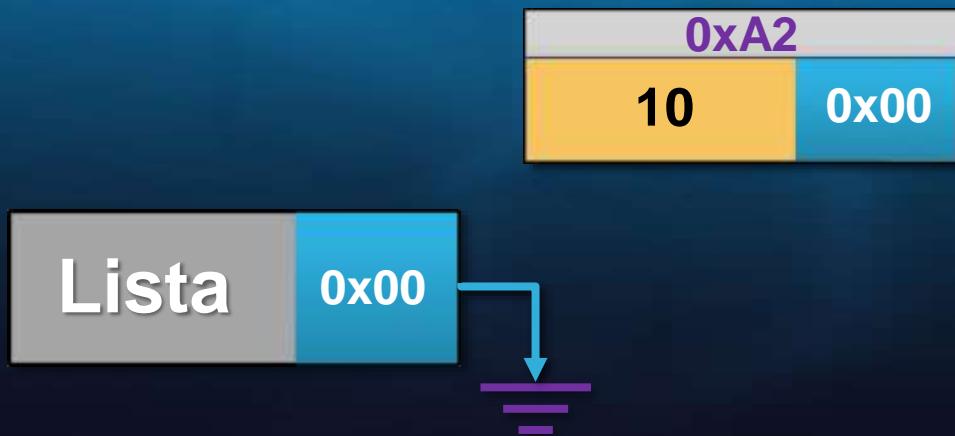
INserir no Início

Simulação 1

Inserir no início – Simulação 1



Inserir no início – Simulação 1



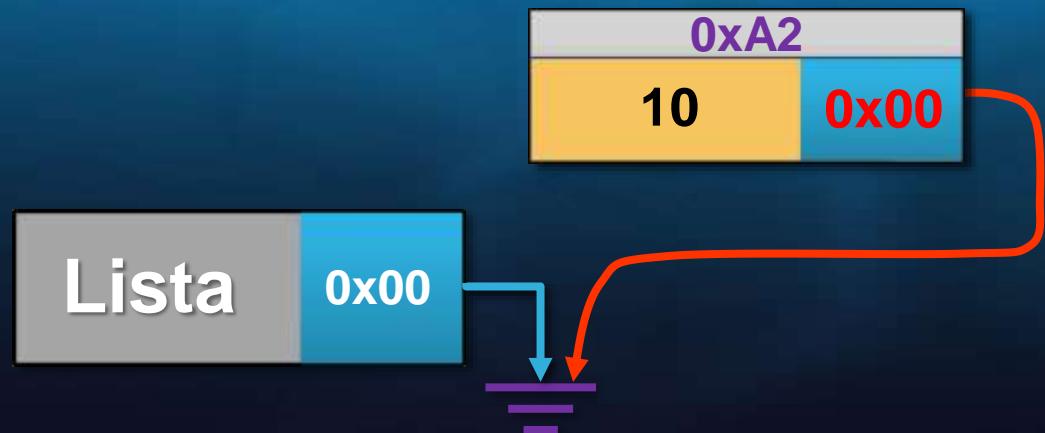
Inserir no início – Simulação 1

```
ptrNoNovo->proxNo = ptrLista->inicio;
```

```
ptrLista->inicio = ptrNoNovo;
```

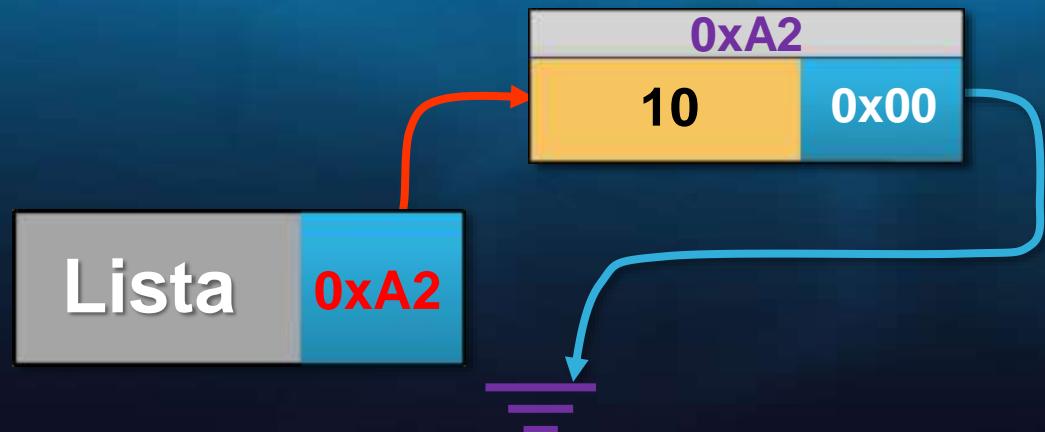
```
// Incrementa o quantidade de Nós
```

```
ptrLista->qtdNo++;
```



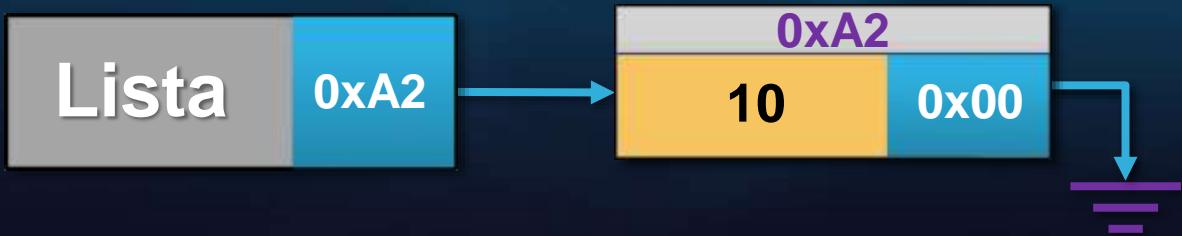
Inserir no início – Simulação 1

```
ptrNoNovo->proxNo = ptrLista->inicio;  
  
ptrLista->inicio = ptrNoNovo;  
  
// Incrementa o quantidade de Nós  
ptrLista->qtdNo++;
```



Inserir no início – Simulação 1

```
ptrNoNovo->proxNo = ptrLista->inicio;  
  
ptrLista->inicio = ptrNoNovo;  
  
// Incrementa o quantidade de Nós  
ptrLista->qtdNo++;
```



INSERIR NO INÍCIO

SIMULAÇÃO 2

Inserir no início – Simulação 2



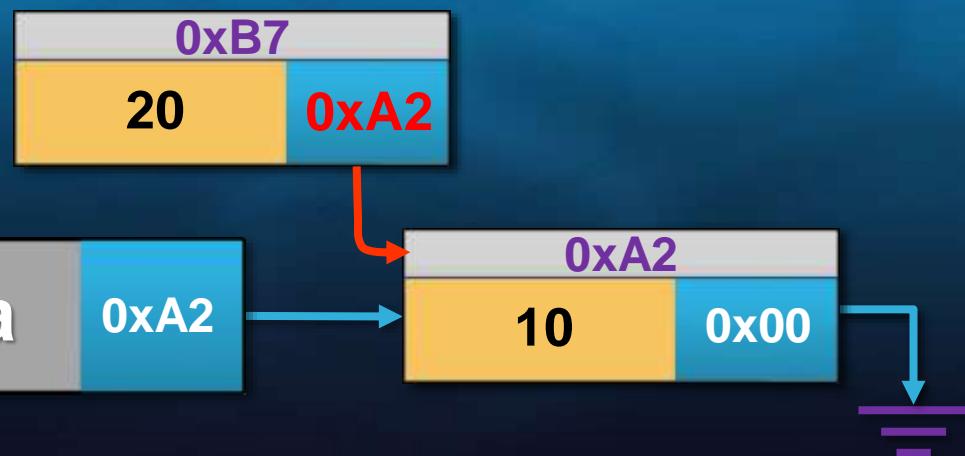
Inserir no início – Simulação 2

```
ptrNoNovo->proxNo = ptrLista->inicio;
```

```
ptrLista->inicio = ptrNoNovo;
```

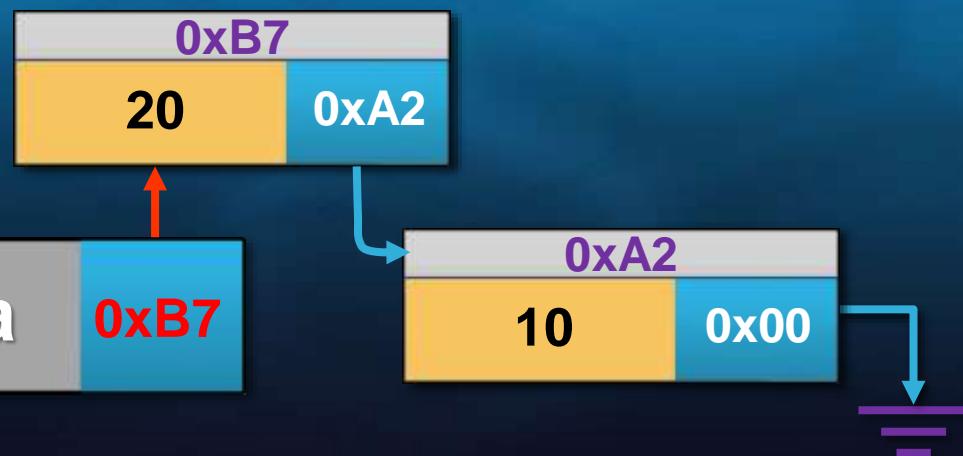
```
// Incrementa o quantidade de Nós
```

```
ptrLista->qtdNo++;
```



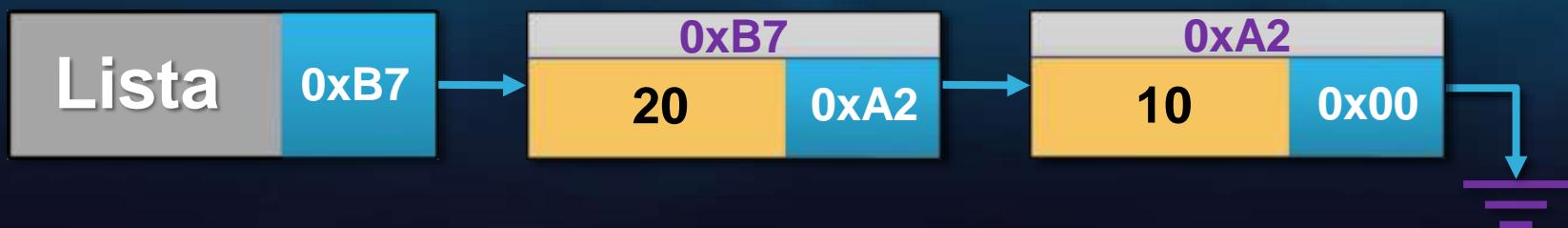
Inserir no início – Simulação 2

```
ptrNoNovo->proxNo = ptrLista->inicio;  
ptrLista->inicio = ptrNoNovo;  
// Incrementa o quantidade de Nós  
ptrLista->qtdNo++;
```



Inserir no início – Simulação 2

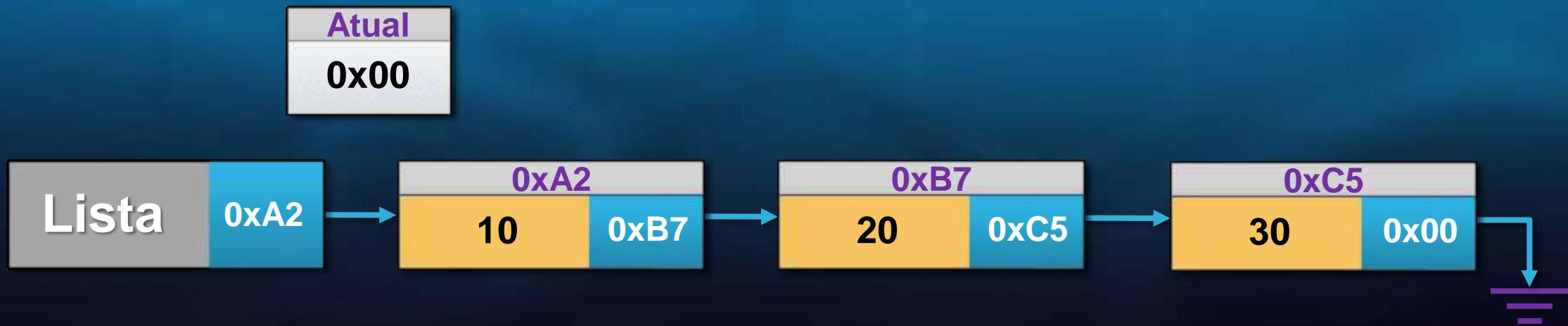
```
ptrNoNovo->proxNo = ptrLista->inicio;  
  
ptrLista->inicio = ptrNoNovo;  
  
// Incrementa o quantidade de Nós  
ptrLista->qtdNo++;
```



EXIBIR LISTA

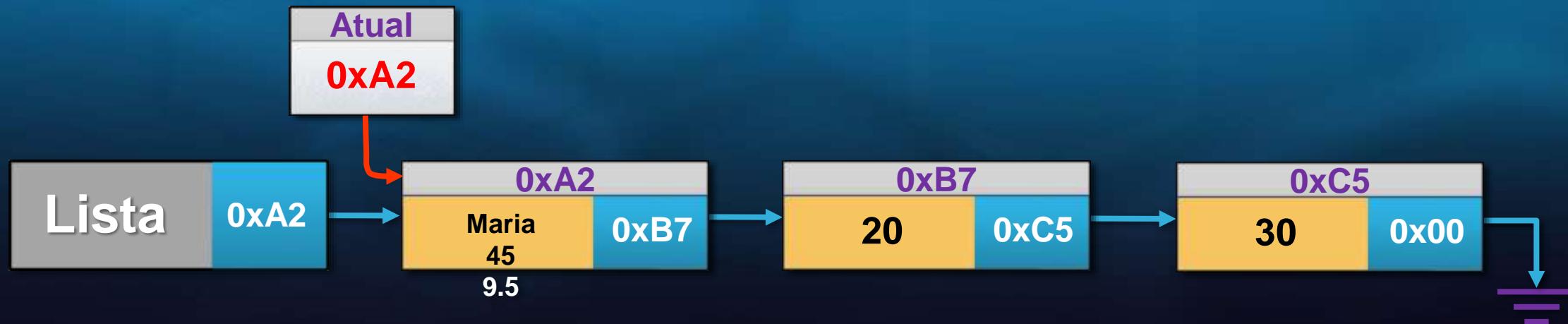
SIMULAÇÃO
Função básica de exibição

```
void exibirLista(Lista *ptrLista) {  
    No *ptrNoAtual;
```



```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {  
  
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;  
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;  
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;  
  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

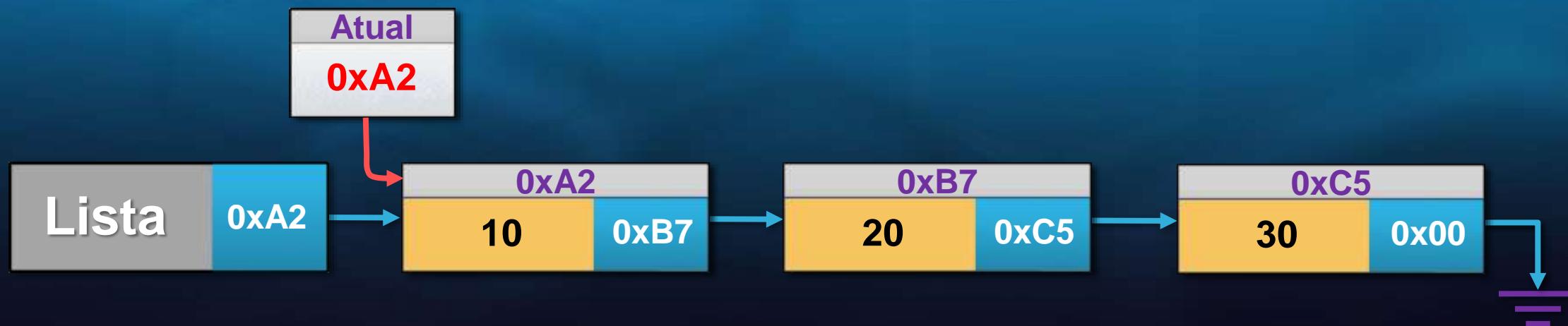
```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
```

```
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
```

```
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```



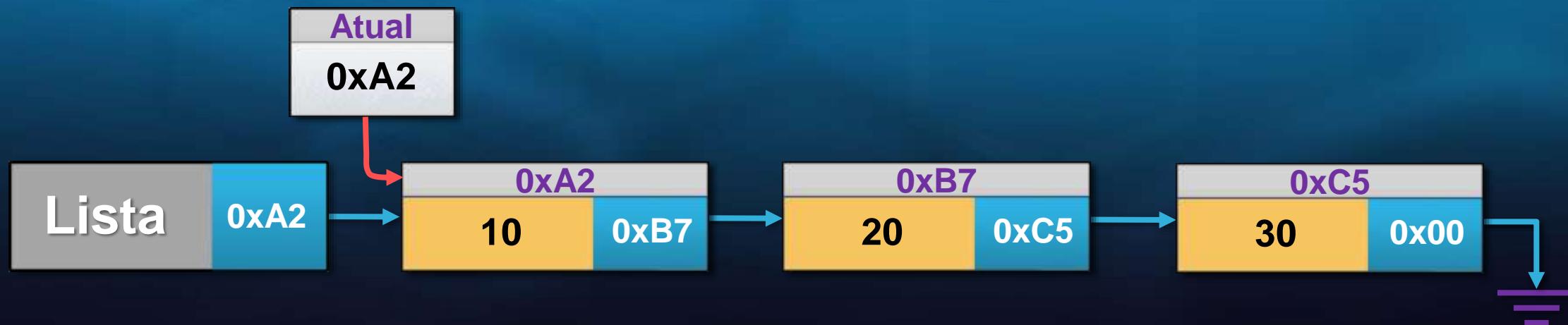
```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```

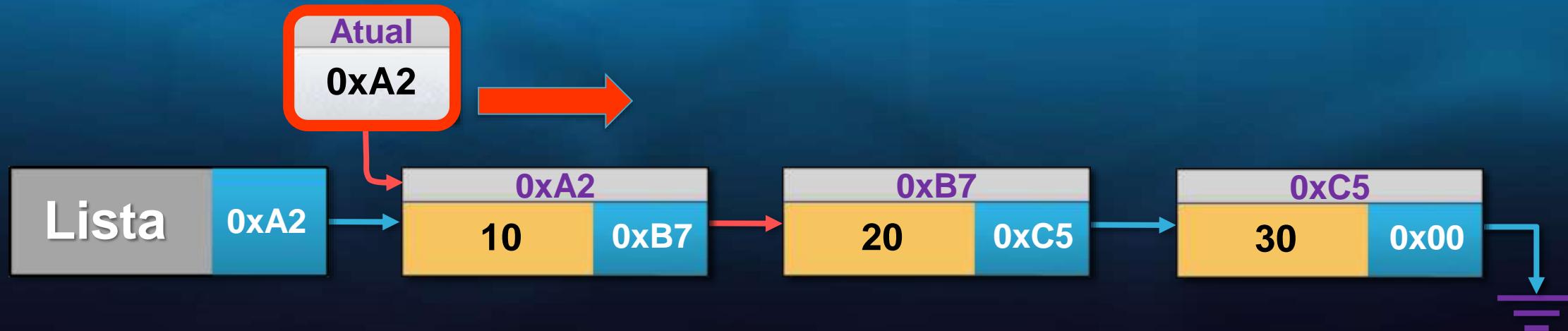


```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}
```

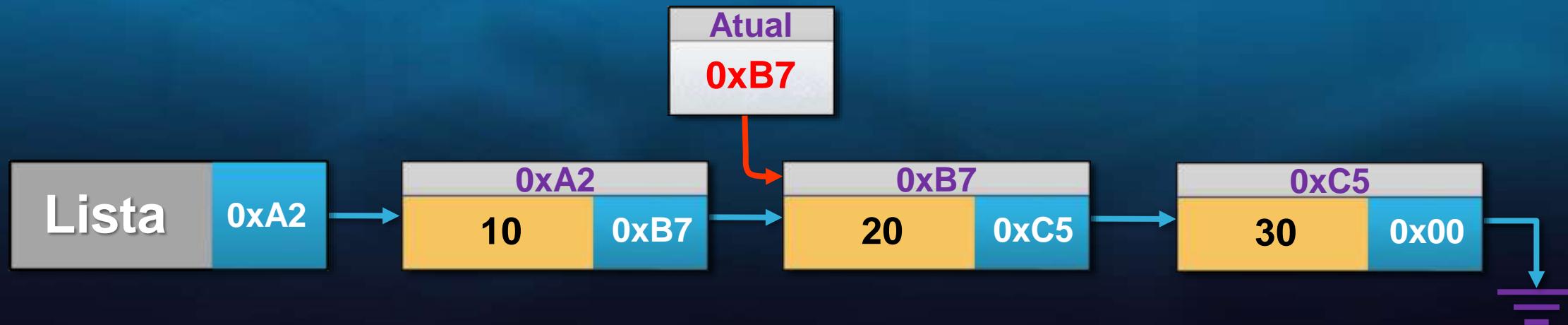


```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}
```



```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

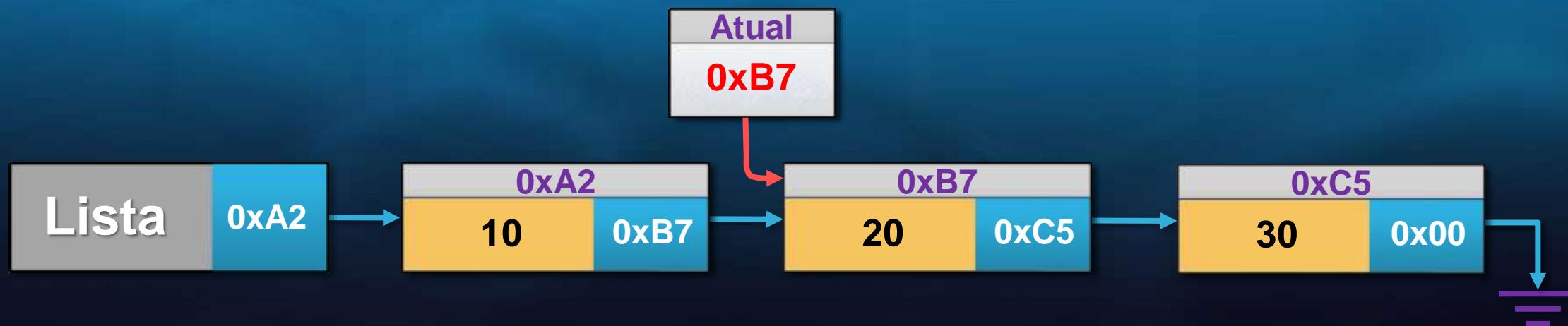
```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
```

```
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
```

```
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```



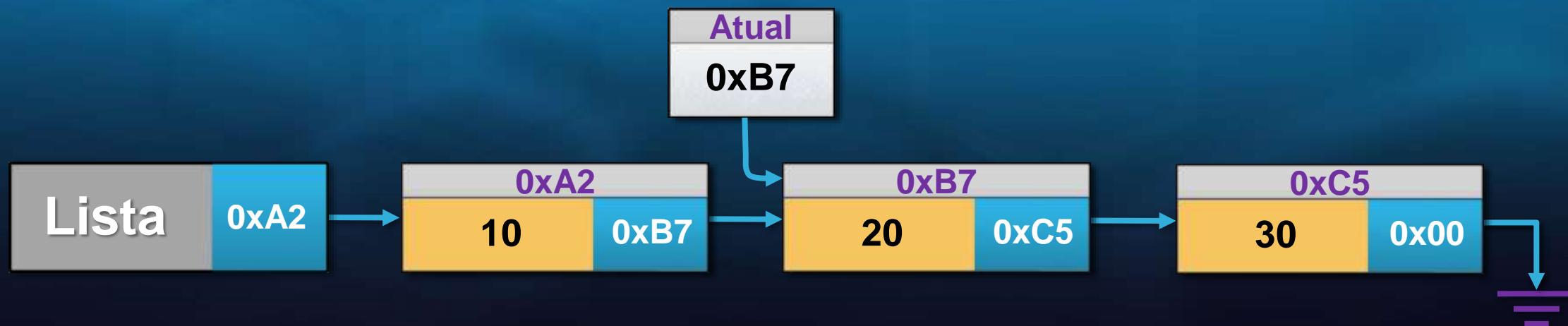
```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```

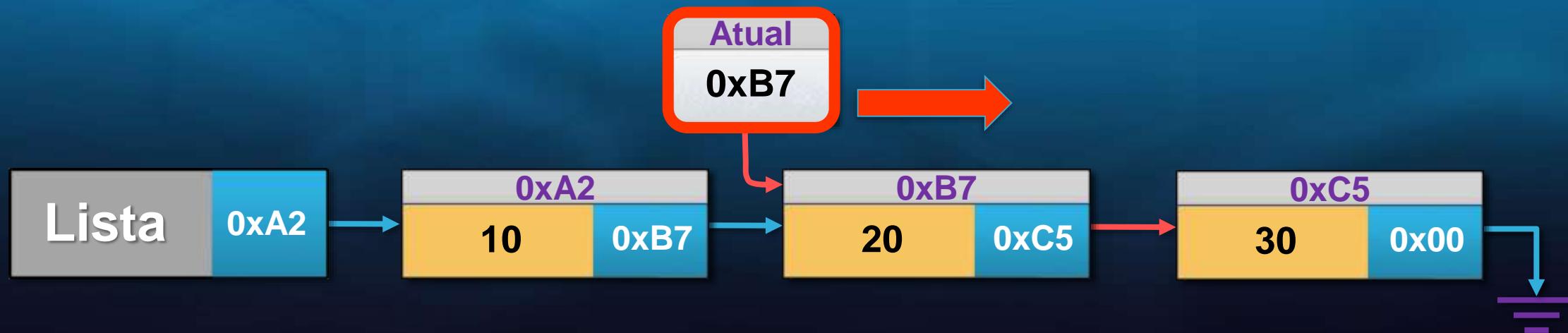


```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}
```

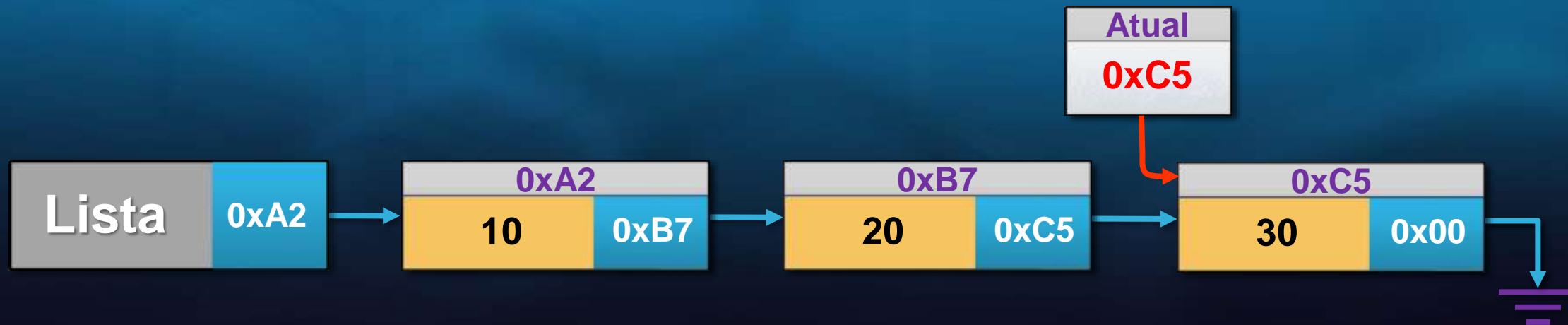


```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}
```



```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

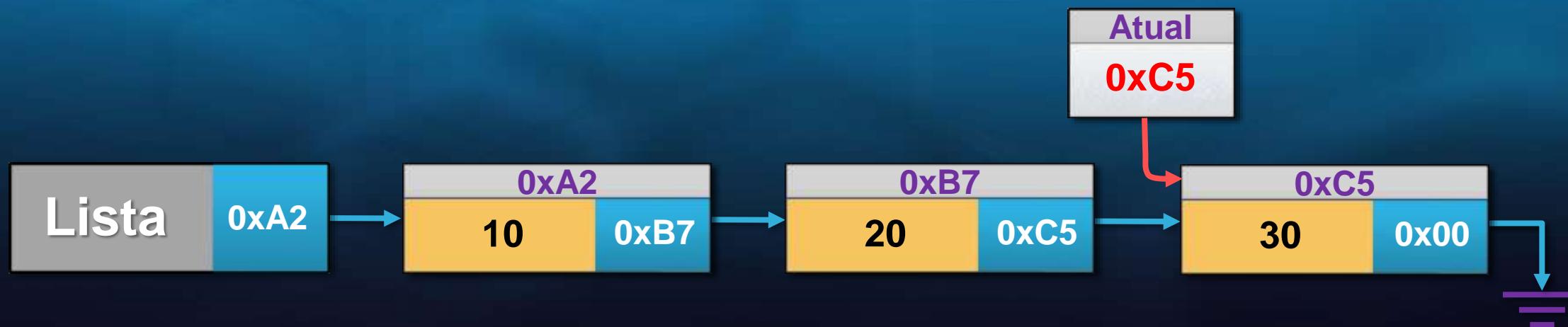
```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
```

```
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
```

```
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```



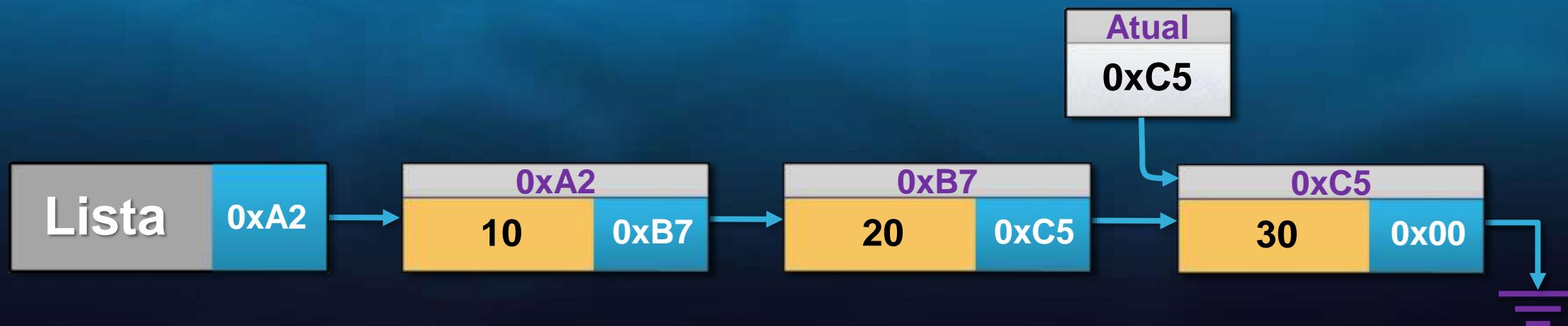
```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```

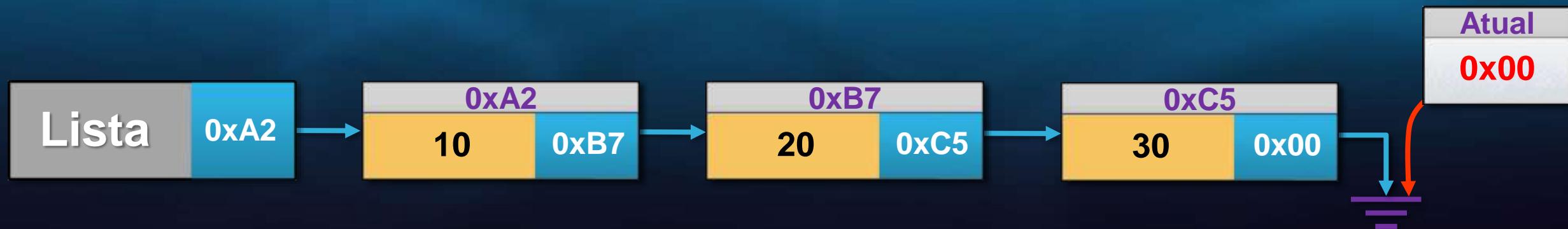


```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}
```



```
ptrNoAtual = ptrLista->inicio;
```

```
while (ptrNoAtual != NULL) {
```

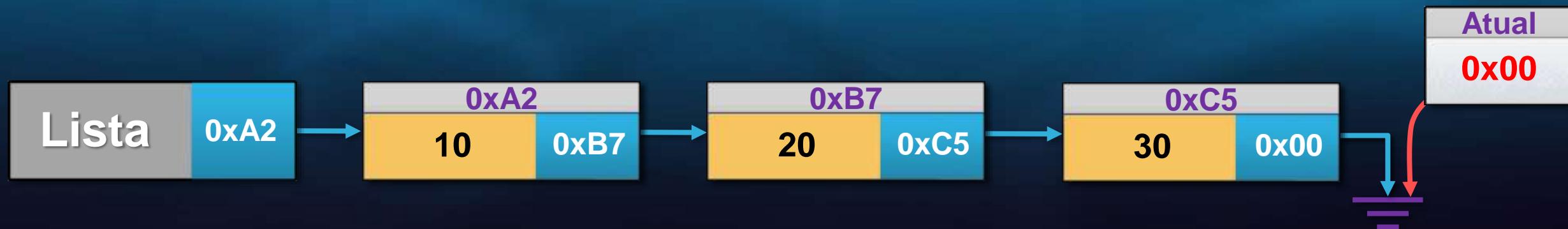
```
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
```

```
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
```

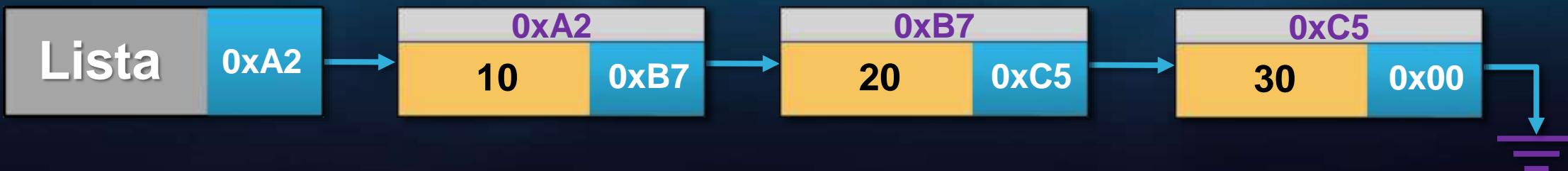
```
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```



```
ptrNoAtual = ptrLista->inicio;  
  
while (ptrNoAtual != NULL) {  
  
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;  
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;  
    cout << "Média: " << ptrNoAtual->dados.media << endl << endl << endl;  
  
    ptrNoAtual = ptrNoAtual->proxNo;
```



EXIBIR LISTA

SIMULAÇÃO
Função avançada de exibição

=====

LISTA

Quantidade: 3

Inicio: 0028CE50

=====

0028CE50

Matrícula: 10

Nome: José

Média: 7

Próximo-> 0028CEA8

0028CEA8

Matrícula: 20

Nome: Jesus

Média: 10

Próximo-> 00289768

00289768

Matrícula: 30

Nome: Maria

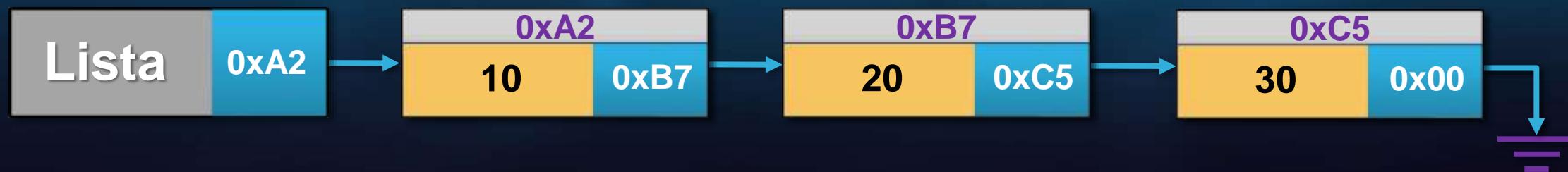
Média: 8

Próximo-> 00000000

```
// Nó inicio da lista
struct Lista {
    int qtdNo;
    No *inicio;
};
```

```
cout << "=====LISTA" << endl;
cout << "Quantidade: " << ptrLista->qtdNo << endl;
cout << "\tInício: " << ptrLista->inicio << endl;
cout << "=====
```

```
=====
LISTA
Quantidade: 3
```

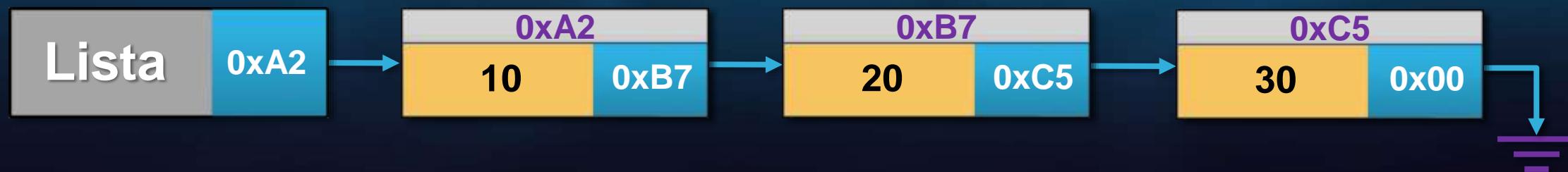


```
// Nó inicio da lista
struct Lista {
    int qtdNo;
    No *inicio;
};
```



```
cout << "====="
cout << "          LISTA" << endl;
cout << "  Quantidade: " << ptrLista->qtdNo << endl;
cout << " \tInício: " << ptrLista->inicio << endl;
cout << "=====
```

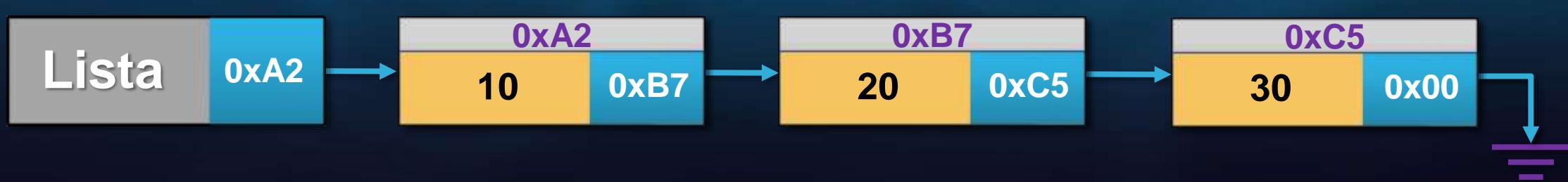
```
=====
LISTA
Quantidade: 3
Início: 0028CE50
```



```
No *ptrNoAtual;
```

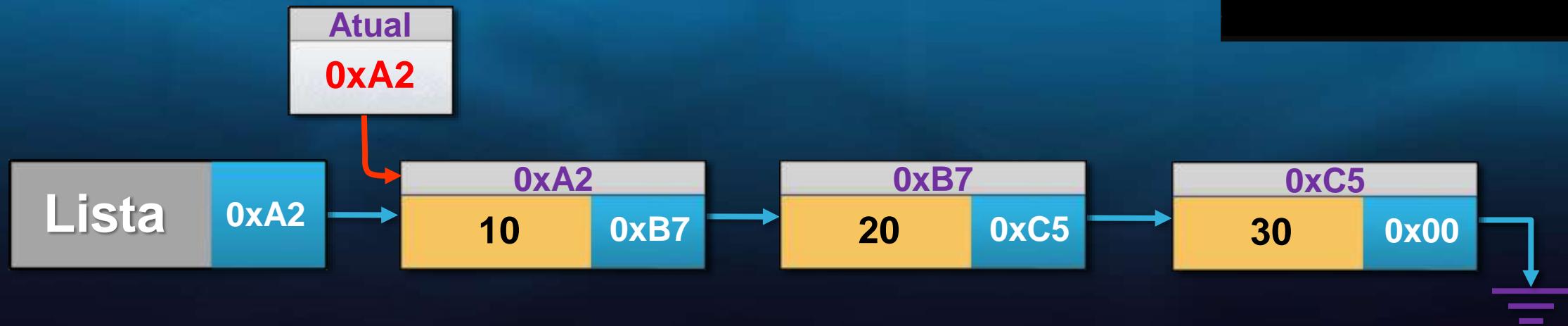
```
ptrNoAtual = ptrLista->inicio;
```

Atual
0x00



```
=====
LISTA
Quantidade: 3
Início: 0028CE50
=====
```

```
No *ptrNoAtual;  
ptrNoAtual = ptrLista->inicio;
```



```
===== LISTA =====  
Quantidade: 3  
Início: 0028CE50  
=====
```

```

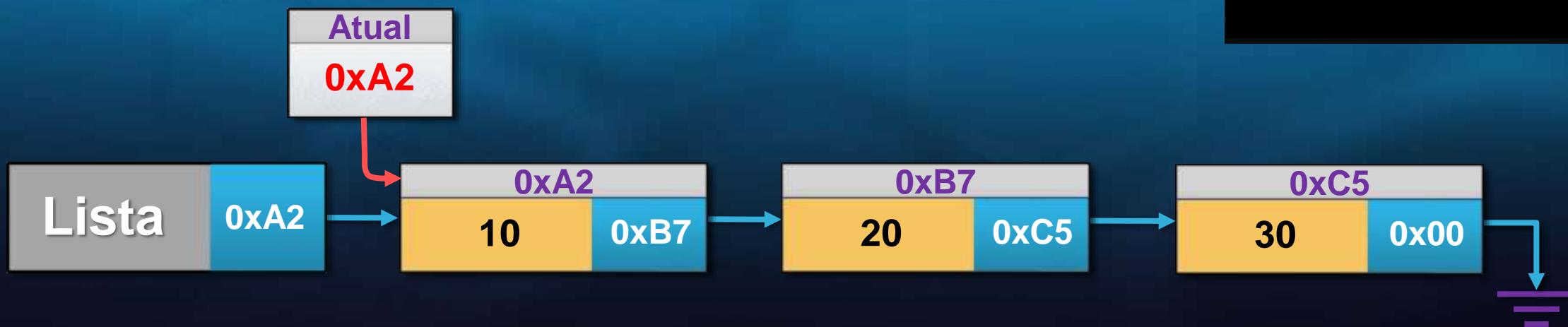
while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
=====
```



```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

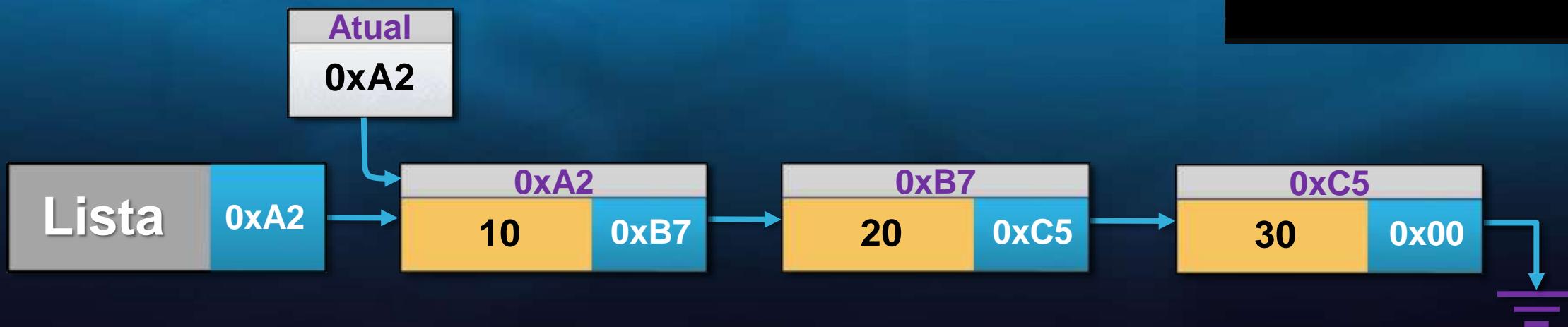
    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50

```



```

// Estrutura do Nó          // Dados sobre o ALUNO
struct No {
    Aluno dadosAluno;
    No *proxNo;
};

struct Aluno {
    int matricula;
    string nome;
    float media;
};

```

```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
}

```

```

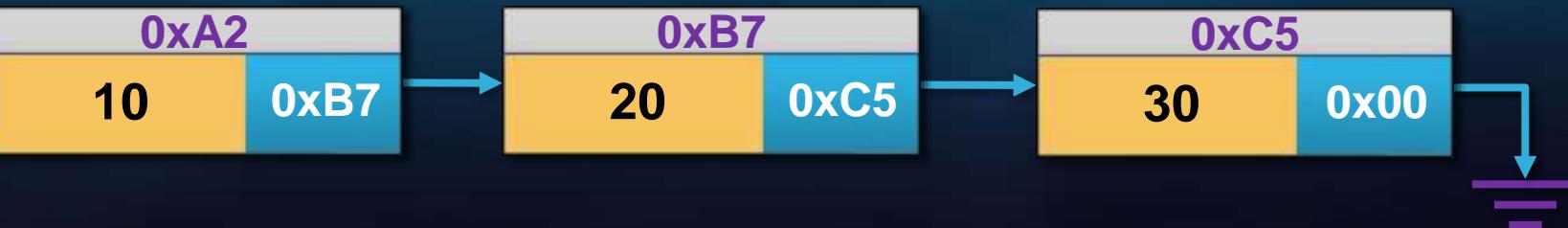
ptrNoAtual = ptrNoAtual->proxNo;
}

```



Lista

0xA2



```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10

```

```

// Estrutura do Nó
struct No {
    Aluno dadosAluno;
    No *proxNo;
};

// Dados sobre o ALUNO
struct Aluno {
    int matricula;
    string nome;
    float media;
};

```

```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
}

```

```

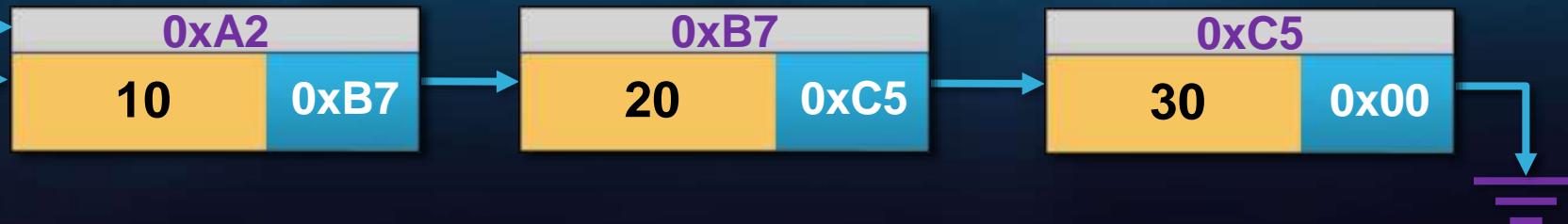
ptrNoAtual = ptrNoAtual->proxNo;
}

```



Lista

0xA2



```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10
Nome: José

```

```

// Estrutura do Nó
struct No {
    Aluno dadosAluno;
    No *proxNo;
};

// Dados sobre o ALUNO
struct Aluno {
    int matricula;
    string nome;
    float media;
};

```

```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
}

```

```

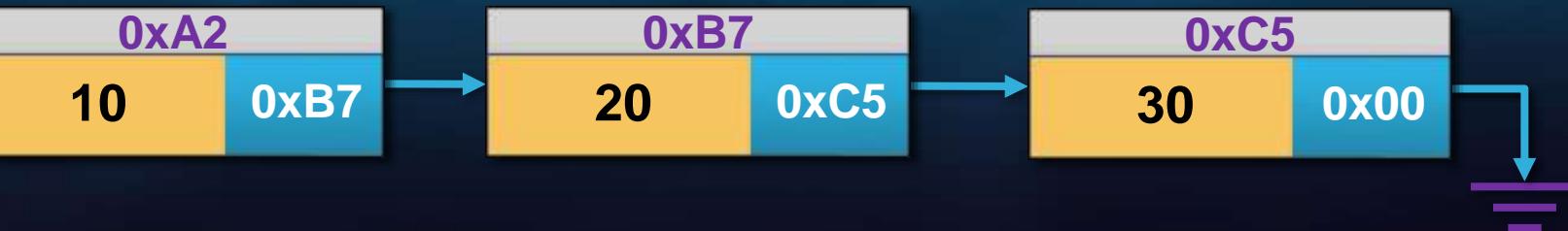
ptrNoAtual = ptrNoAtual->proxNo;
}

```



Lista

0xA2



```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10
Nome: José
Média: 7

```

```
// Estrutura do Nó
struct No {
    Aluno dadosAluno;
    No *proxNo;
};
```

```
// Dados sobre o ALUNO
struct Aluno {
    int matricula;
    string nome;
    float media;
};
```

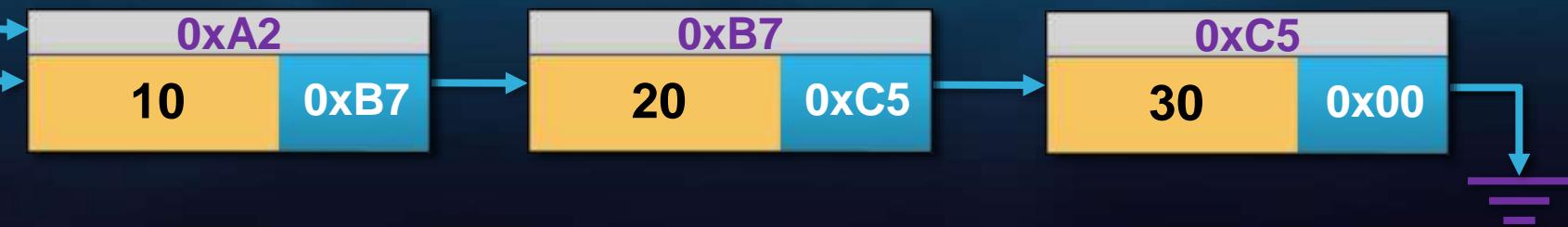
```
while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
}
```

```
ptrNoAtual = ptrNoAtual->proxNo;
```



Lista

`0xA2`



```
=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
```

```

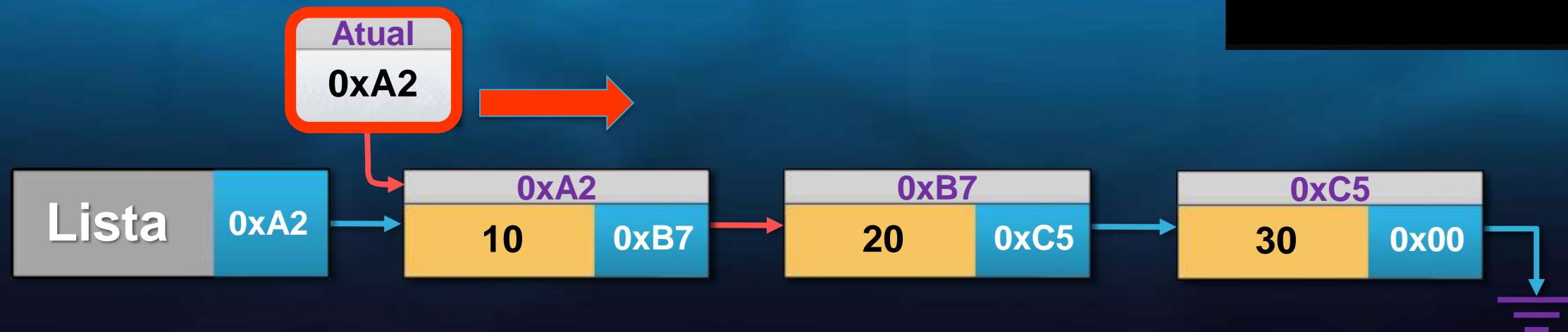
while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
}

ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
=====
```



```

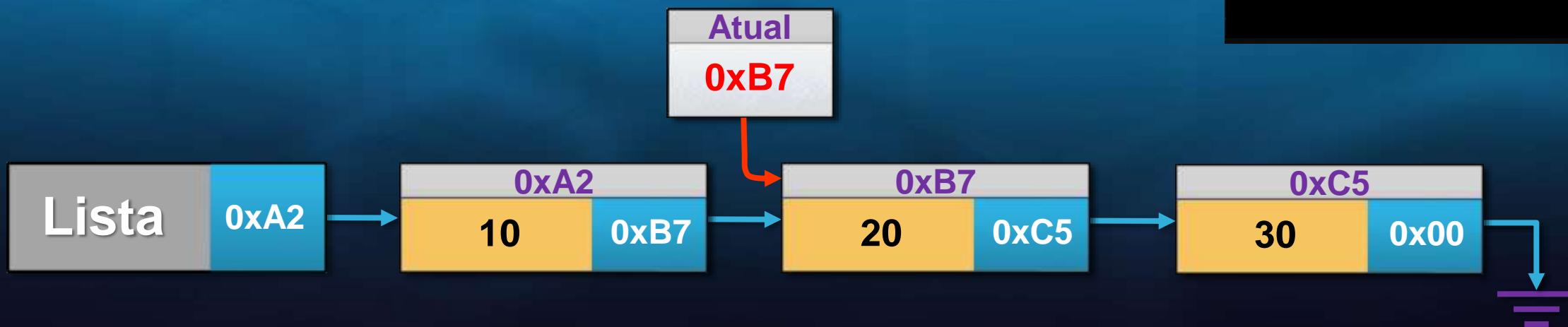
while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
=====
```



```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

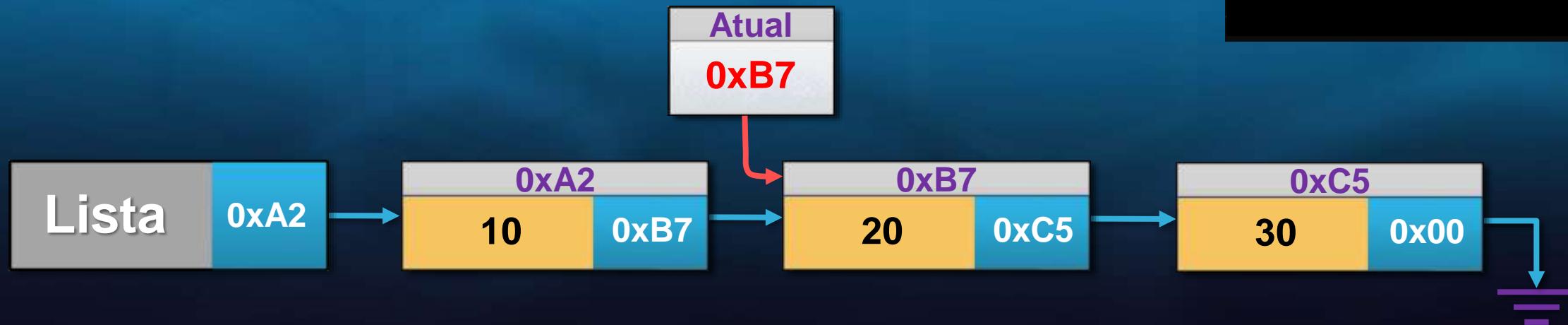
    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
-----
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8

```



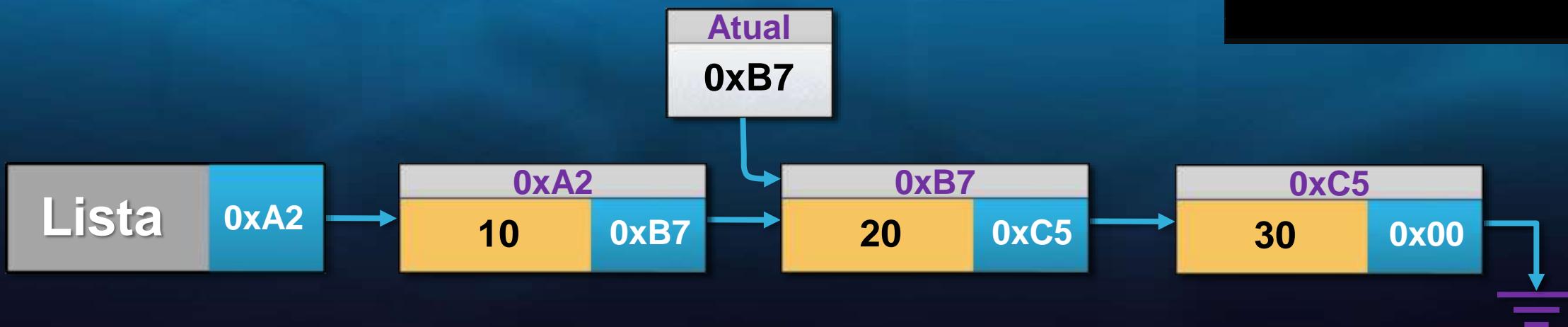
```
while (ptrNoAtual != NULL) {
```

```
    cout << "-----" << endl;
    cout << "        " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```

```
=====
LISTA
Quantidade: 3
Início: 0028CE50
=====
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
=====
0028CEA8
Matrícula: 20
Nome: Jesus
Média: 10
Próximo-> 00289768
```



```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

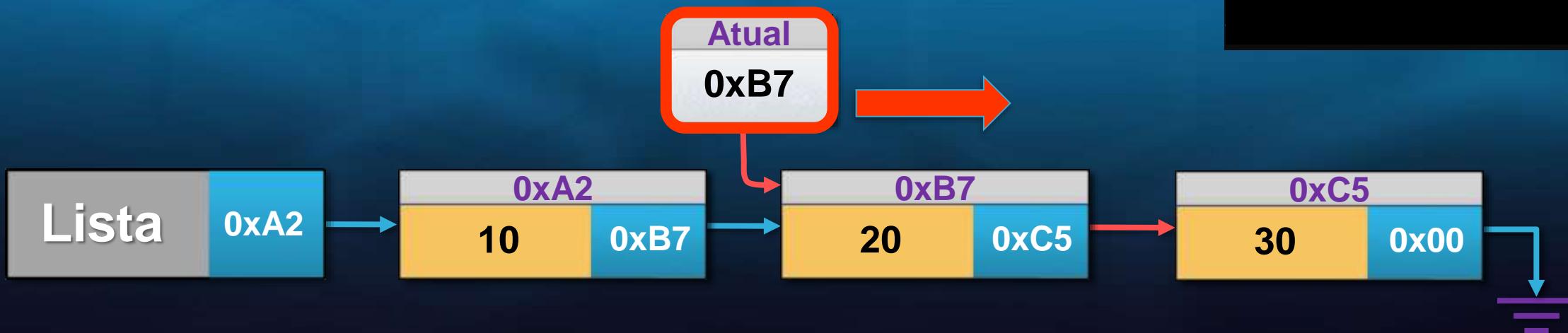
    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
=====
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
0028CEA8
Matrícula: 20
Nome: Jesus
Média: 10
Próximo-> 00289768

```



```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

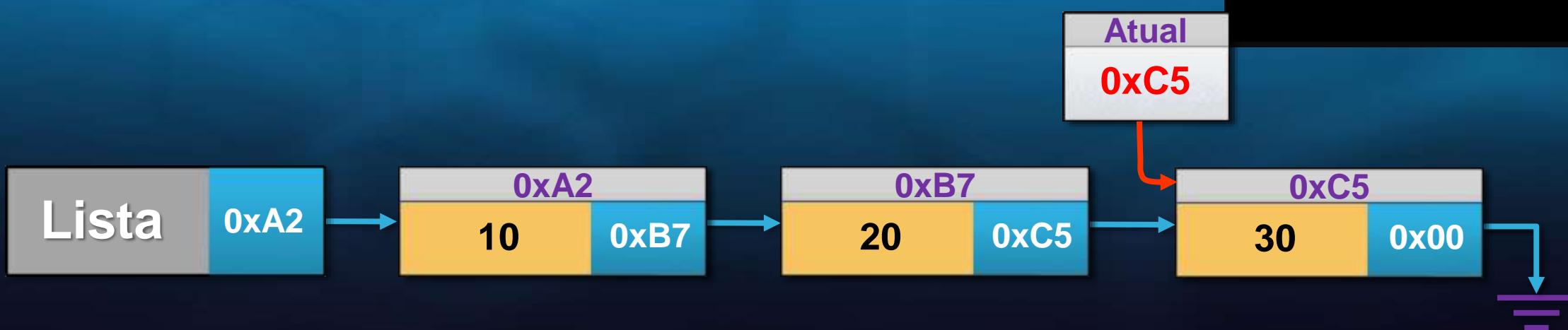
    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
=====
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
0028CEA8
Matrícula: 20
Nome: Jesus
Média: 10
Próximo-> 00289768

```



```
while (ptrNoAtual != NULL) {
```

```
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;
```

```
    ptrNoAtual = ptrNoAtual->proxNo;
```

```
}
```

```
===== LISTA =====
Quantidade: 3
Início: 0028CE50
=====
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
=====
0028CEA8
Matrícula: 20
Nome: Jesus
Média: 10
Próximo-> 00289768
=====
00289768
Matrícula: 30
Nome: Maria
Média: 8
Próximo-> 00000000
=====
```



```

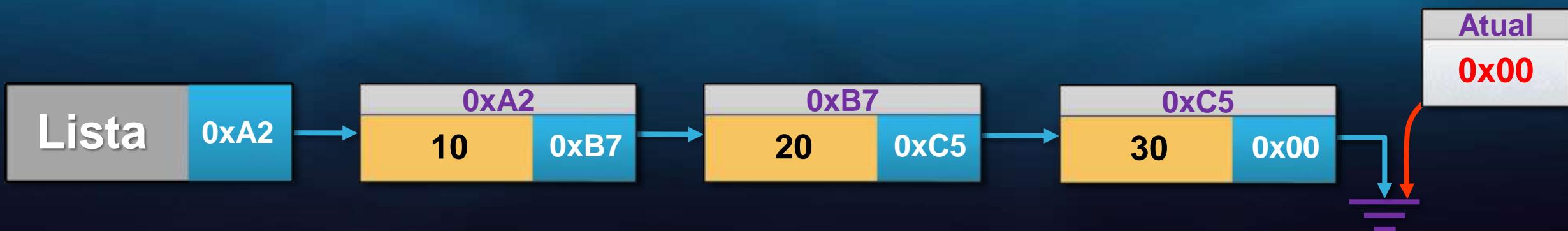
while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}

```

```

=====
LISTA
Quantidade: 3
Início: 0028CE50
=====
0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8
=====
0028CEA8
Matrícula: 20
Nome: Jesus
Média: 10
Próximo-> 00289768
=====
00289768
Matrícula: 30
Nome: Maria
Média: 8
Próximo-> 00000000
=====
```



```

while (ptrNoAtual != NULL) {
    cout << "-----" << endl;
    cout << "      " << ptrNoAtual << endl;
    cout << "Matrícula: " << ptrNoAtual->dados.matricula << endl;
    cout << "Nome: " << ptrNoAtual->dados.nome << endl;
    cout << "Média: " << ptrNoAtual->dados.media << endl;
    cout << "\tPróximo-> " << ptrNoAtual->proxNo << endl;

    ptrNoAtual = ptrNoAtual->proxNo;
}

```

===== LISTA =====

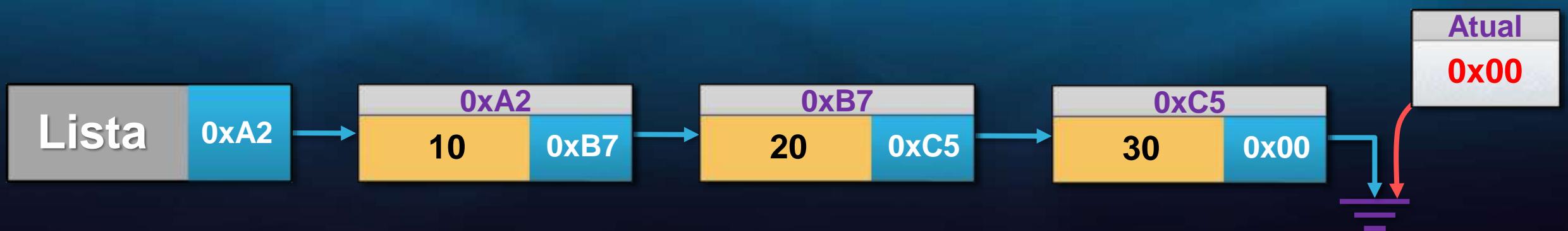
Quantidade: 3
Início: 0028CE50

0028CE50
Matrícula: 10
Nome: José
Média: 7
Próximo-> 0028CEA8

0028CEA8
Matrícula: 20
Nome: Jesus
Média: 10
Próximo-> 00289768

00289768
Matrícula: 30
Nome: Maria
Média: 8
Próximo-> 00000000

=====



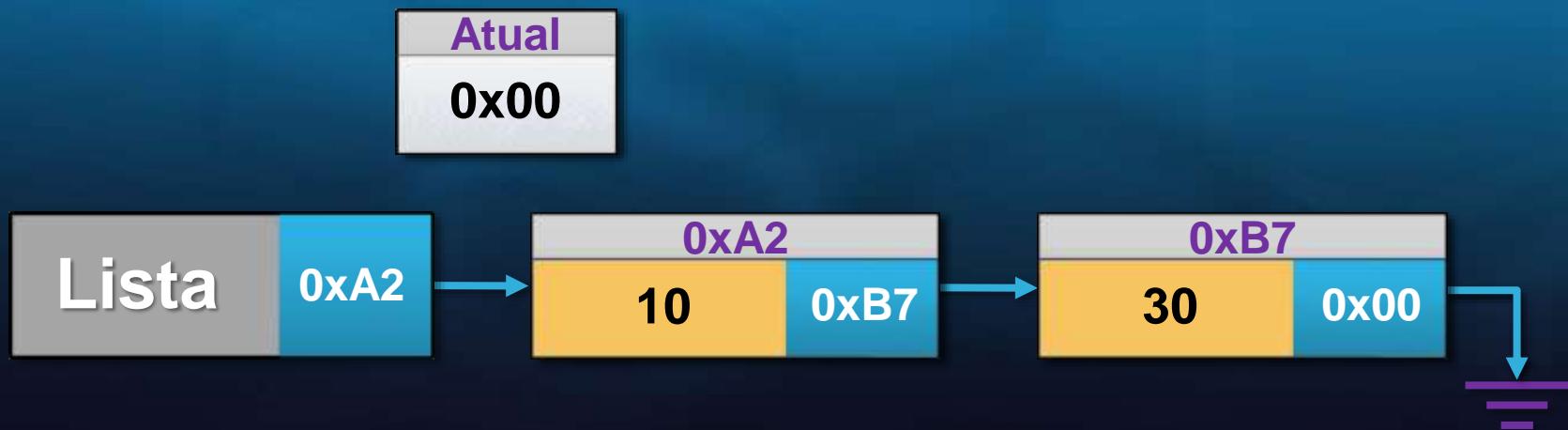
LIBERAR LISTA

SIMULAÇÃO

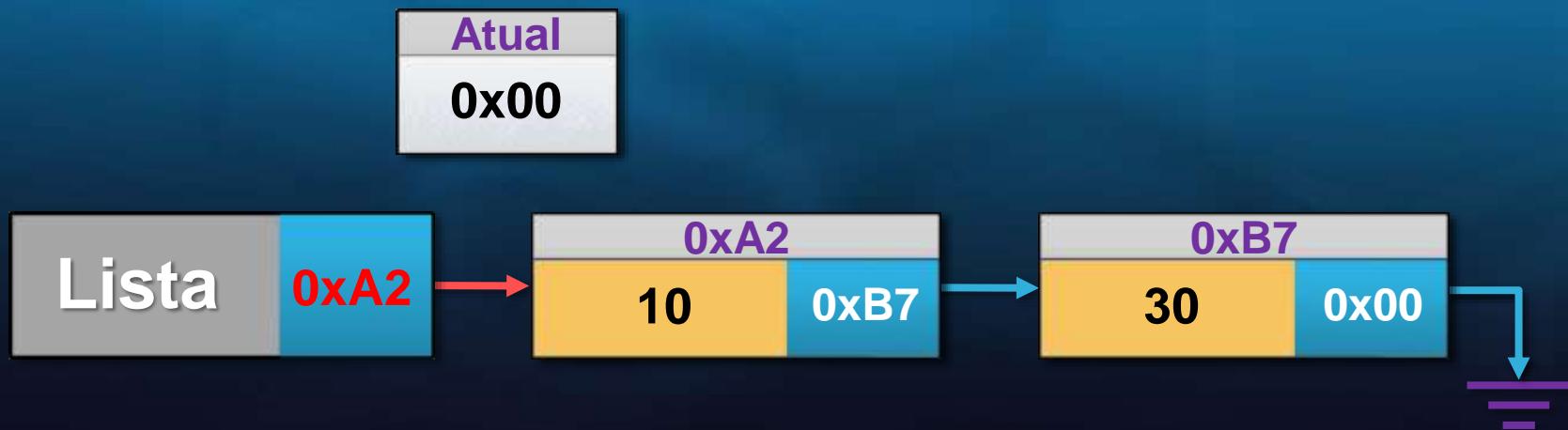
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



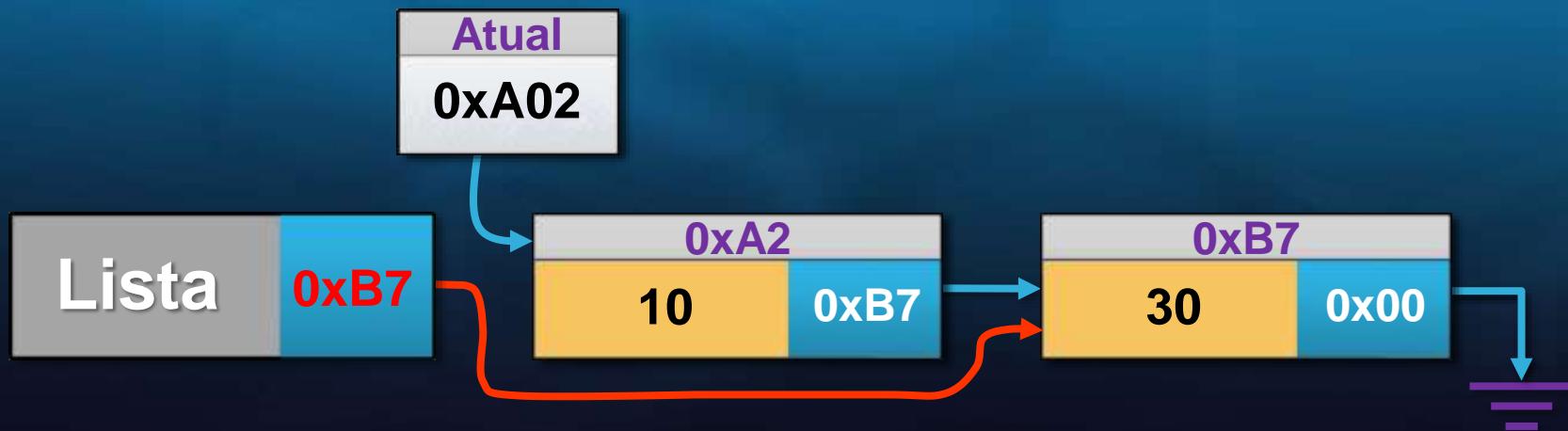
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



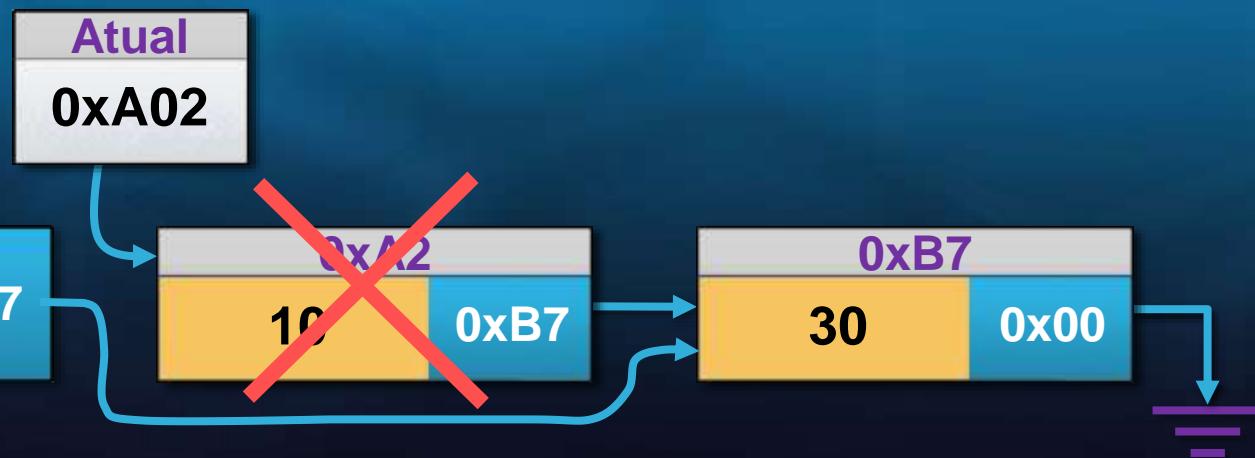
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



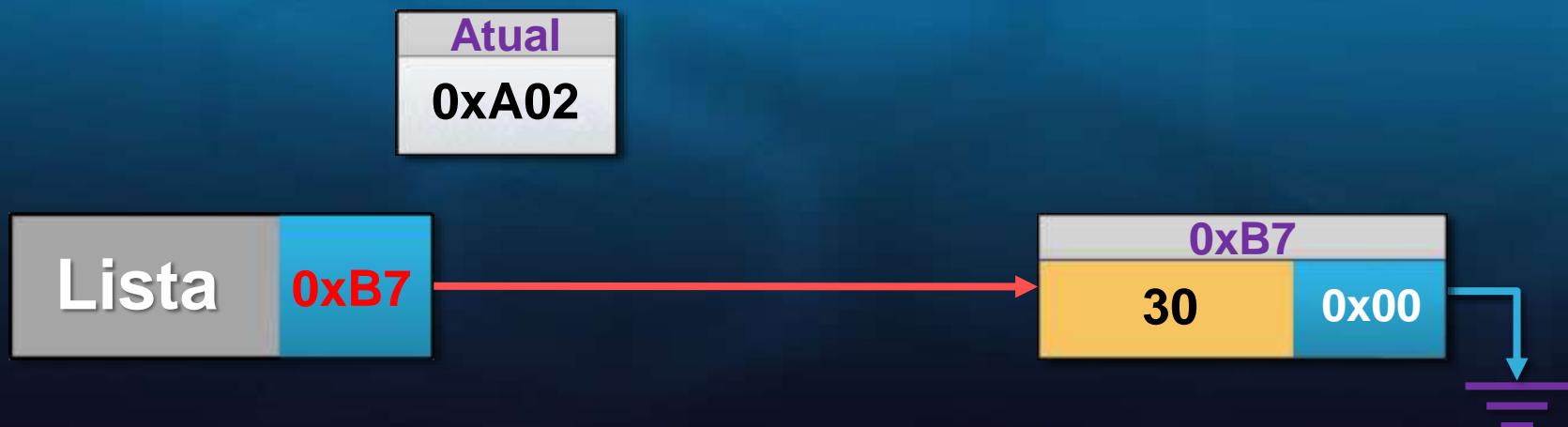
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



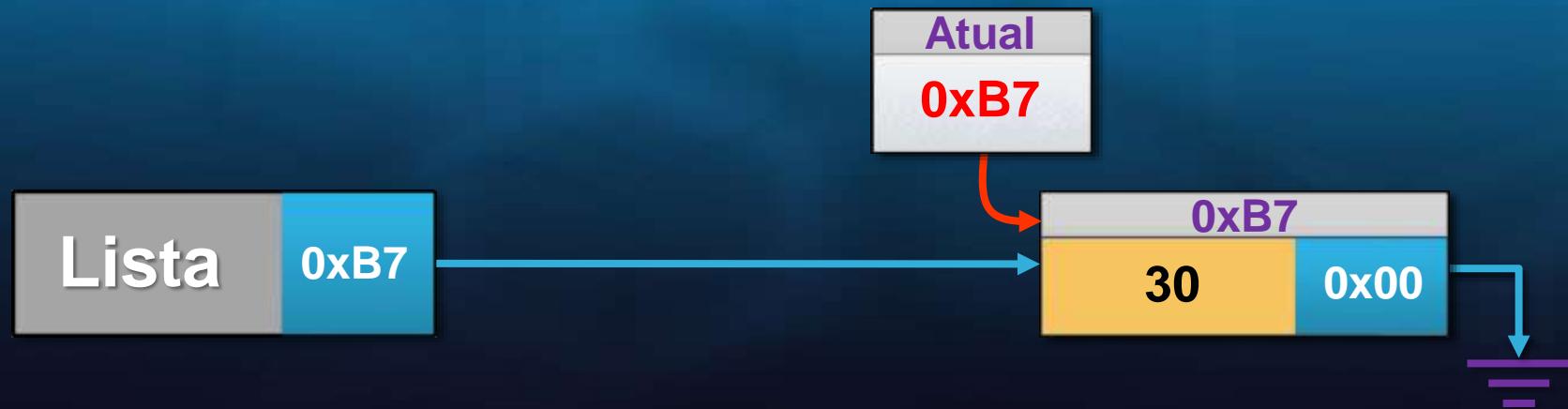
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



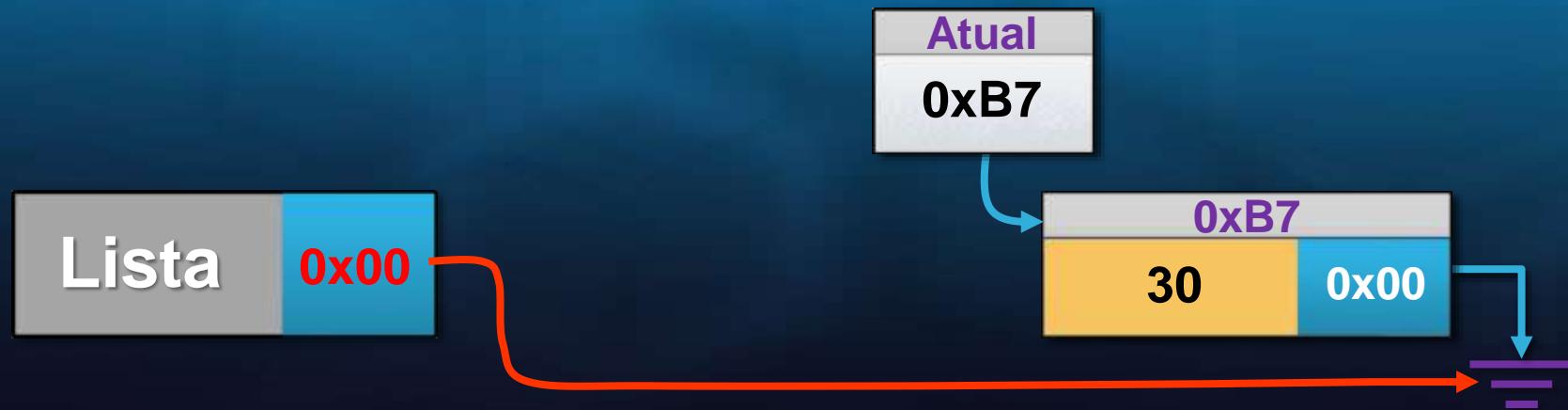
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



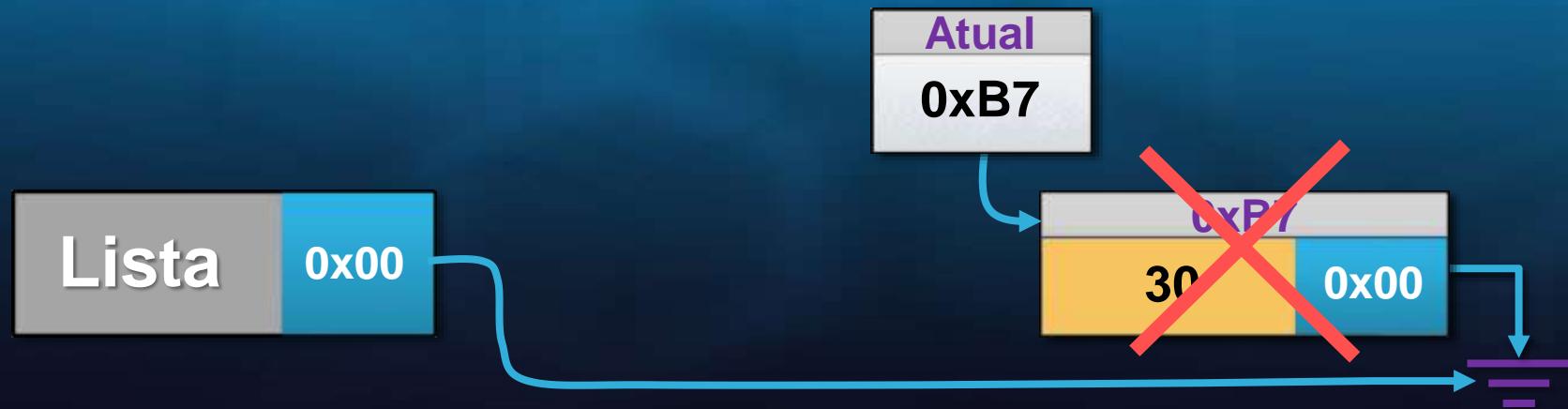
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



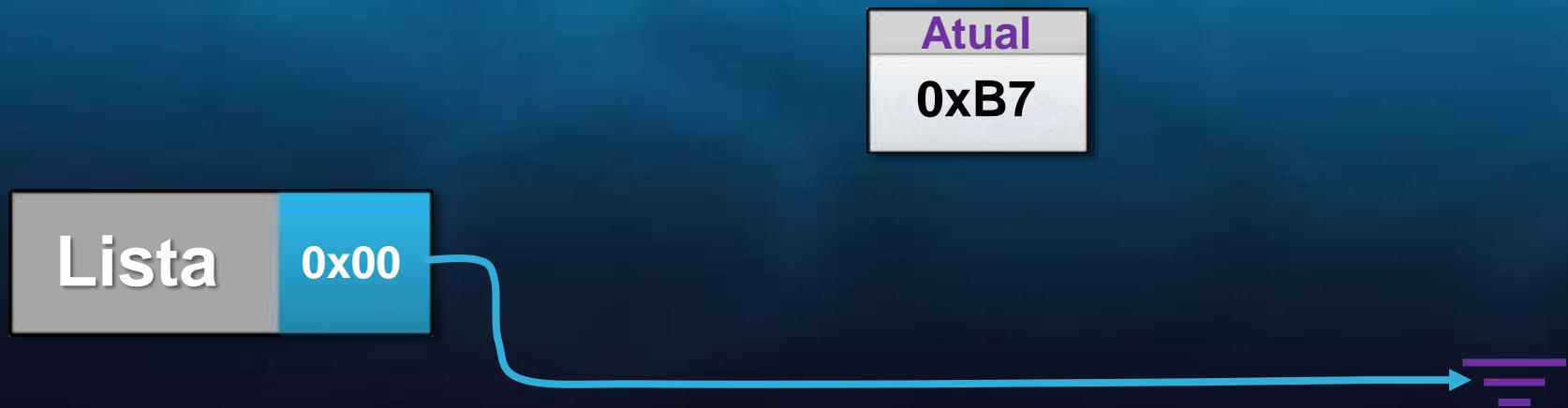
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



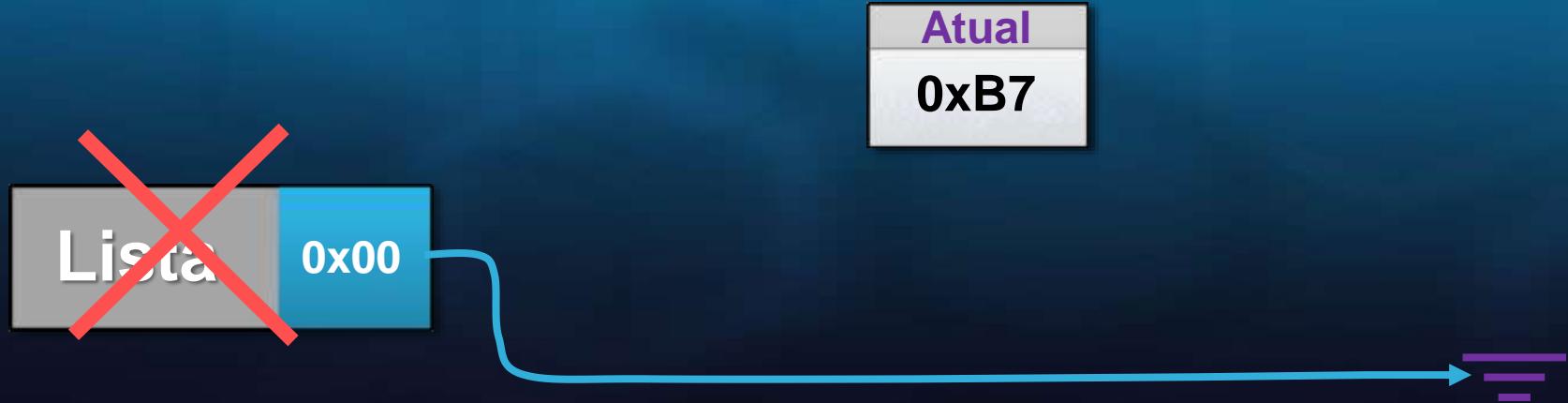
```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



```
No *ptrNoAtual;  
  
// Exclui cada Nó da lista  
while (ptrLista->inicio != NULL)  
{  
    ptrNoAtual = ptrLista->inicio;  
  
    ptrLista->inicio = ptrNoAtual->proxNo;  
  
    delete ptrNoAtual;  
}  
  
delete ptrLista;
```



```
No *ptrNoAtual;

// Exclui cada Nó da lista
while (ptrLista->inicio != NULL)
{
    ptrNoAtual = ptrLista->inicio;

    ptrLista->inicio = ptrNoAtual->proxNo;

    delete ptrNoAtual;
}

delete ptrLista;
```

Exercício

- Criar uma lista de produtos para armazenar os seguintes dados: código do produto, descrição, preço e fabricante.
- A lista deve ter as funções de criação, inserção, exibição e liberação da lista.

Exercício

```
// Dados sobre o PRODUTO
struct Dados {
    int codProduto;
    string descricao;
    float preco;
    string fabricante;
};
```

```
bool inserirListaInicio(Lista* ptrLista, int codProduto, string descricao,
                      float preco, string fabricante) {
    No* ptrNoNovo;

    // Se a lista NÃO foi criada
    if (ptrLista == NULL) {
        cout << "A lista não está criada!" << endl;
        return false;
    }

    // Cria o novo nó
    ptrNoNovo = new No;

    if (ptrNoNovo == NULL) {
        cout << "Memória insuficiente!" << endl;
        return false;
    }

    ptrNoNovo->dados.codProduto = codProduto;
    ptrNoNovo->dados.descricao = descricao;
    ptrNoNovo->dados.preco = preco;
    ptrNoNovo->dados.fabricante = fabricante;
    ptrNoNovo->proxNo = ptrLista->inicio;

    ptrLista->inicio = ptrNoNovo;
    ptrLista->qtdNo++; // Incrementa a quantidade de Nós

    return true;
}
```

```
void exibirLista(Lista* ptrLista) {
    No* ptrNoAtual;

    //Se a lista NÃO foi criada
    if (ptrLista == NULL) {
        cout << "A lista não está criada!" << endl;
        return;
    }

    //Se não tiver nenhum Nó na lista
    if (ptrLista->inicio == NULL) {
        cout << "A lista esta vazia!" << endl;
        return;
    }

    ptrNoAtual = ptrLista->inicio;

    while (ptrNoAtual != NULL) {

        cout << "Código Produto: " << ptrNoAtual->dados.codProduto << endl;
        cout << "Descrição: " << ptrNoAtual->dados.descricao << endl;
        cout << "Preço: " << ptrNoAtual->dados.preco << endl ;
        cout << "Fabricante: " << ptrNoAtual->dados.fabricante << endl << endl;

        ptrNoAtual = ptrNoAtual->proxNo;
    }

    cout << endl;
}
```

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    Lista* pLista;
    pLista = criarLista();

    inserirListaInicio(pLista, 100, "Sabão em Pó", 5.5, "Omo");

    int codProduto = 200;
    string descricao = "Picanha";
    float preco = 45.3;
    string fabricante = "Frigorífico Ex-Touro";

    inserirListaInicio(pLista, codProduto, descricao, preco, fabricante);
    inserirListaInicio(pLista, 300, "Guarana", 8.0, "Antartica");

    exibirLista(pLista);

    liberarLista(pLista);
    system("pause");
    return 0;
}
```

Exemplos

- 1)** Criar uma função que exiba os produtos que sejam **maiores** que R\$ 5,00.

- 2)** Crie uma segunda função que receba a lista e um preço de produto. A função deverá procurar os produtos que estejam abaixo do preço informado.

- 3)** Crie uma função que receba como parâmetro a lista e o nome de um produto. A função deverá informar se o produto se encontra ou não na lista de produtos

Exemplos

- Dados utilizados para testes das funções.

```
inserirListaInicio(pLista, 100, "Sabão em Pó", 3.5, "Omo");  
  
int codProduto = 200;  
string descricao = "Picanha";  
float preco = 45.3;  
string fabricante = "Frigorífico Ex-Touro";  
  
inserirListaInicio(pLista, codProduto, descricao, preco, fabricante);  
inserirListaInicio(pLista, 300, "Guarana", 6.0, "Antartica");  
inserirListaInicio(pLista, 400, "Coca-cola", 7.5, "Refrescos Ipiranga");  
inserirListaInicio(pLista, 500, "Alface", 3.0, "Varejão X");  
  
exibirLista(pLista);
```

Exemplo 1 – Parte 1

- Criação da função **buscarProdMaiorQue5Reais(...)**

```
void buscarProdMaiorQue5Reais(Lista* ptrLista) {
    No* ptrNoAtual;

    //Se a lista NÃO foi criada
    if (ptrLista == NULL) {
        cout << "A lista não está criada!" << endl;
        return;
    }

    //Se não tiver nenhum Nó na lista
    if (ptrLista->inicio == NULL) {
        cout << "A lista esta vazia!" << endl;
        return;
    }
```

Exemplo 1 – Parte 2

- Criação da função **buscarProdMaiorQue5Reais(...)**

```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    if (ptrNoAtual->dados.preco > 5) {
        cout << "Código Produto: " << ptrNoAtual->dados.codProduto << endl;
        cout << "Descrição: " << ptrNoAtual->dados.descricao << endl;
        cout << "Preço: " << ptrNoAtual->dados.preco << endl;
        cout << "Fabricante: " << ptrNoAtual->dados.fabricante << endl << endl;
    }
    ptrNoAtual = ptrNoAtual->proxNo;
}
cout << endl;
```

Exemplo 1 – Parte 3

- Chamada da função **buscarProdMaiorQue5Reais(...)**

```
buscarProdMaiorQue5Reais(pLista);
```

Exemplo 2 – Parte 1

- Criação da função **buscarMenorPreco (...)**

```
void buscarMenorPreco(Lista* ptrLista, float vlrPreco) {  
    No* ptrNoAtual;  
  
    //Se a lista NÃO foi criada  
    if (ptrLista == NULL) {  
        cout << "A lista não está criada!" << endl;  
        return;  
    }  
  
    //Se não tiver nenhum Nó na lista  
    if (ptrLista->inicio == NULL) {  
        cout << "A lista esta vazia!" << endl;  
        return;  
    }  
}
```

Exemplo 2 – Parte 2

- Criação da função **buscarMenorPreco (...)**

```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    if (ptrNoAtual->dados.preco < vlrPreco) {
        cout << "Código Produto: " << ptrNoAtual->dados.codProduto << endl;
        cout << "Descrição: " << ptrNoAtual->dados.descricao << endl;
        cout << "Preço: " << ptrNoAtual->dados.preco << endl;
        cout << "Fabricante: " << ptrNoAtual->dados.fabricante << endl << endl;
    }

    ptrNoAtual = ptrNoAtual->proxNo;
}

cout << endl;
}
```

Exemplo 2 – Parte 3

- Chamada da função **buscarMenorPreco(...)**

```
buscarMenorPreco(pLista, 6.5);
```

Exemplo 3 – Parte 1

- Criação da função **buscarProduto(...)**

```
void buscarProduto(Lista* ptrLista, string descricaoProd) {  
    No* ptrNoAtual;  
  
    //Se a lista NÃO foi criada  
    if (ptrLista == NULL) {  
        cout << "A lista não está criada!" << endl;  
        return;  
    }  
  
    //Se não tiver nenhum Nó na lista  
    if (ptrLista->inicio == NULL) {  
        cout << "A lista esta vazia!" << endl;  
        return;  
    }
```

Exemplo 3 – Parte 2

- Criação da função **buscarProduto(...)**

```
ptrNoAtual = ptrLista->inicio;

while (ptrNoAtual != NULL) {

    if (ptrNoAtual->dados.descricao == descricaoProd) {
        cout << "O produto se encontra no estoque." << endl;
        return;
    }

    ptrNoAtual = ptrNoAtual->proxNo;
}

cout << "O produto não se encontra em estoque no momento." << endl;
cout << endl;
```

Exemplo 3 – Parte 3

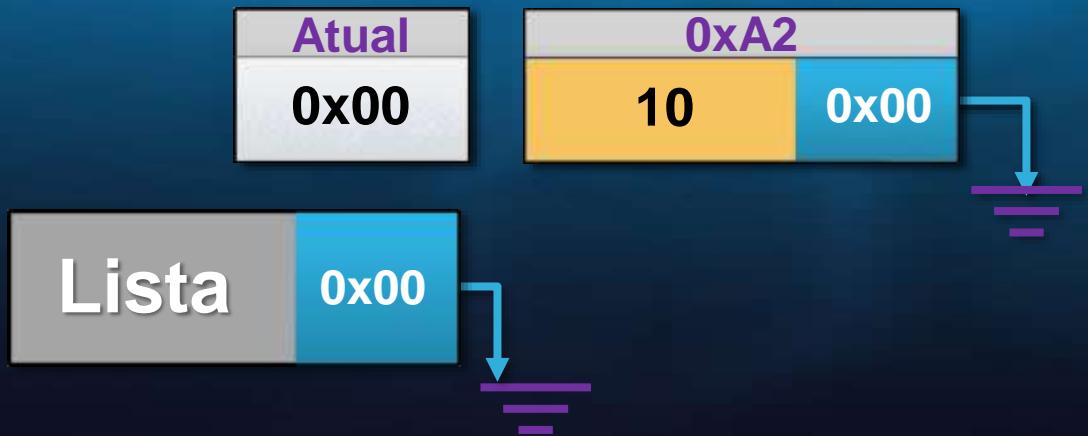
- Chamada da função **buscarProduto(...)**

```
buscarProduto(pLista, "Veja limpeza pesada");  
buscarProduto(pLista, "Alface");
```

INSERIR NO FINAL

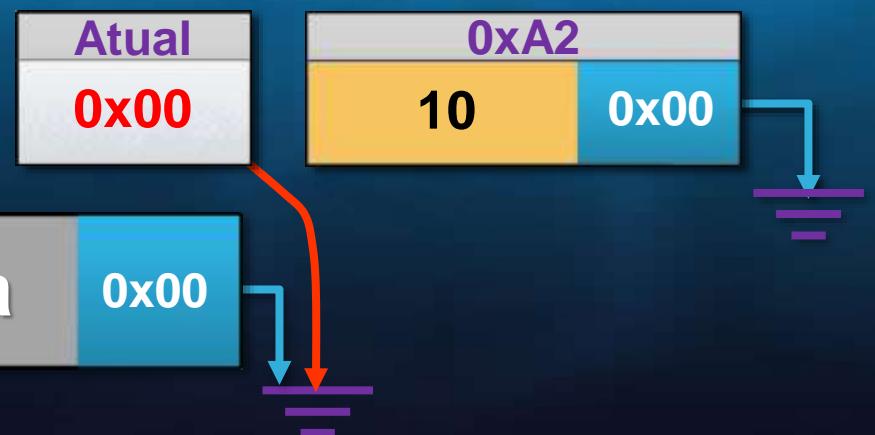
SIMULAÇÃO 1

Inserir no final – Simulação 1



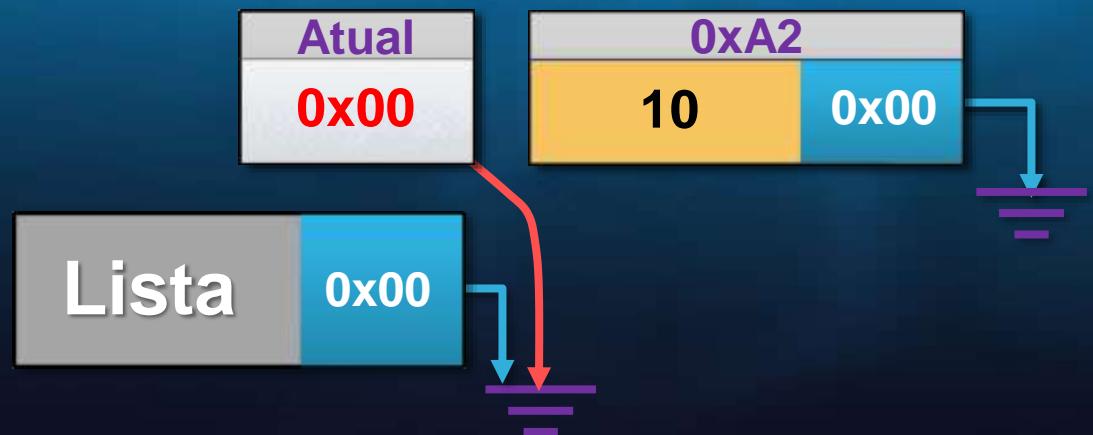
```
ptrNoAtual = ptrLista->inicio;
```

```
// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



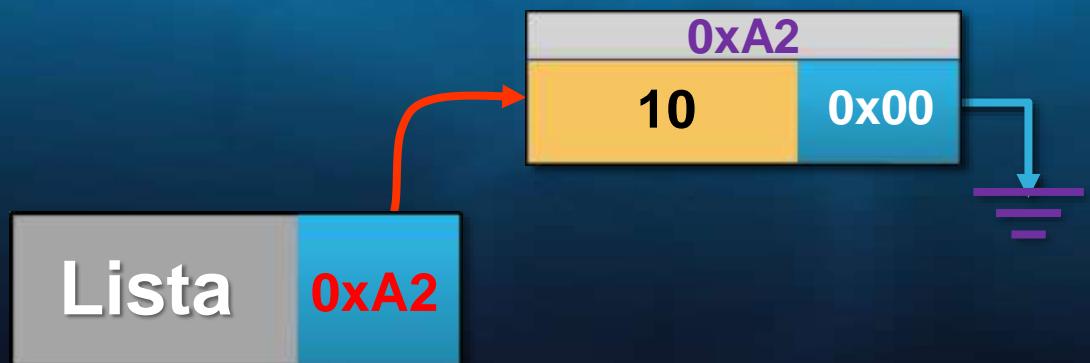
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



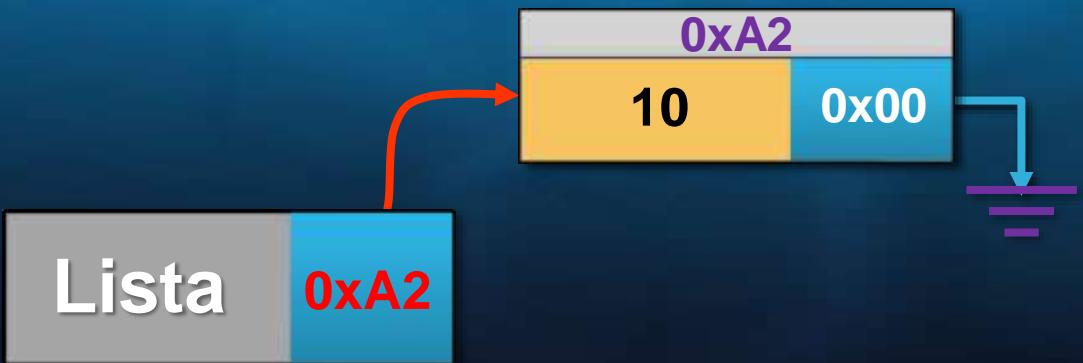
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



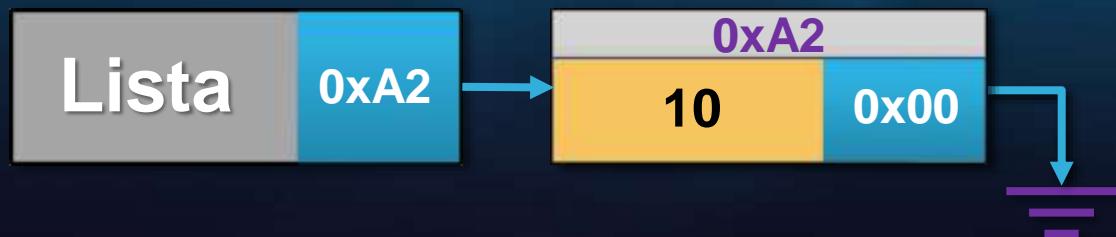
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



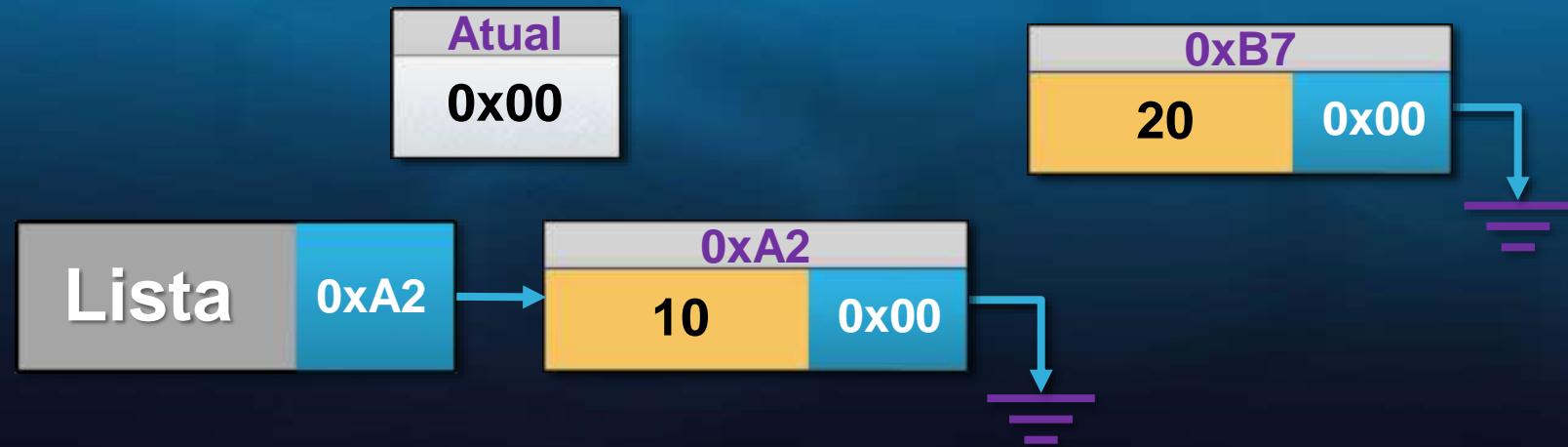
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



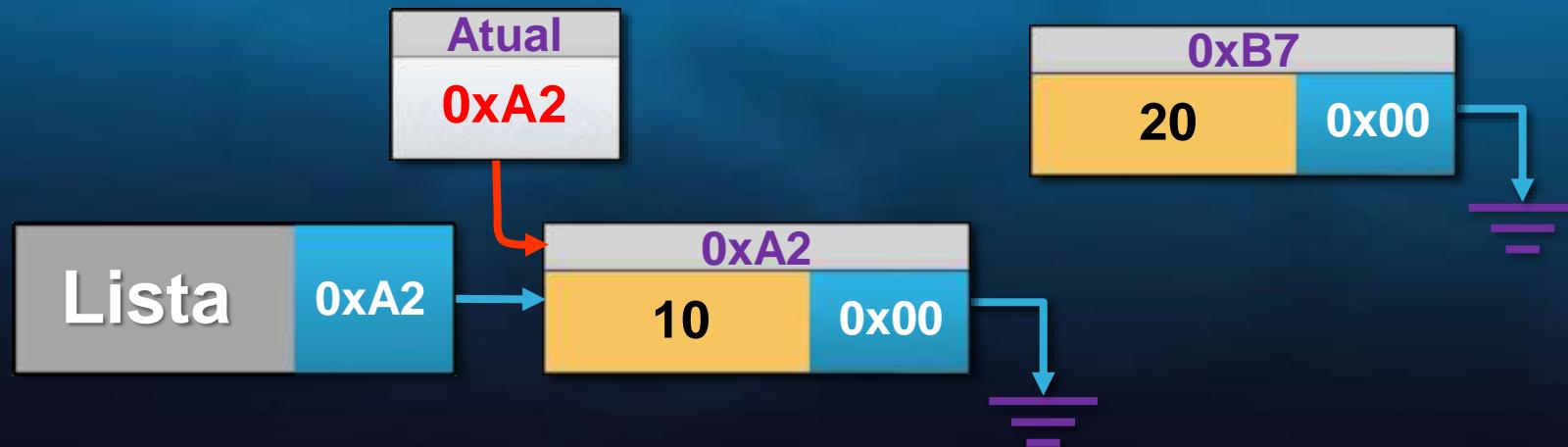
INserir no final

simulação 2



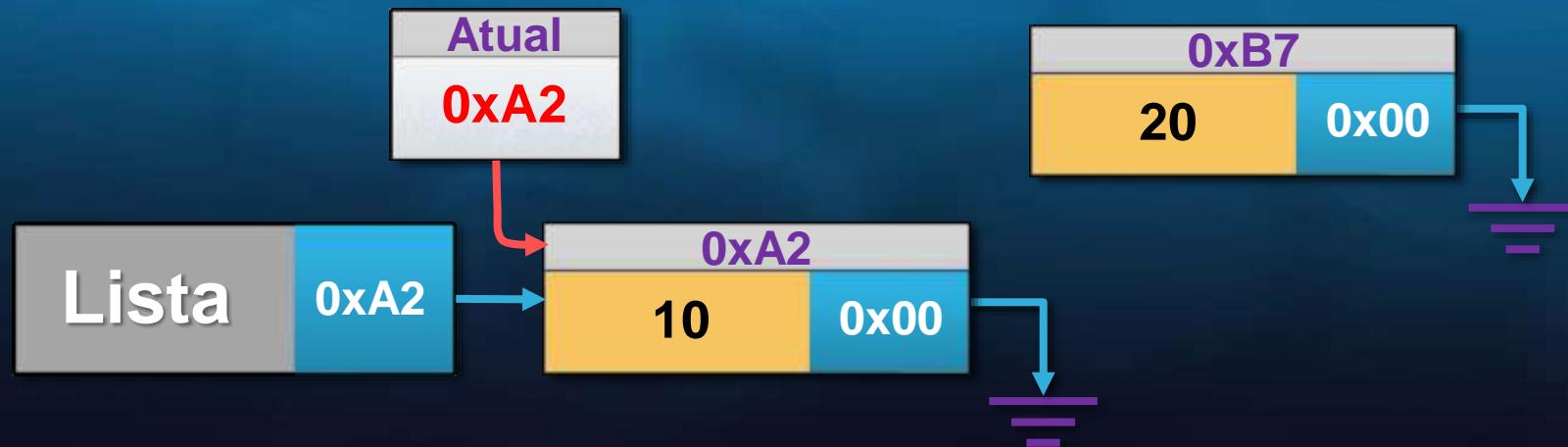
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



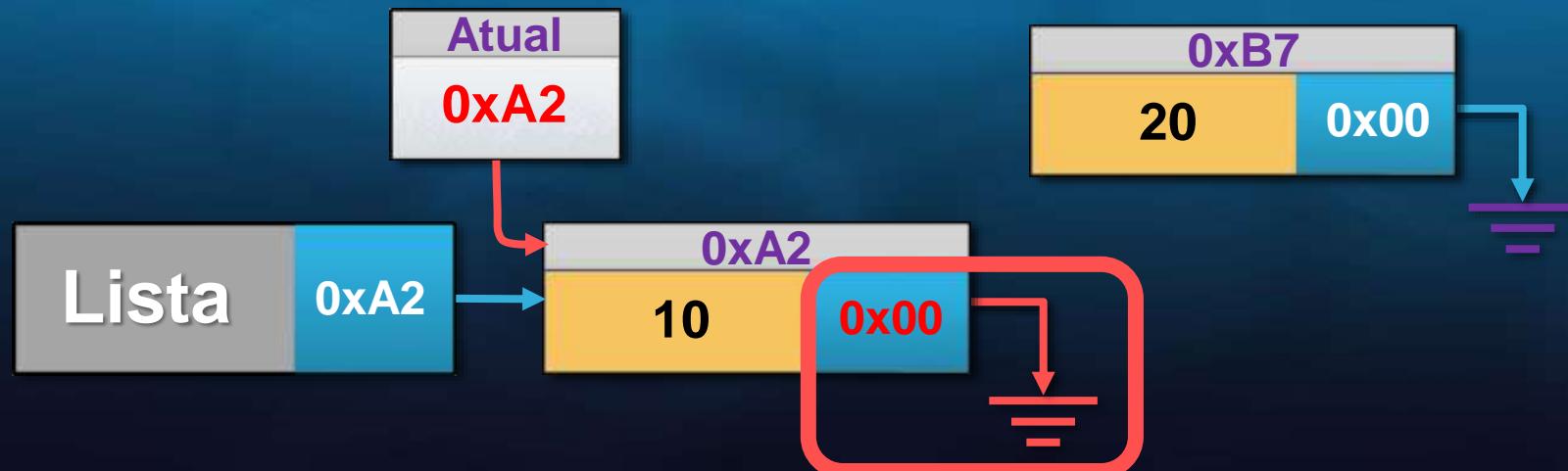
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```

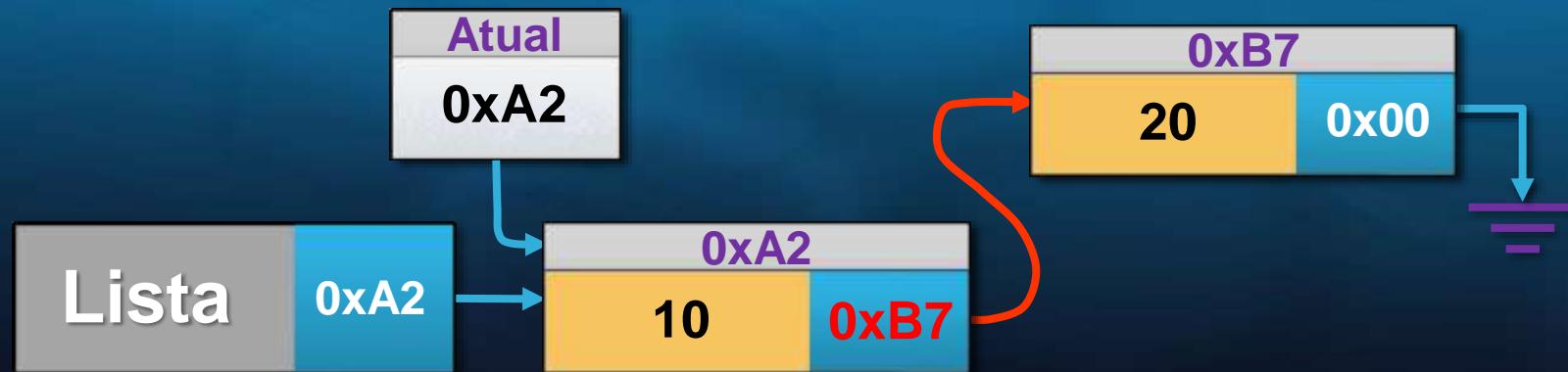


```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```

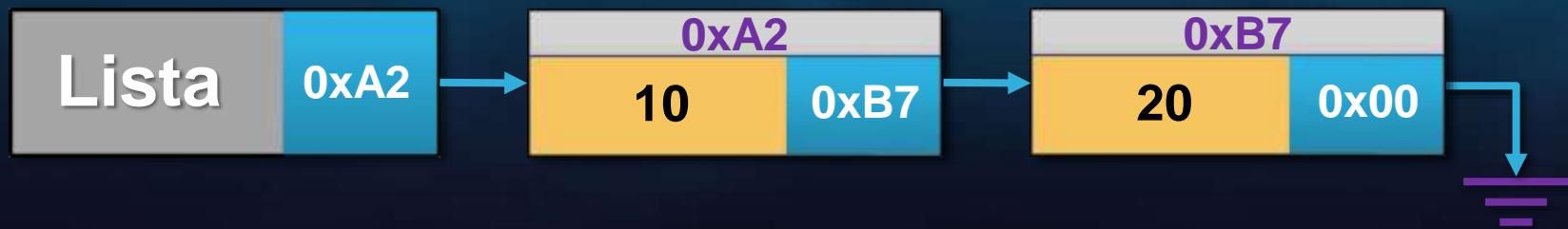


```
ptrNoAtual = ptrLista->inicio;  
  
// Se não houver nenhum nó na lista  
if (ptrNoAtual == NULL) {  
    ptrLista->inicio = ptrNoNovo;  
}  
else {  
    // Localiza o último nó  
    while (ptrNoAtual->proxNo != NULL) {  
        ptrNoAtual = ptrNoAtual->proxNo;  
    }  
    ptrNoAtual->proxNo = ptrNoNovo;  
}
```



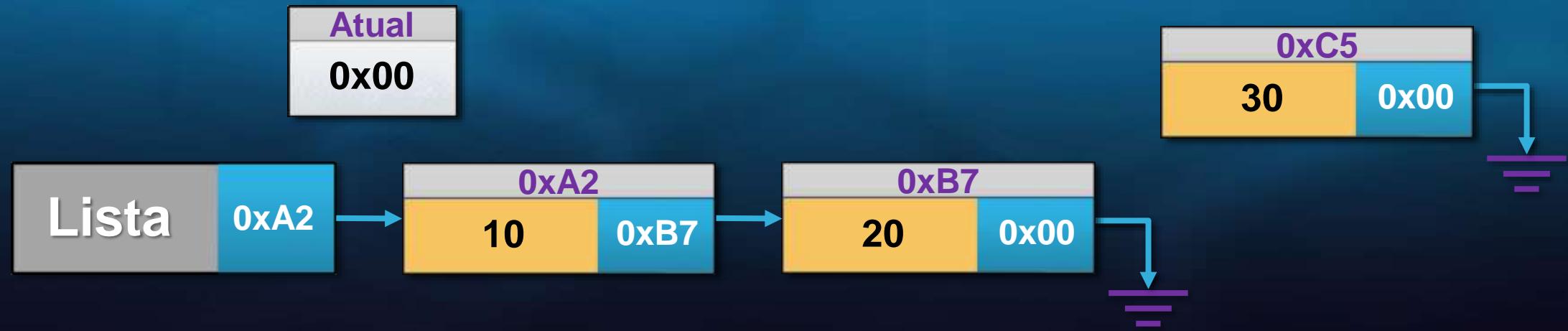
```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



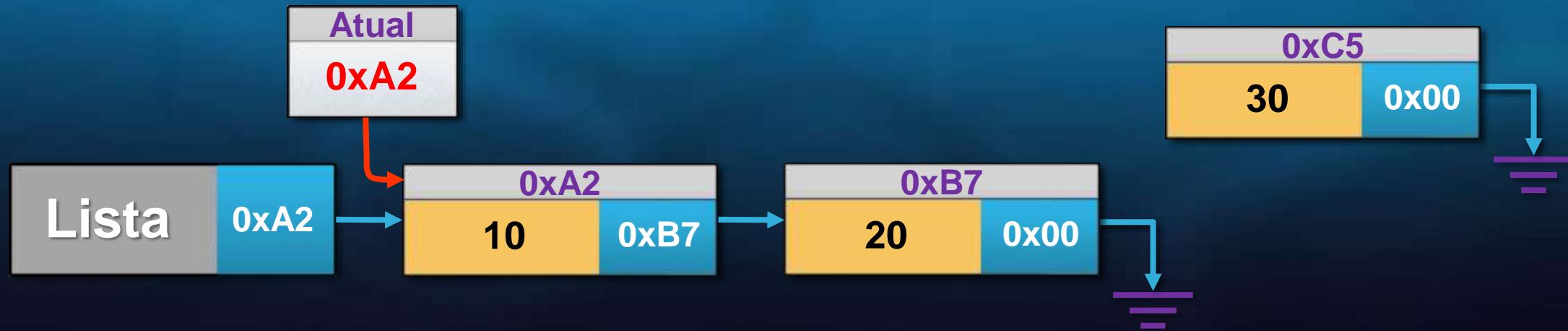
INserir no final

Simulação 3



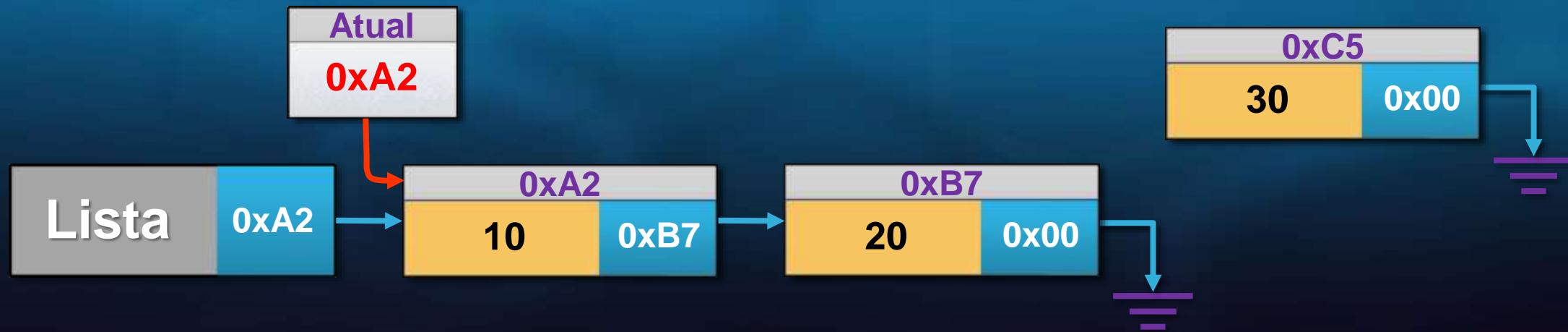
```
ptrNoAtual = ptrLista->inicio;
```

```
// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```

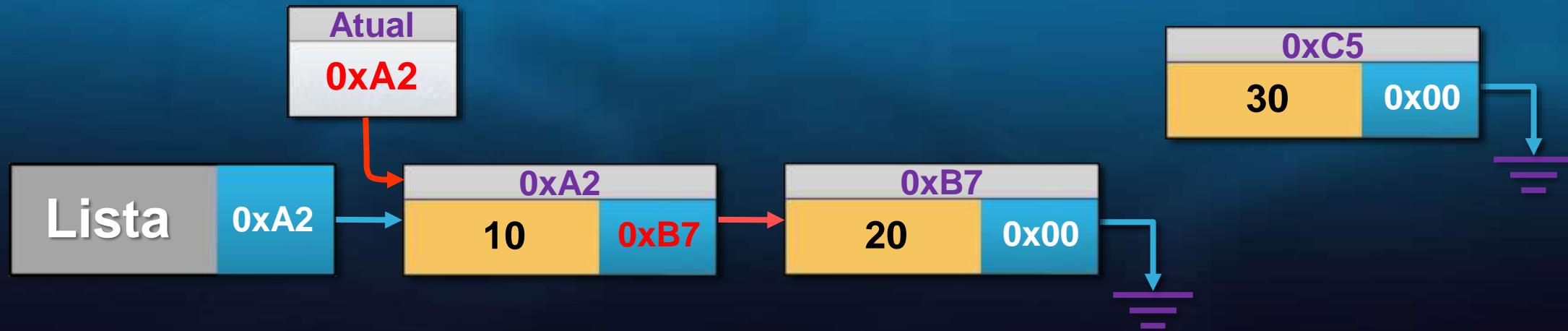


```

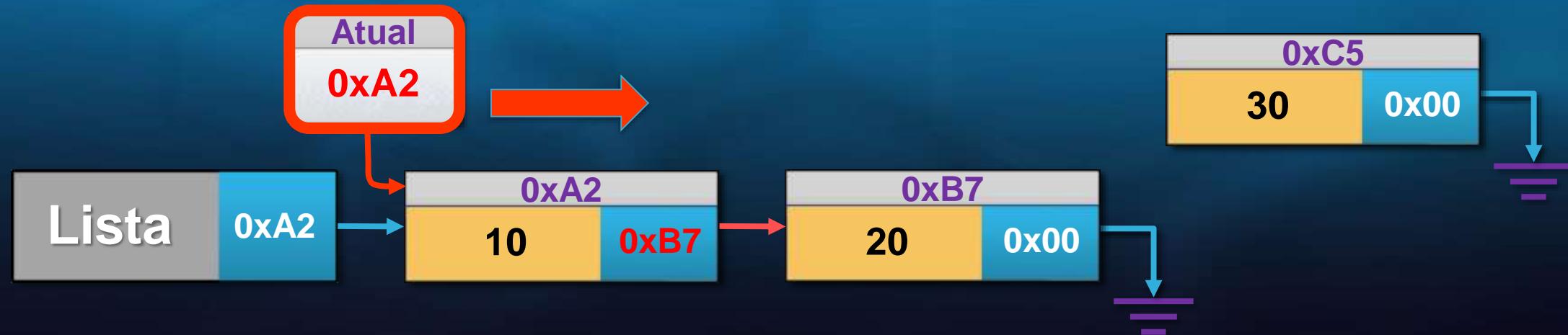
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}

```

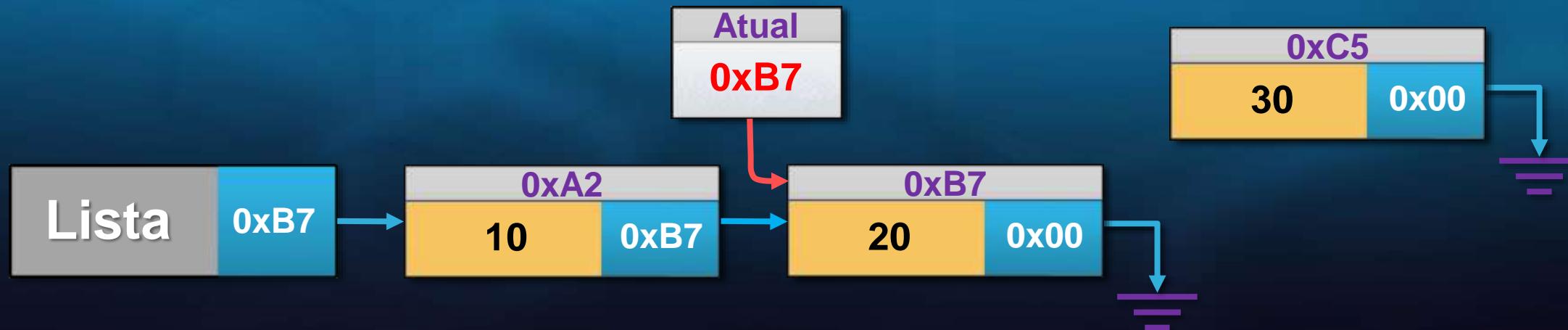


```
ptrNoAtual = ptrLista->inicio;  
  
// Se não houver nenhum nó na lista  
if (ptrNoAtual == NULL) {  
    ptrLista->inicio = ptrNoNovo;  
}  
else {  
    // Localiza o último nó  
    while (ptrNoAtual->proxNo != NULL) {  
        ptrNoAtual = ptrNoAtual->proxNo;  
    }  
    ptrNoAtual->proxNo = ptrNoNovo;  
}
```



```
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```

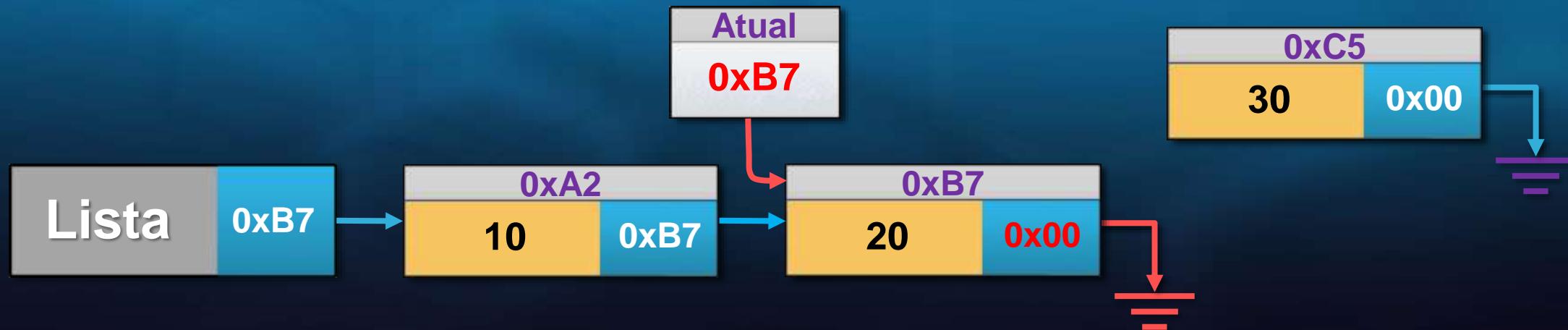


```

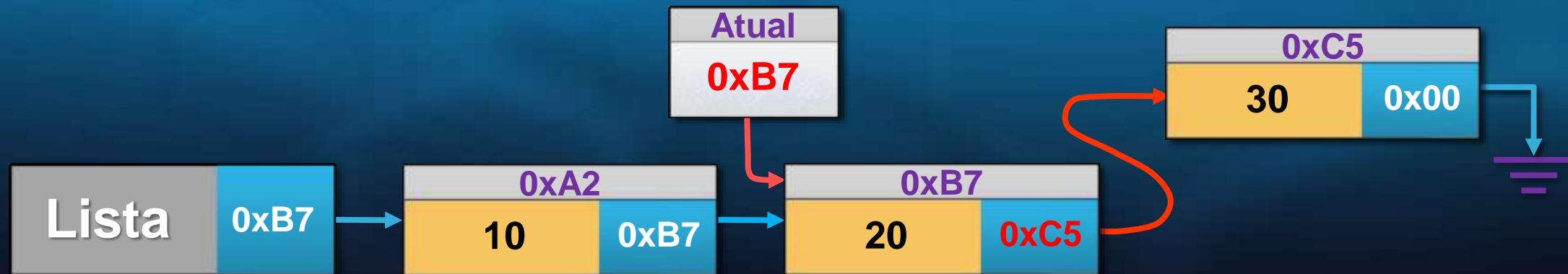
ptrNoAtual = ptrLista->inicio;

// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}

```

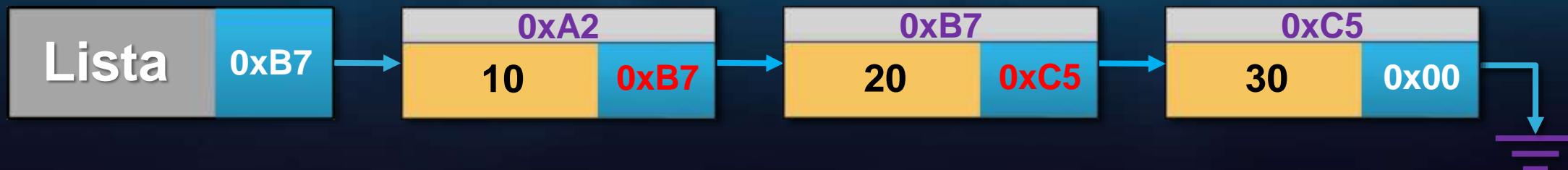


```
ptrNoAtual = ptrLista->inicio;  
  
// Se não houver nenhum nó na lista  
if (ptrNoAtual == NULL) {  
    ptrLista->inicio = ptrNoNovo;  
}  
else {  
    // Localiza o último nó  
    while (ptrNoAtual->proxNo != NULL) {  
        ptrNoAtual = ptrNoAtual->proxNo;  
    }  
    ptrNoAtual->proxNo = ptrNoNovo;  
}
```



```
ptrNoAtual = ptrLista->inicio;

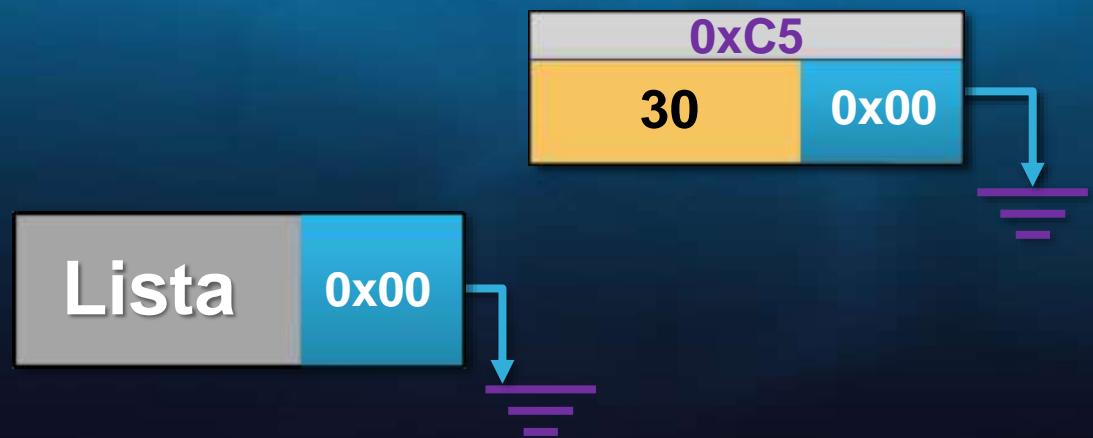
// Se não houver nenhum nó na lista
if (ptrNoAtual == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else {
    // Localiza o último nó
    while (ptrNoAtual->proxNo != NULL) {
        ptrNoAtual = ptrNoAtual->proxNo;
    }
    ptrNoAtual->proxNo = ptrNoNovo;
}
```



INSERIR ORDENADO NA LISTA

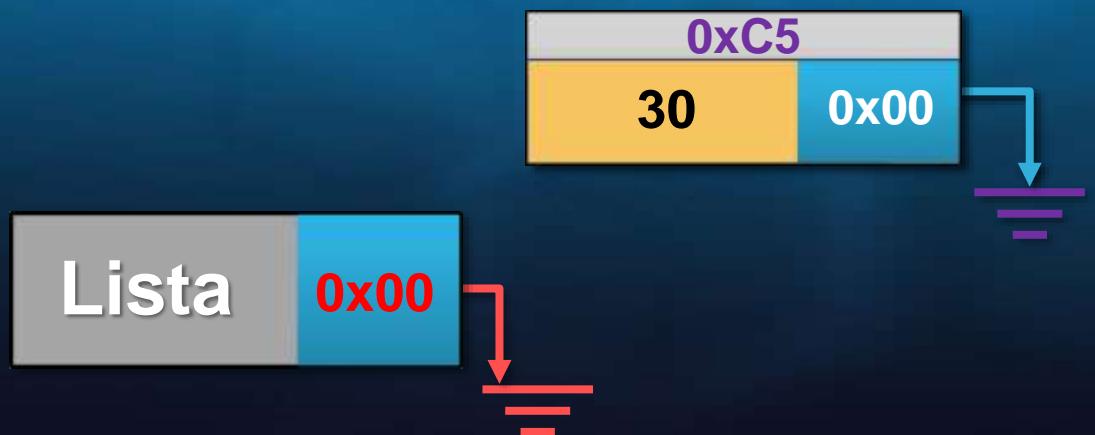
SIMULAÇÃO 1

Inserir ordenado na lista – Simulação 1



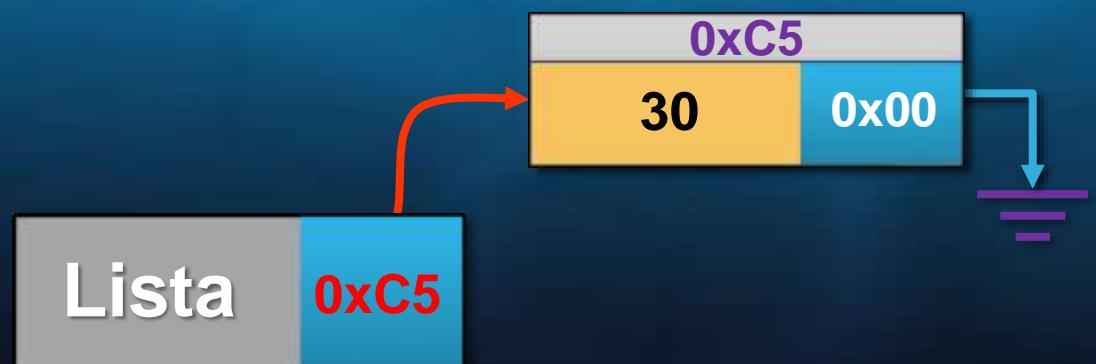
```
//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}
```



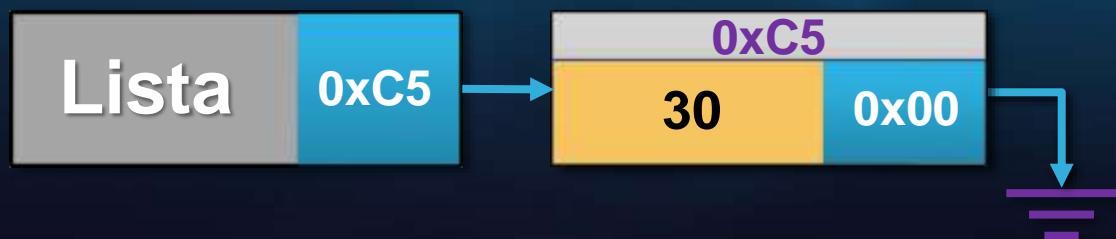
```
//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}
```



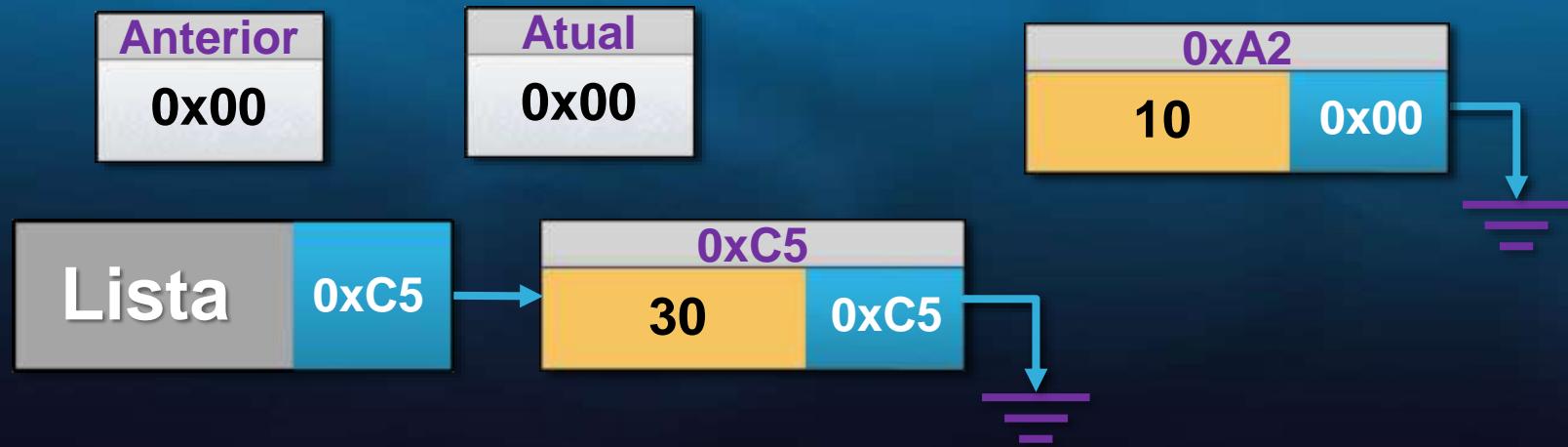
```
//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}
```



INSERIR ORDENADO NA LISTA

SIMULAÇÃO 2



```

ptrNoNovo->proxNo = NULL;

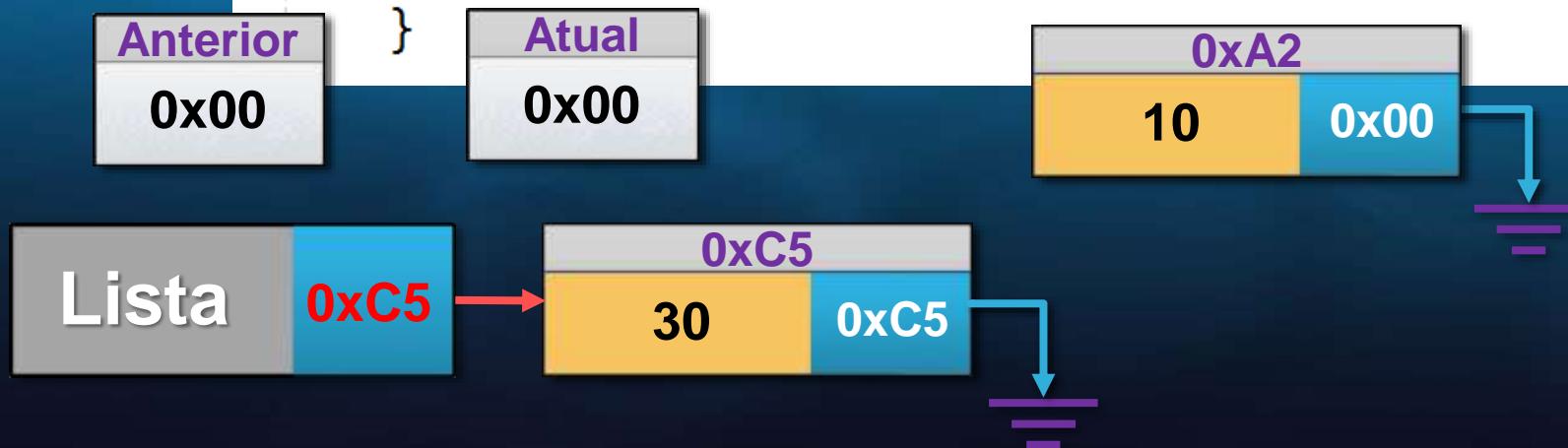
//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;
}

```

```

// Localiza a posição de inserção
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

```



```

ptrNoNovo->proxNo = NULL;

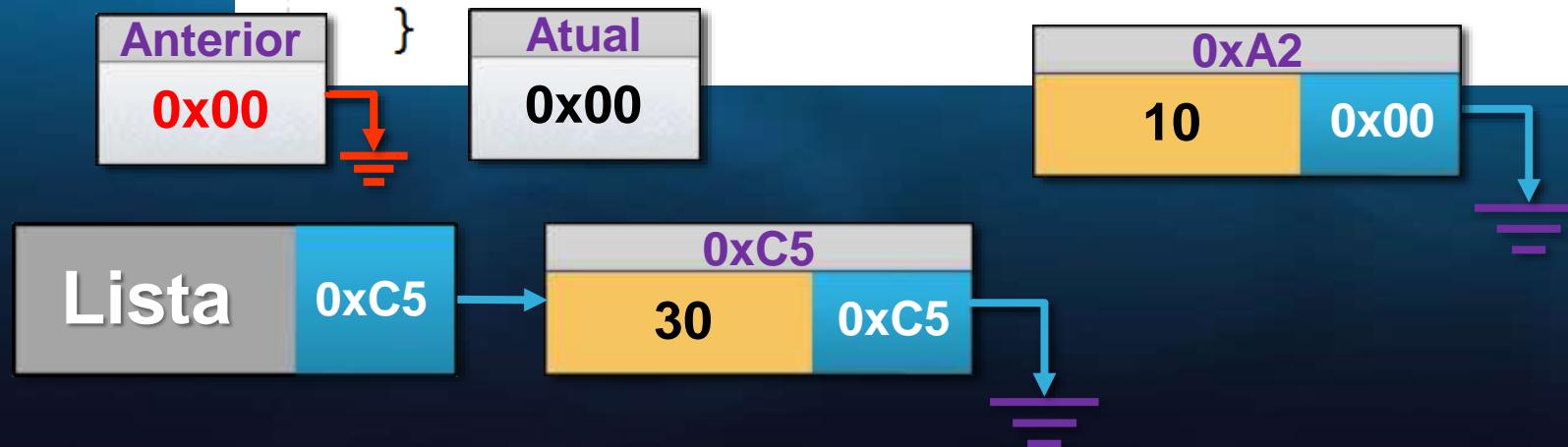
//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;
}

```

```

// Localiza a posição de inserção
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

```



```

ptrNoNovo->proxNo = NULL;

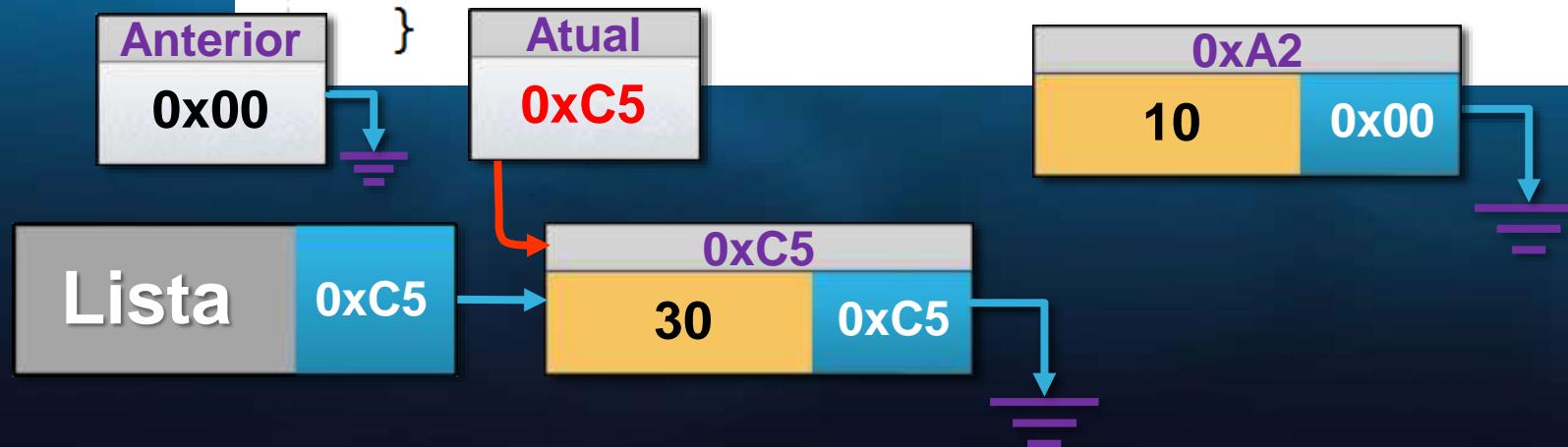
//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;
}

```

```

// Localiza a posição de inserção
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

```



```

ptrNoNovo->proxNo = NULL;

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;
}

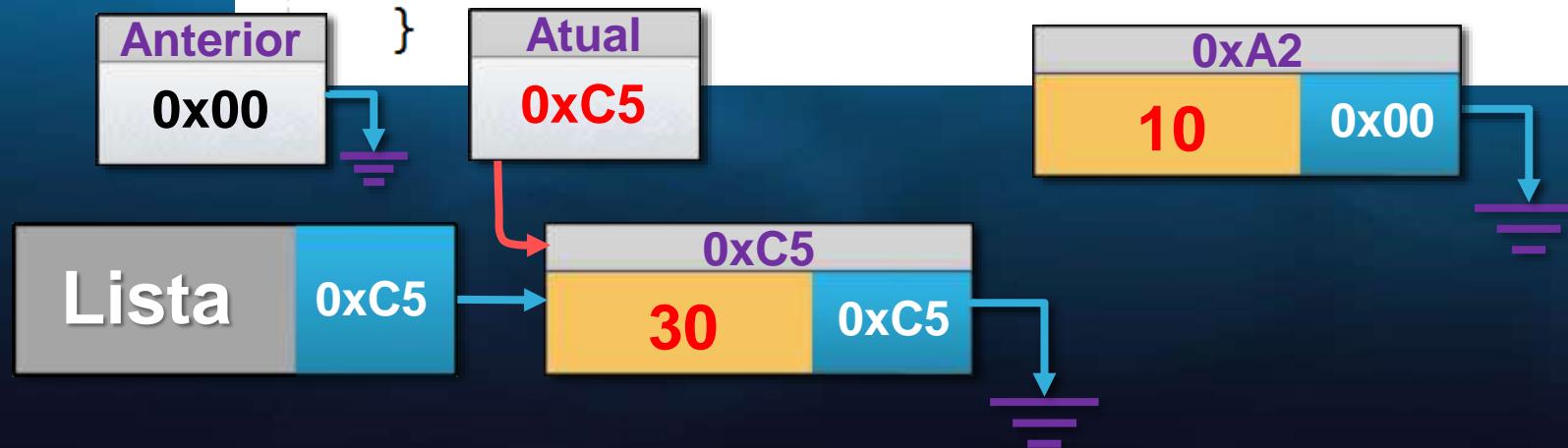
```

// Localiza a posição de inserção

```

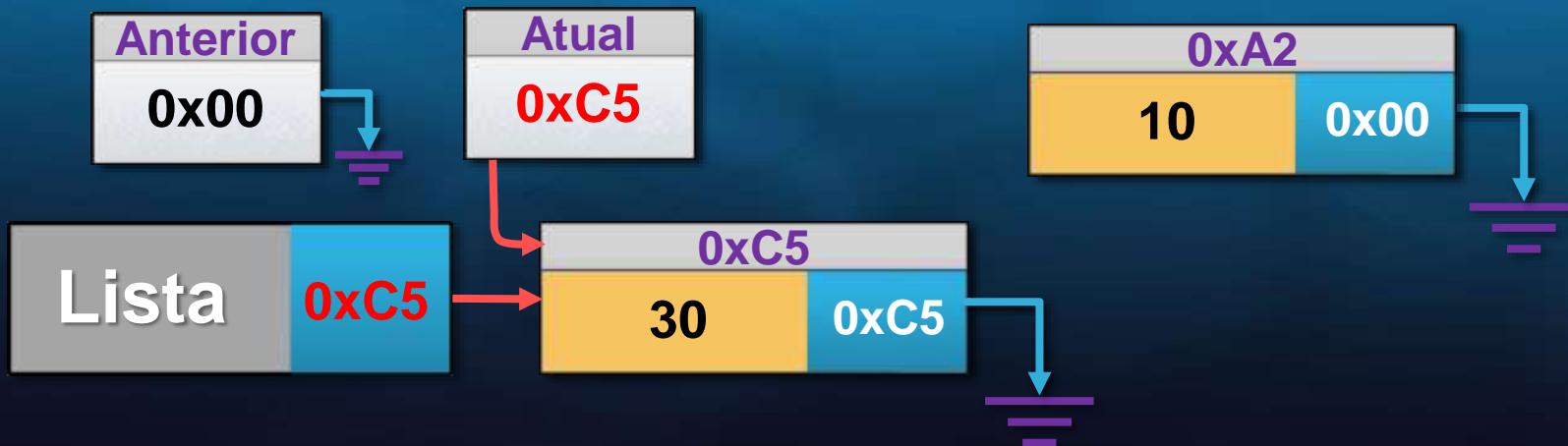
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

```



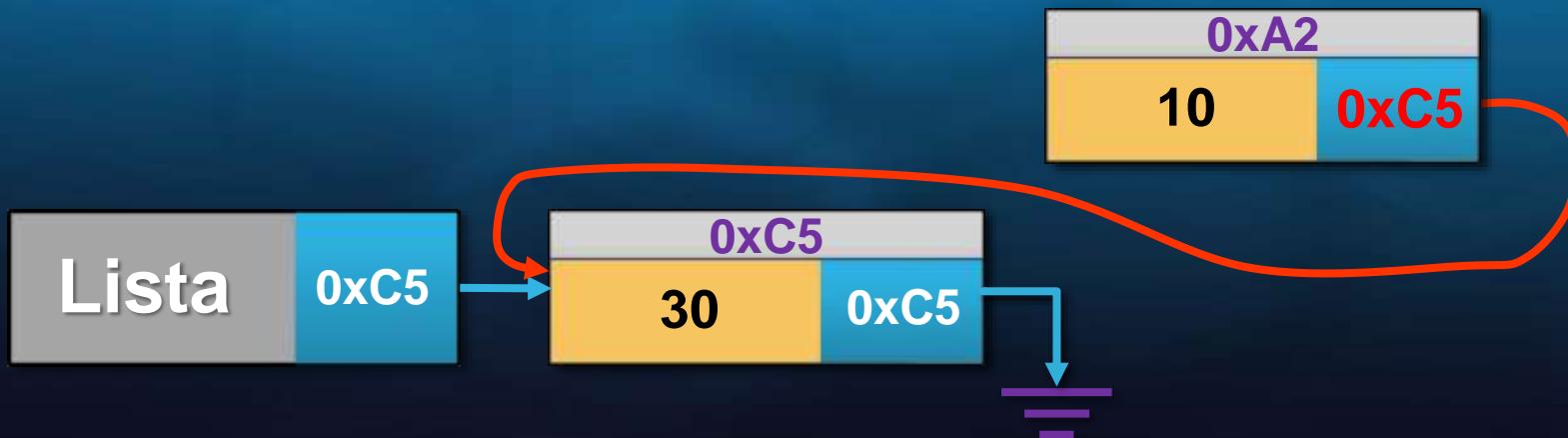
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



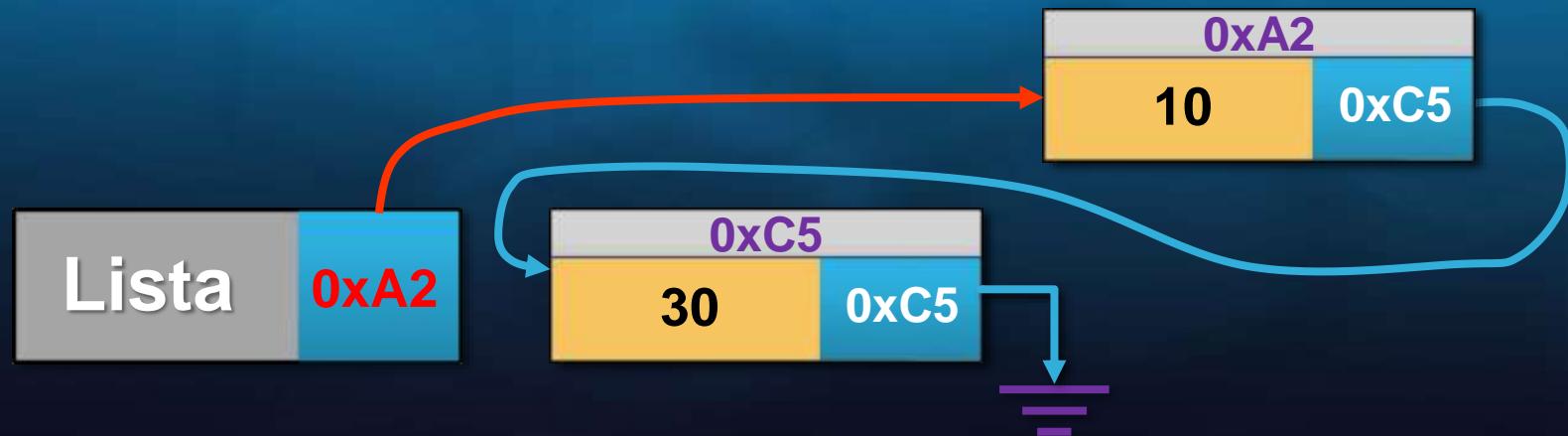
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



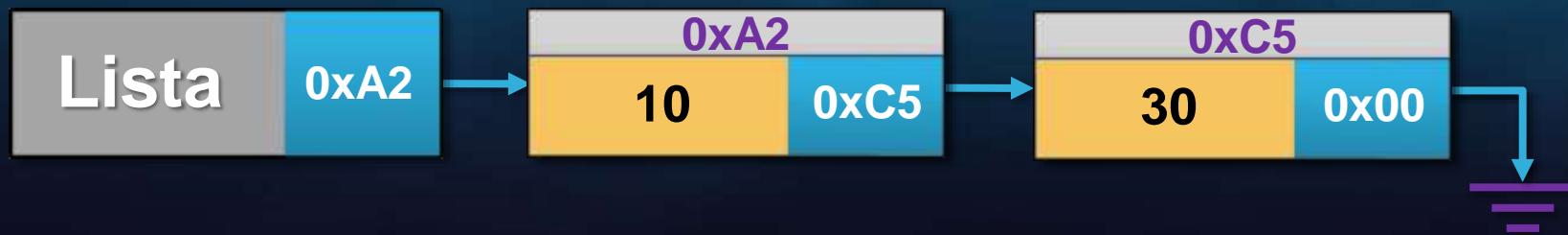
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



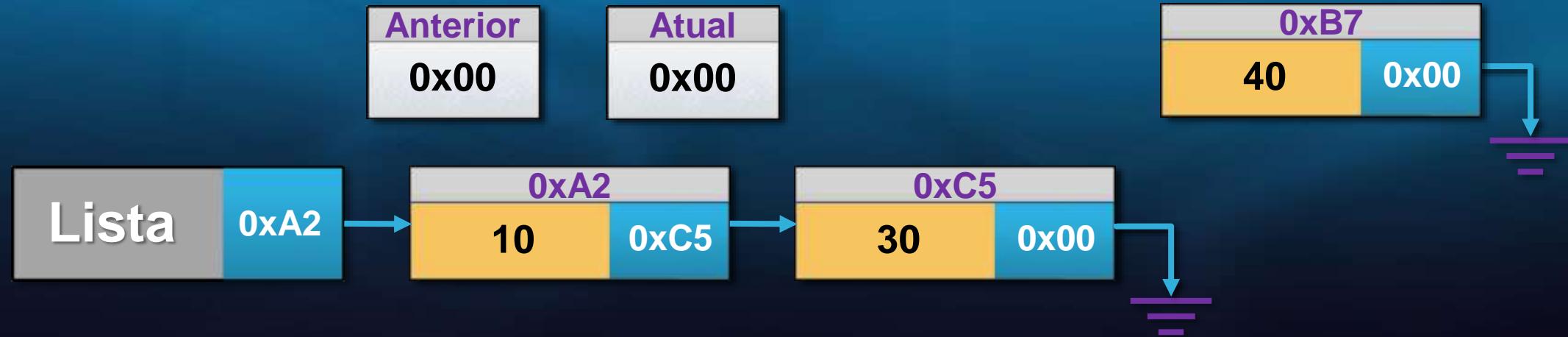
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



INSERIR ORDENADO NA LISTA

SIMULAÇÃO 3

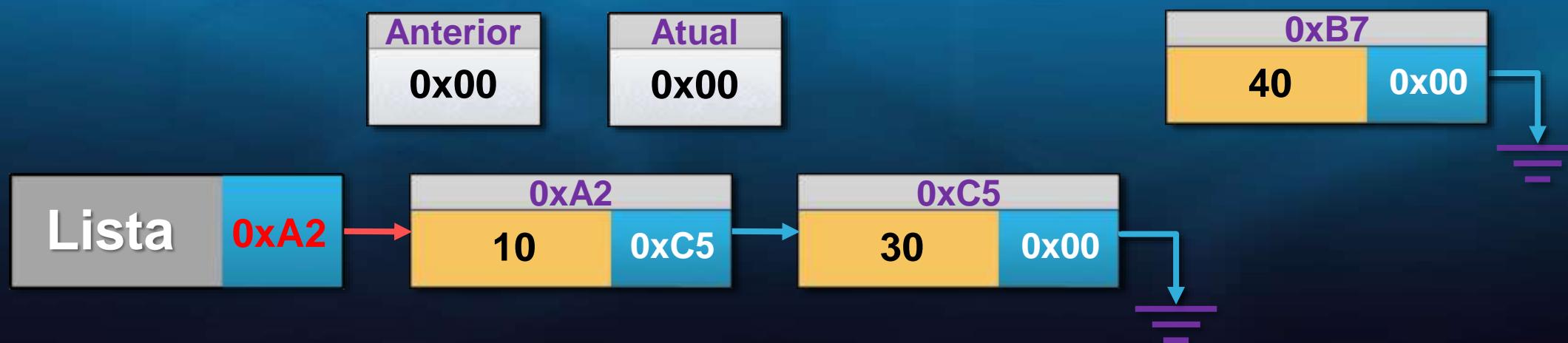


```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

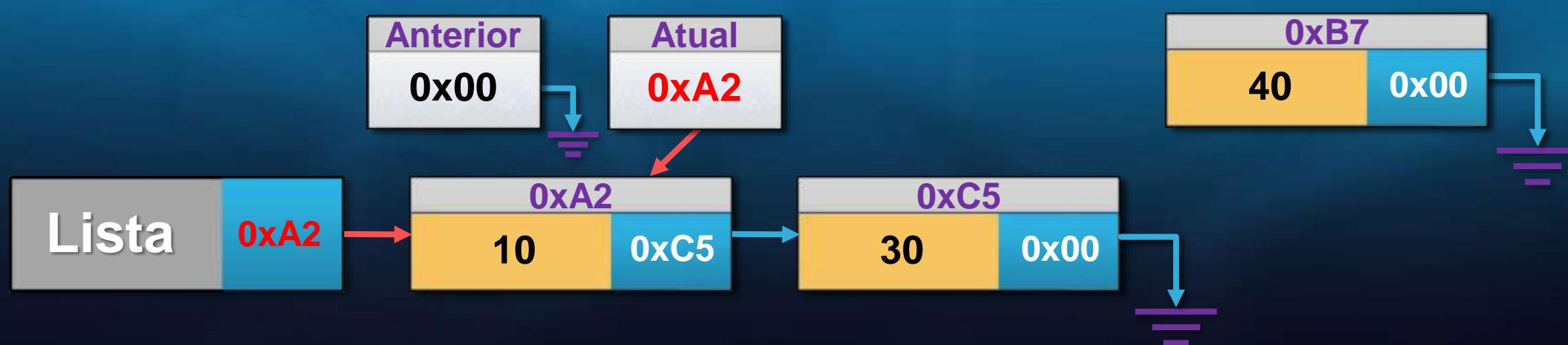


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



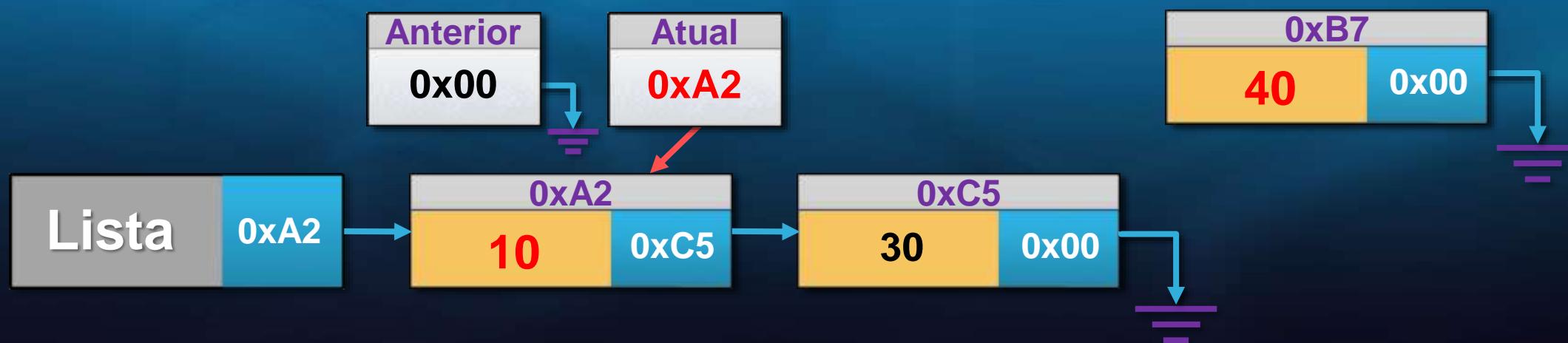
```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

10 < 40

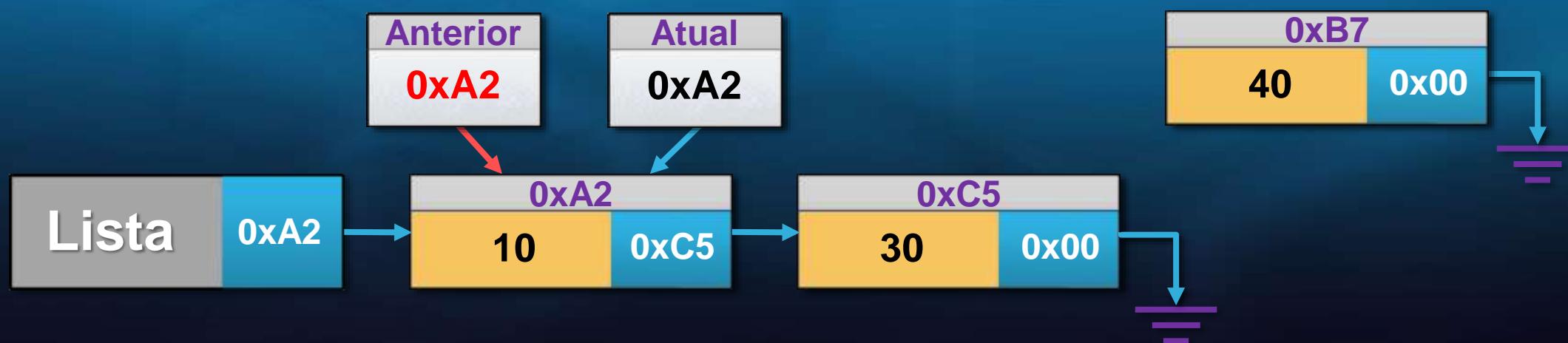


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

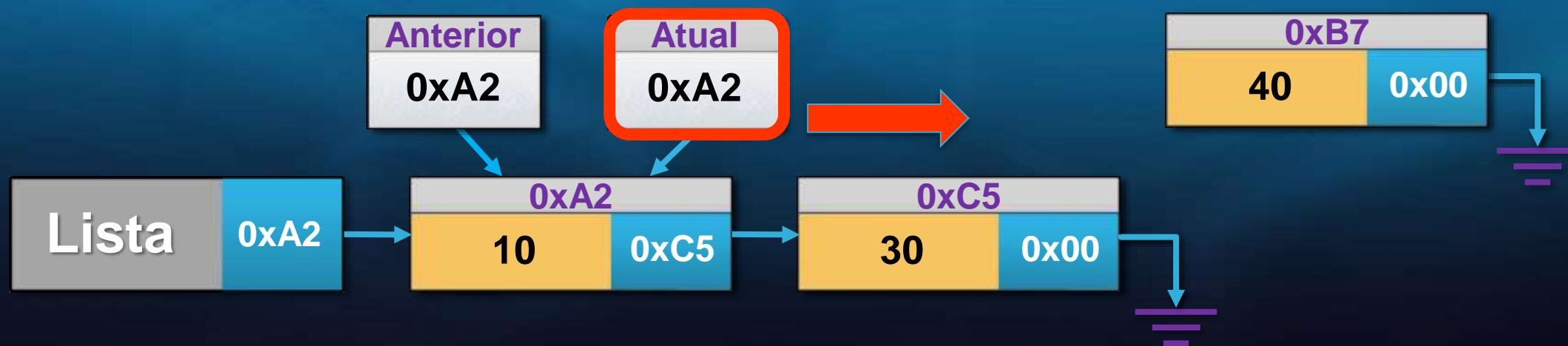


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

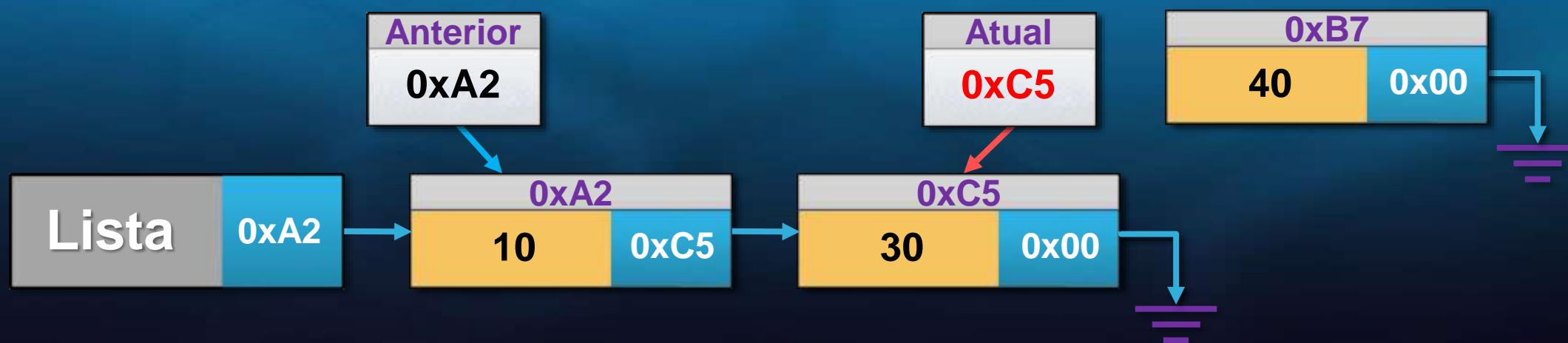


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



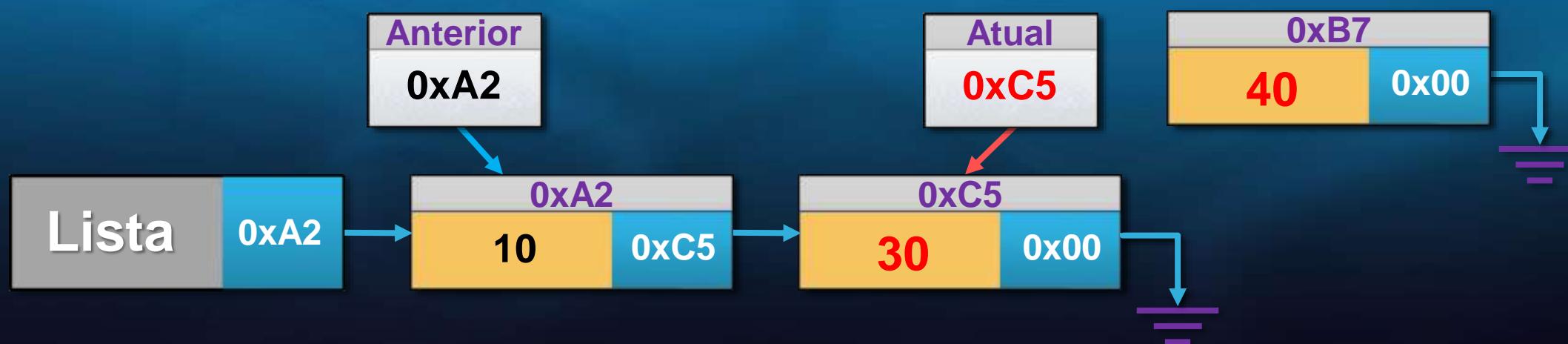
```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

30 < 40

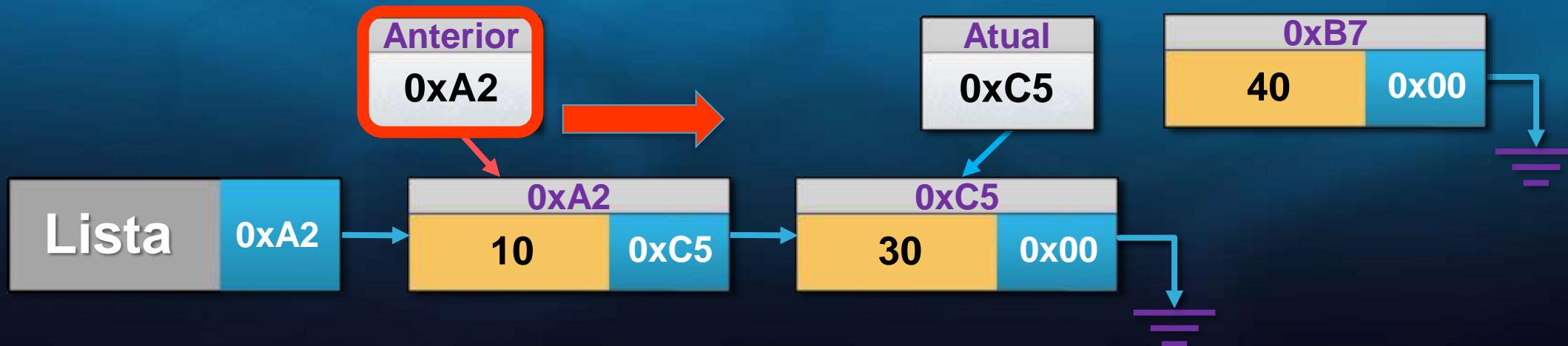


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

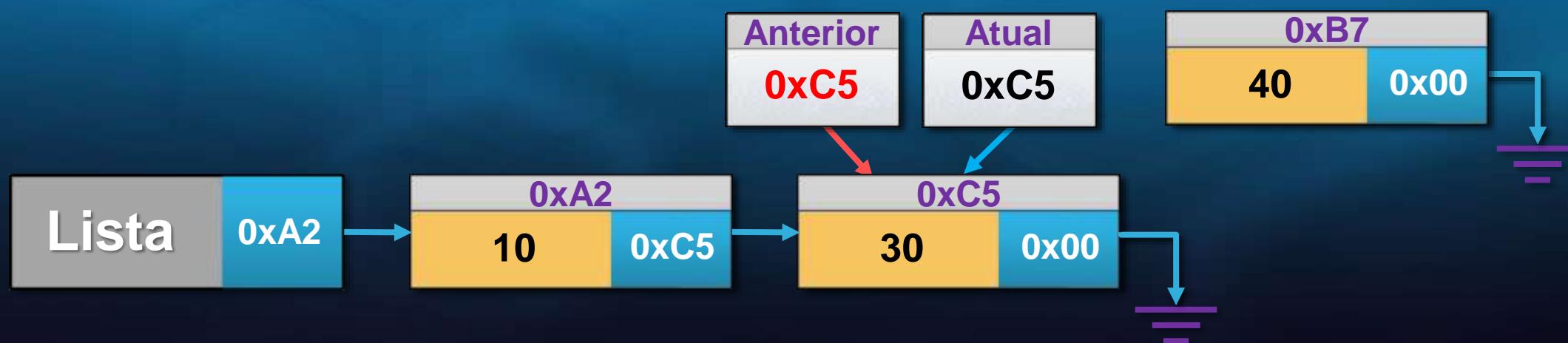


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

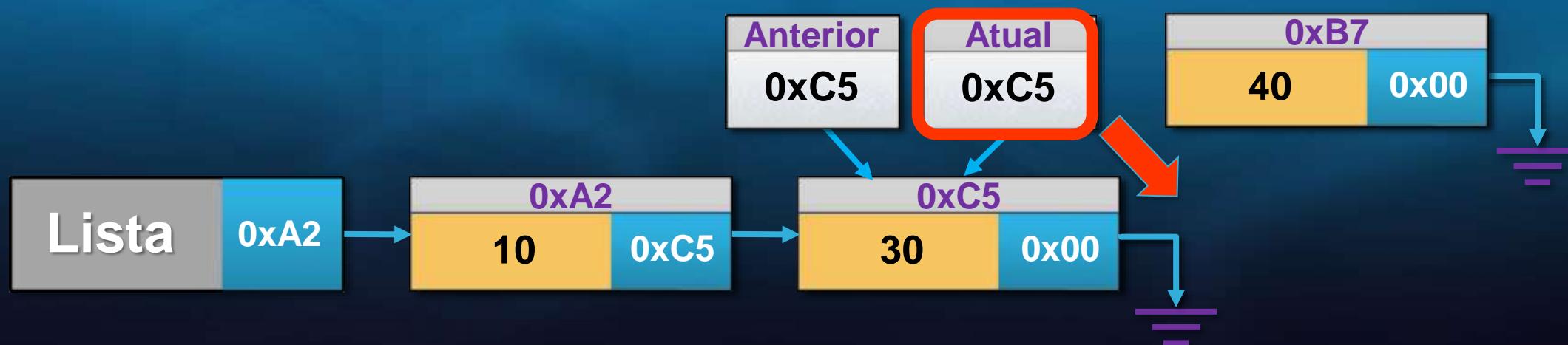


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

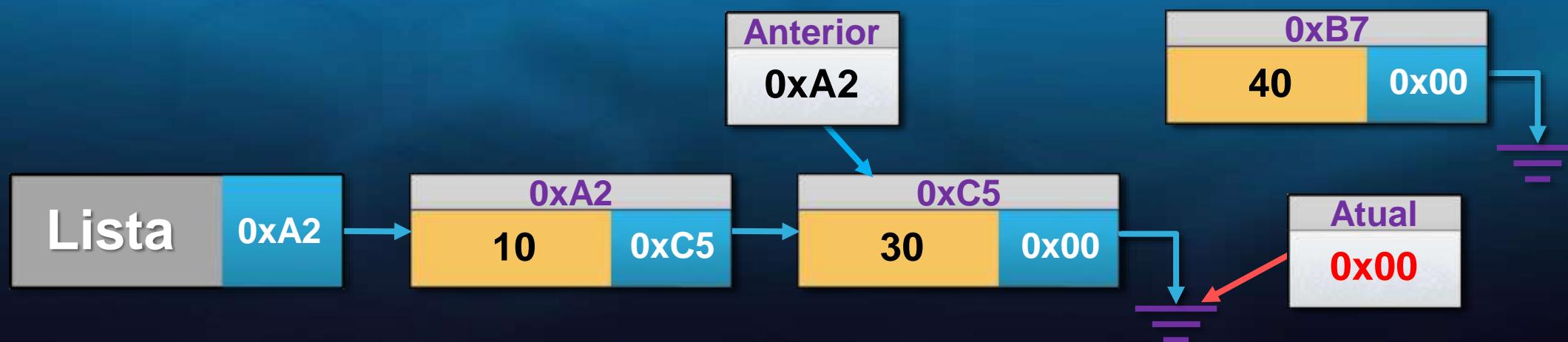


```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;
}

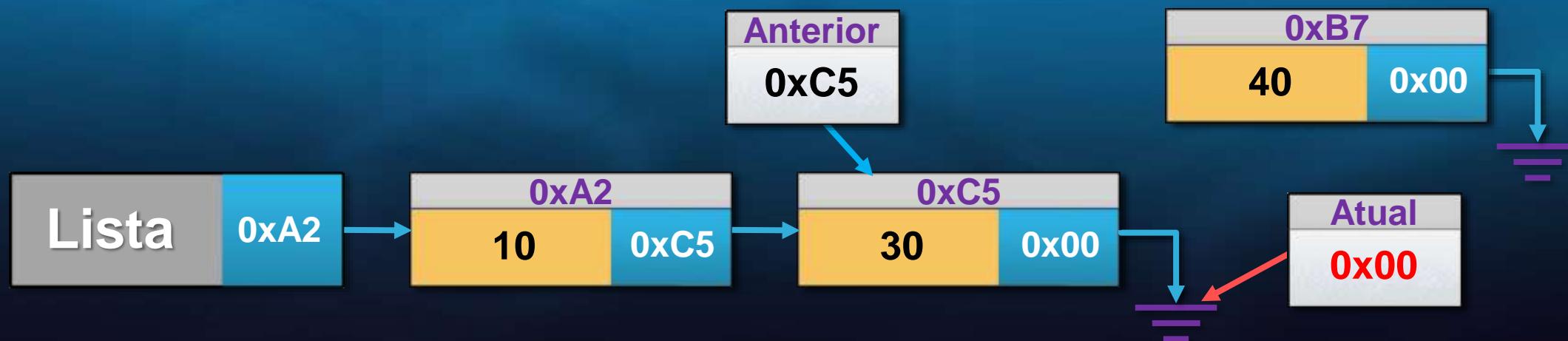
```

FALSO

```

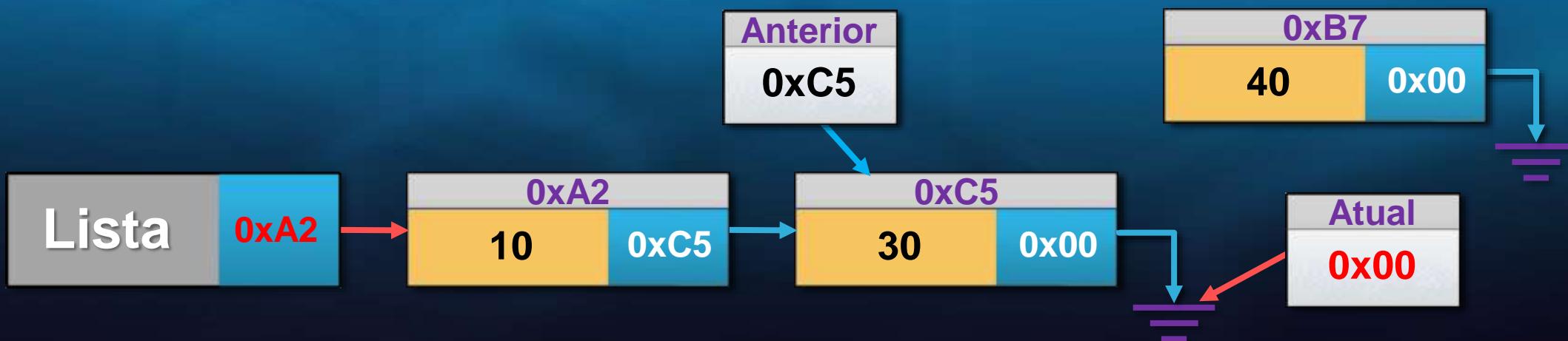
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

```



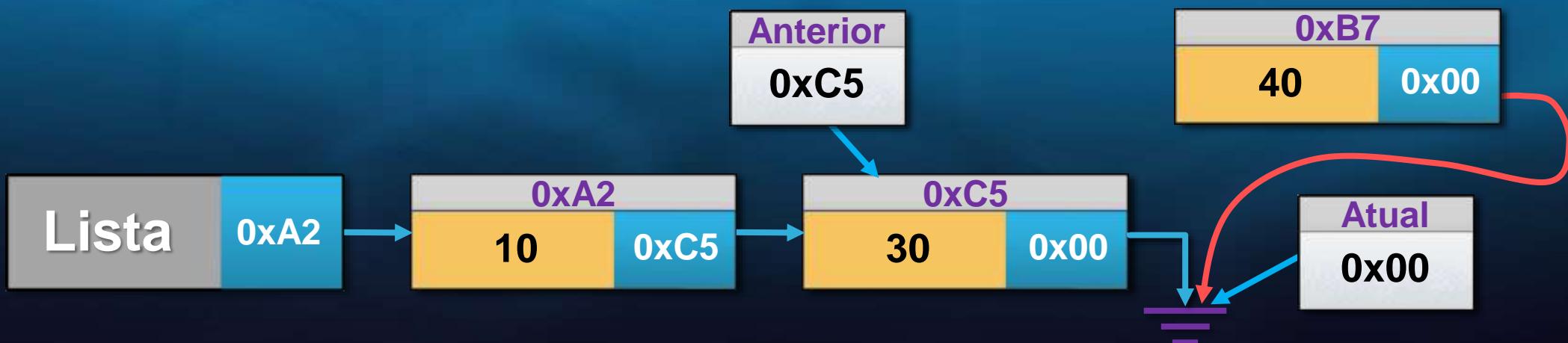
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



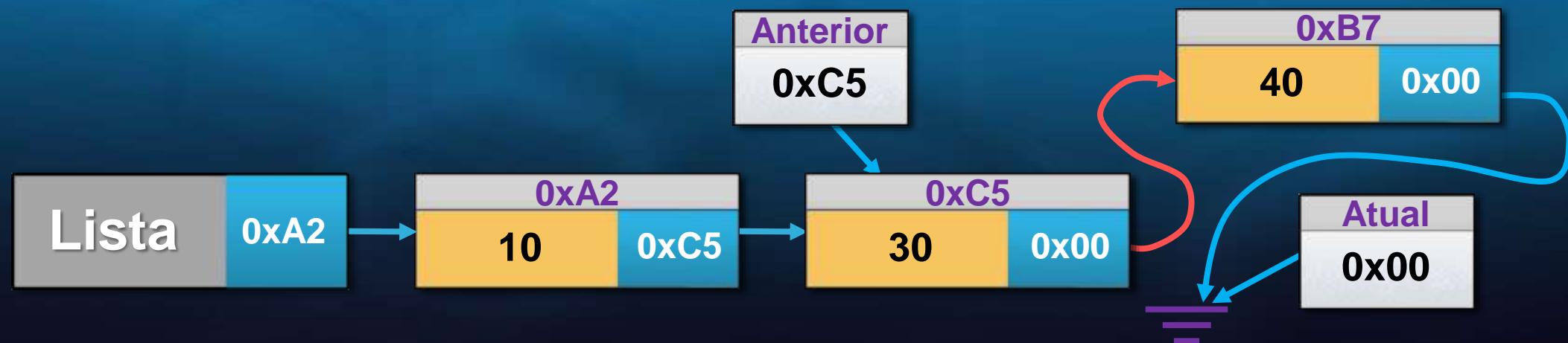
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



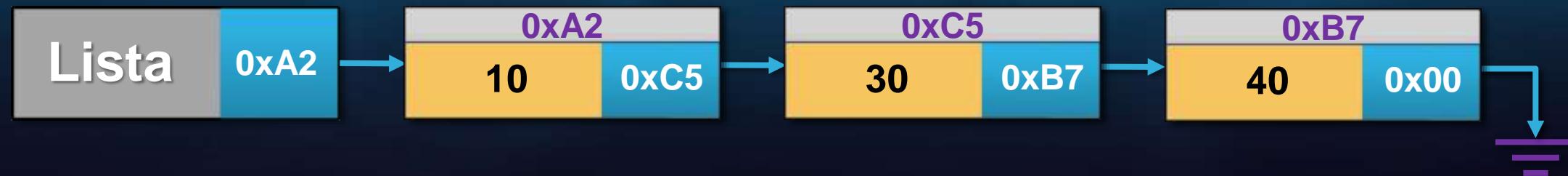
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



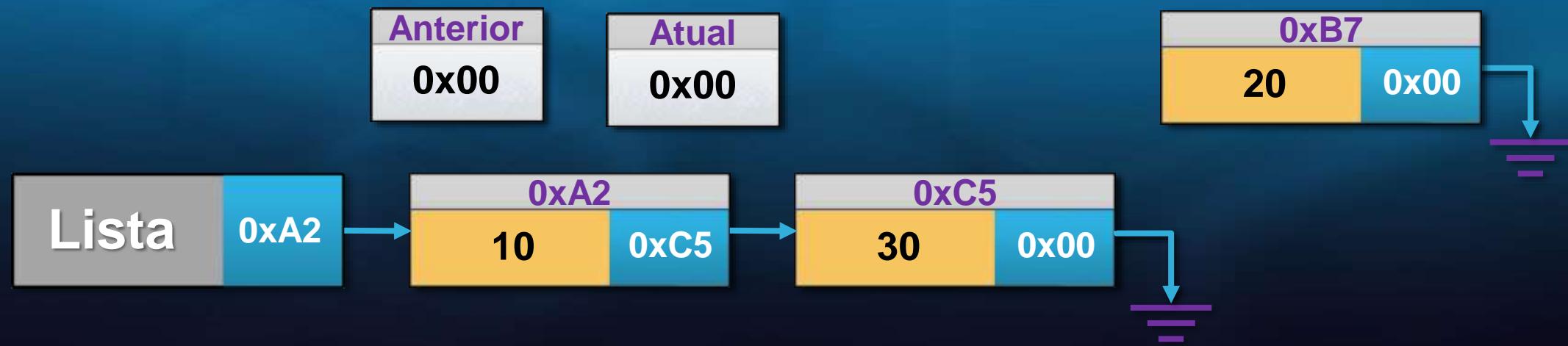
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



INSERIR ORDENADO NA LISTA

SIMULAÇÃO 4

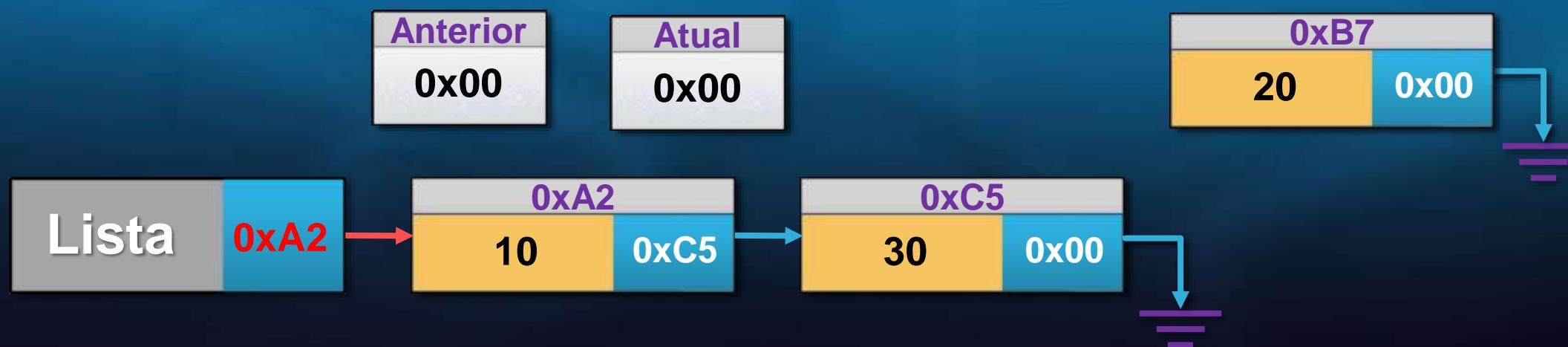


```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

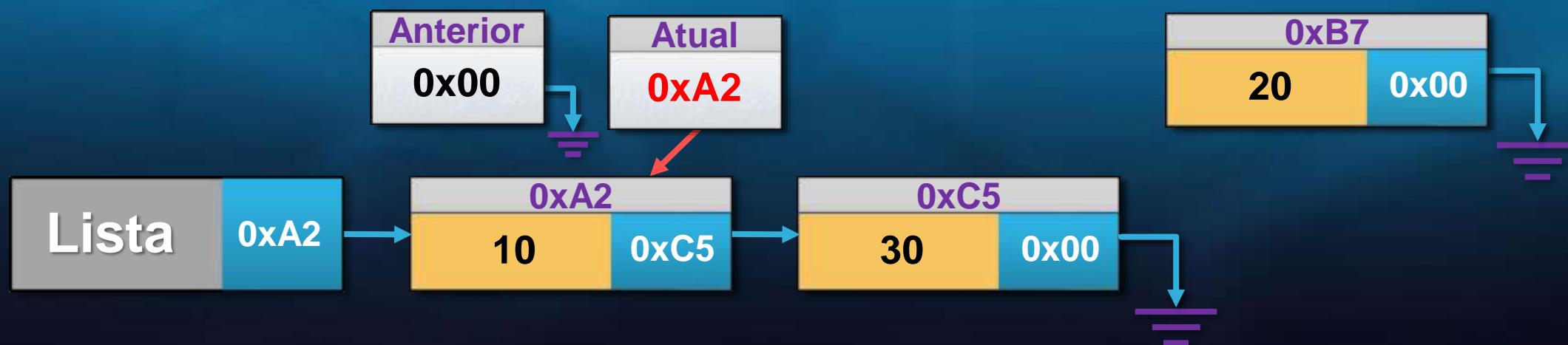


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



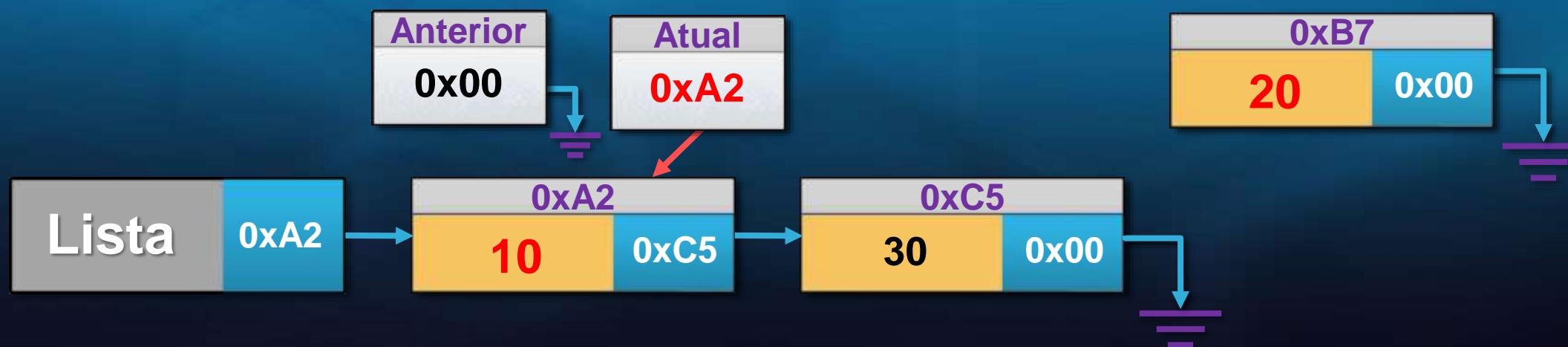
```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

10 < 20

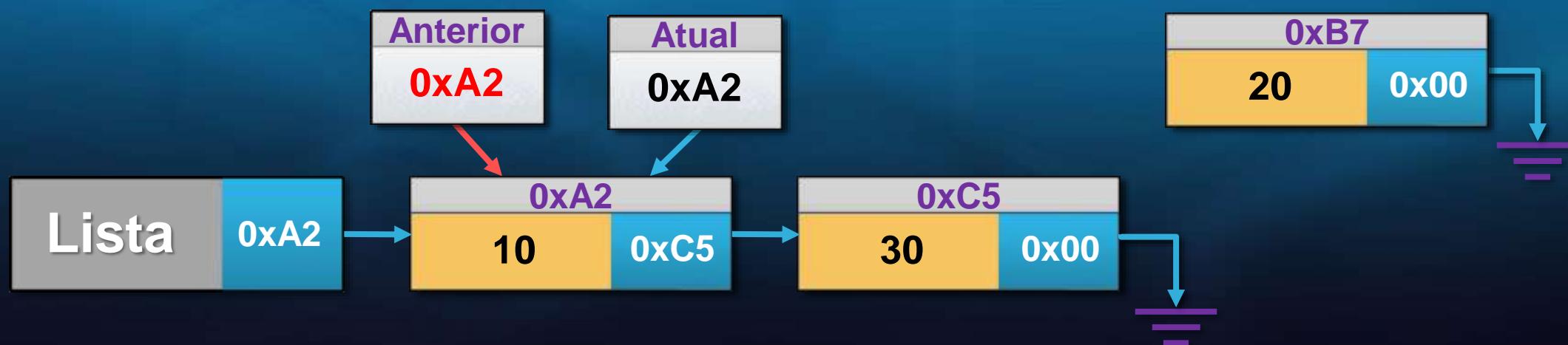


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

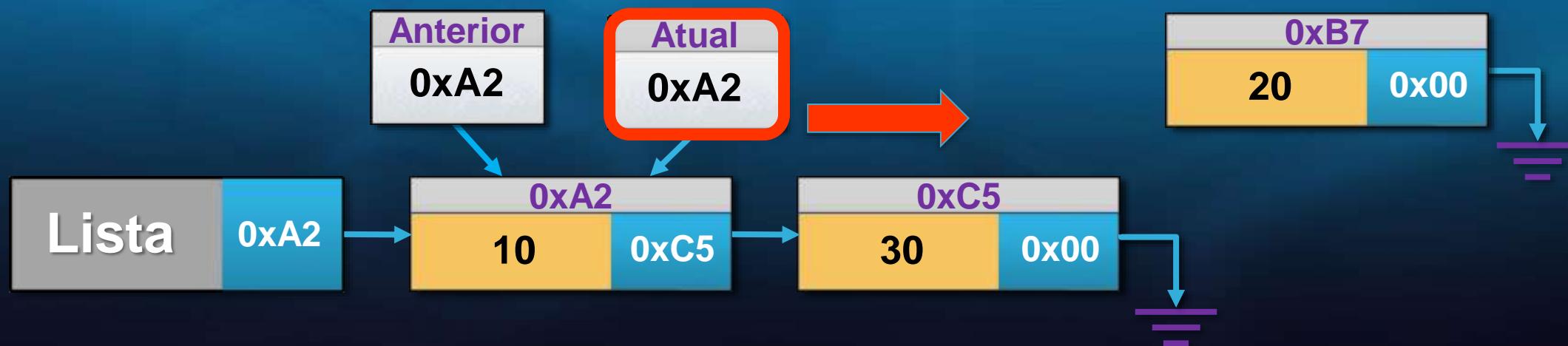


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```

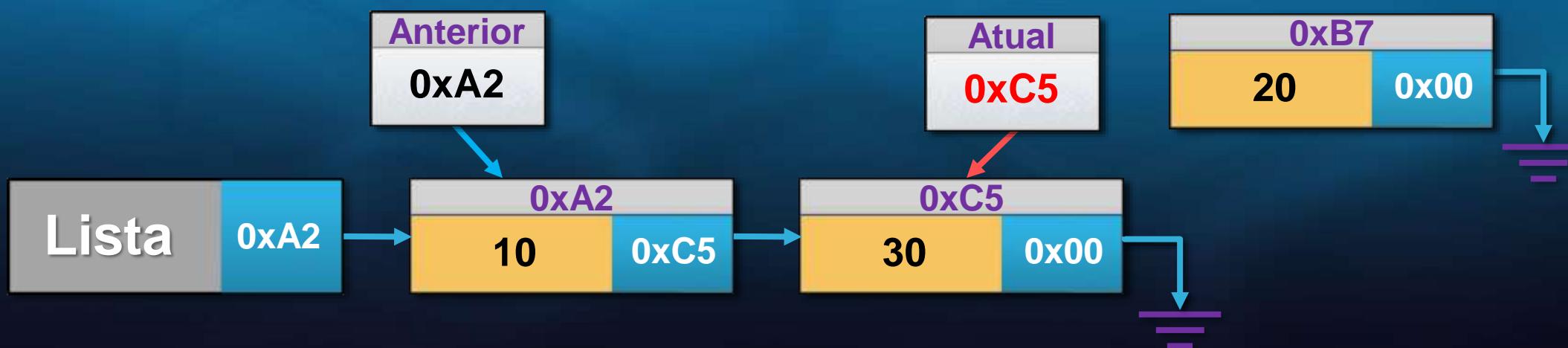


```

// Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

```



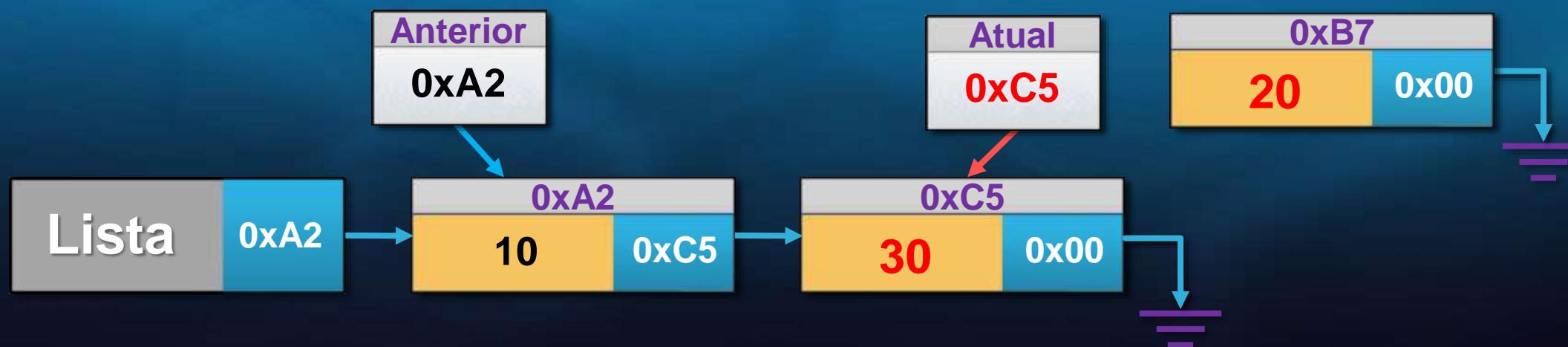
```

//Se a lista estiver vazia
if (ptrLista->inicio == NULL) {
    ptrLista->inicio = ptrNoNovo;
}
else
{
    ptrNoAnterior = NULL;
    ptrNoAtual = ptrLista->inicio;

    // Localiza a posição de inserção
    while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula < matricula) {
        ptrNoAnterior = ptrNoAtual;
        ptrNoAtual = ptrNoAtual->proxNo;
    }
}

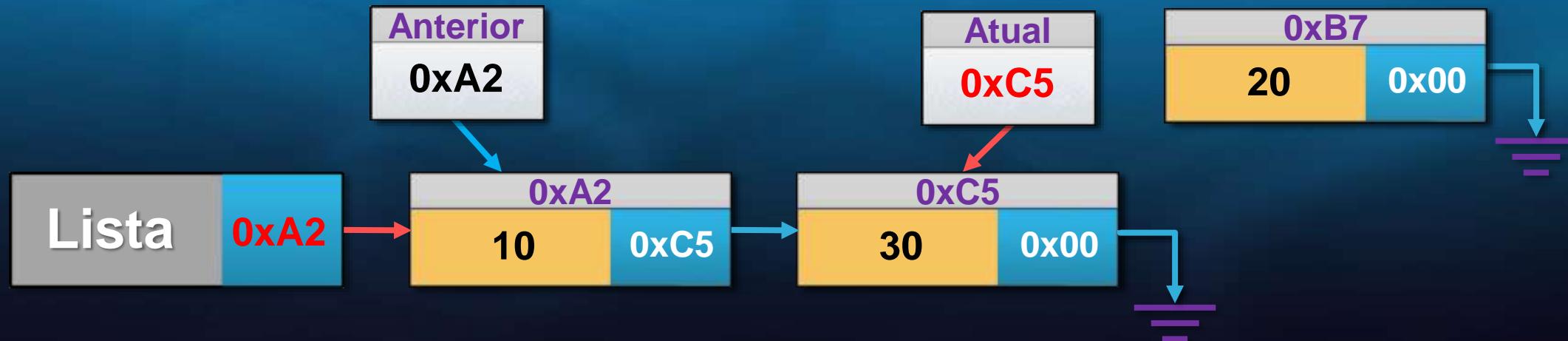
```

30 < 20



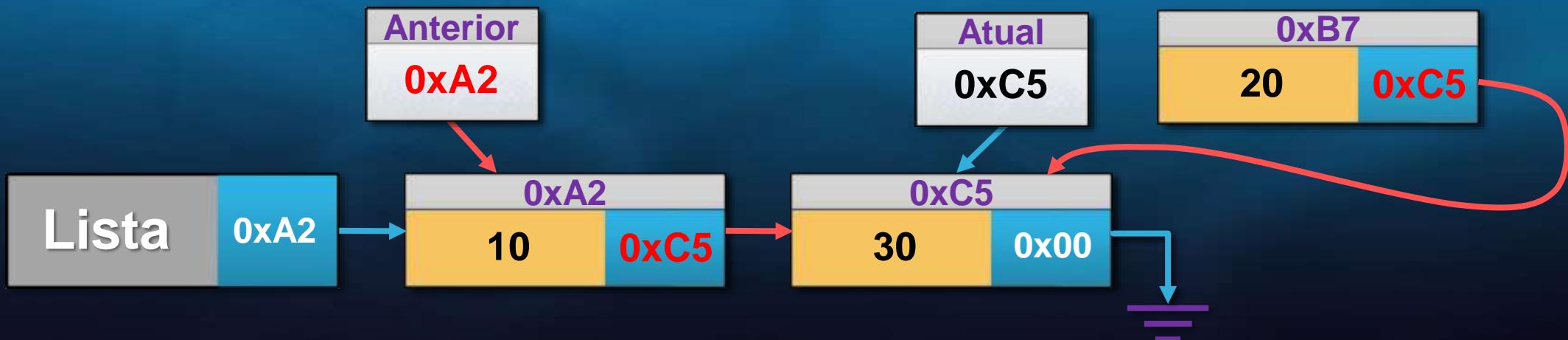
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



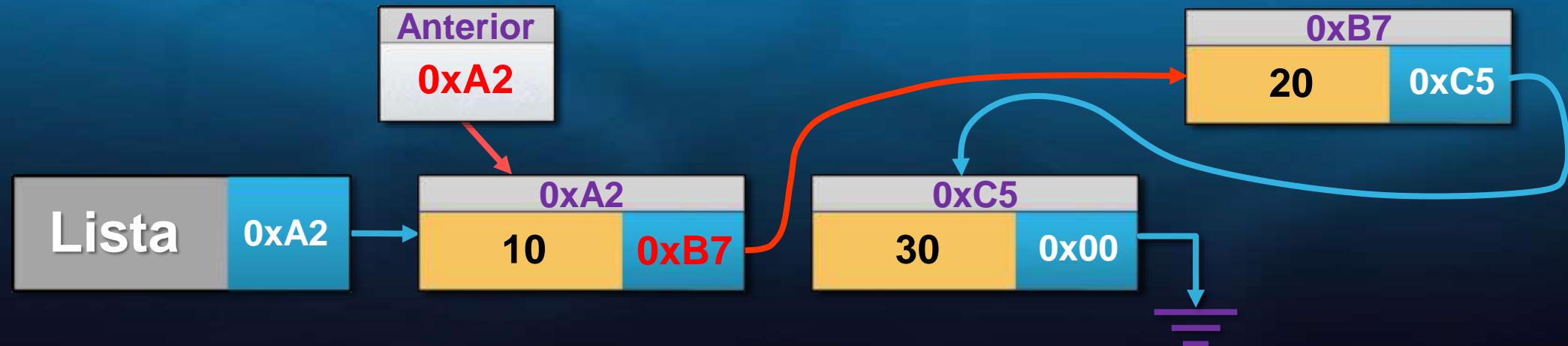
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



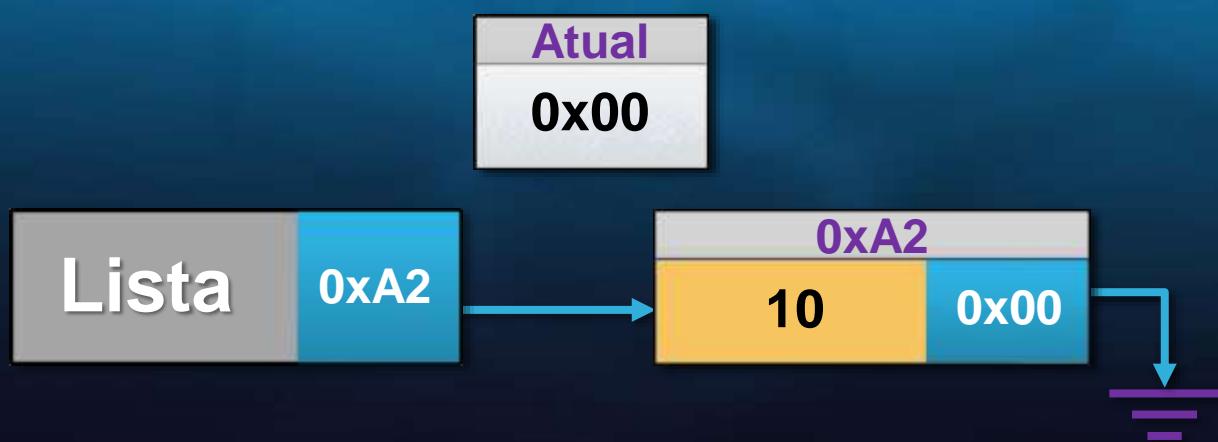
```
// Insere no INÍCIO da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrNoNovo->proxNo = ptrLista->inicio;
    ptrLista->inicio = ptrNoNovo;
}
else { // Insere no MEIO ou no FIM da lista

    ptrNoNovo->proxNo = ptrNoAtual;
    ptrNoAnterior->proxNo = ptrNoNovo;
}
```



EXCLUIR NO INÍCIO DA LISTA

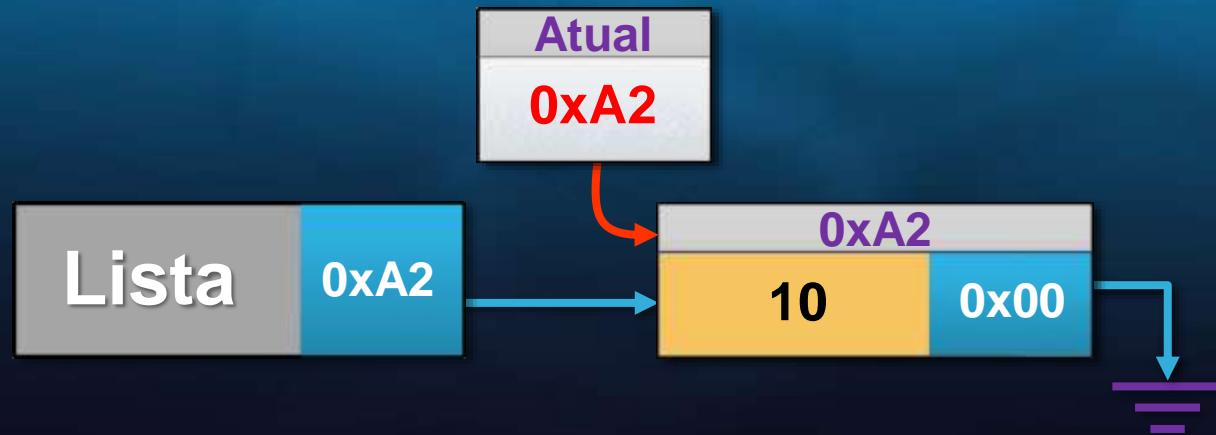
SIMULAÇÃO 1



```
// Ajusta o INÍCIO
ptrNoAtual = ptrLista->inicio;
ptrLista->inicio = ptrNoAtual->proxNo;

// Exclui o primeiro nó
delete ptrNoAtual;

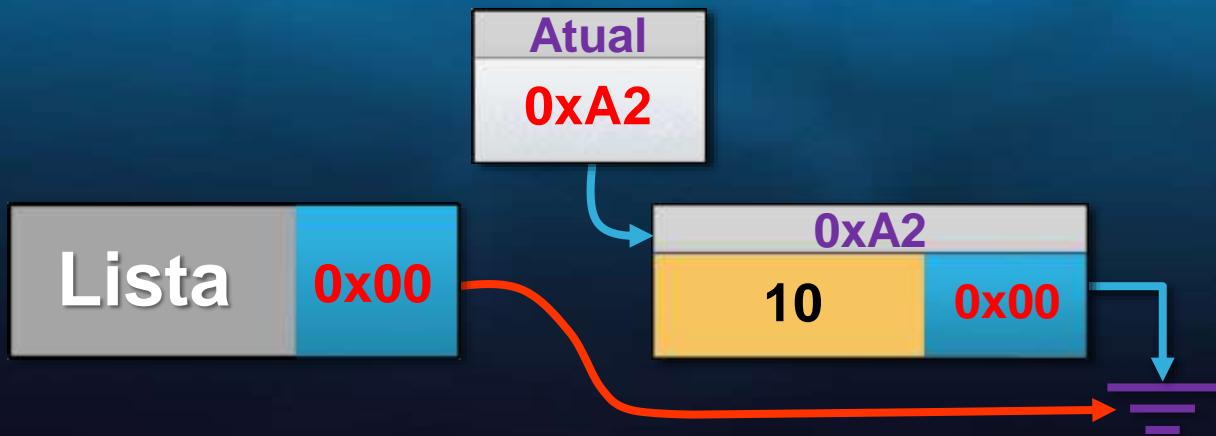
// Decrementa o quantidade de Nós
ptrLista->qtdNo--;
```



```
// Ajusta o INÍCIO  
ptrNoAtual = ptrLista->inicio;  
ptrLista->inicio = ptrNoAtual->proxNo;
```

```
// Exclui o primeiro nó  
delete ptrNoAtual;
```

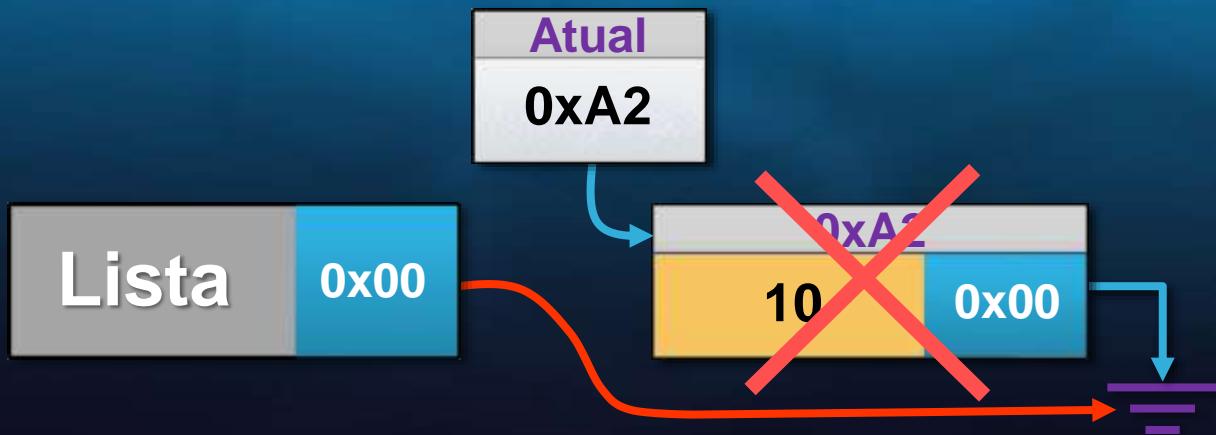
```
// Decrementa o quantidade de Nós  
ptrLista->qtdNo--;
```



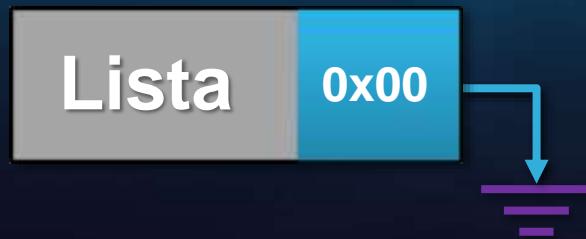
```
// Ajusta o INÍCIO  
ptrNoAtual = ptrLista->inicio;  
ptrLista->inicio = ptrNoAtual->proxNo;
```

```
// Exclui o primeiro nó  
delete ptrNoAtual;
```

```
// Decrementa o quantidade de Nós  
ptrLista->qtdNo--;
```

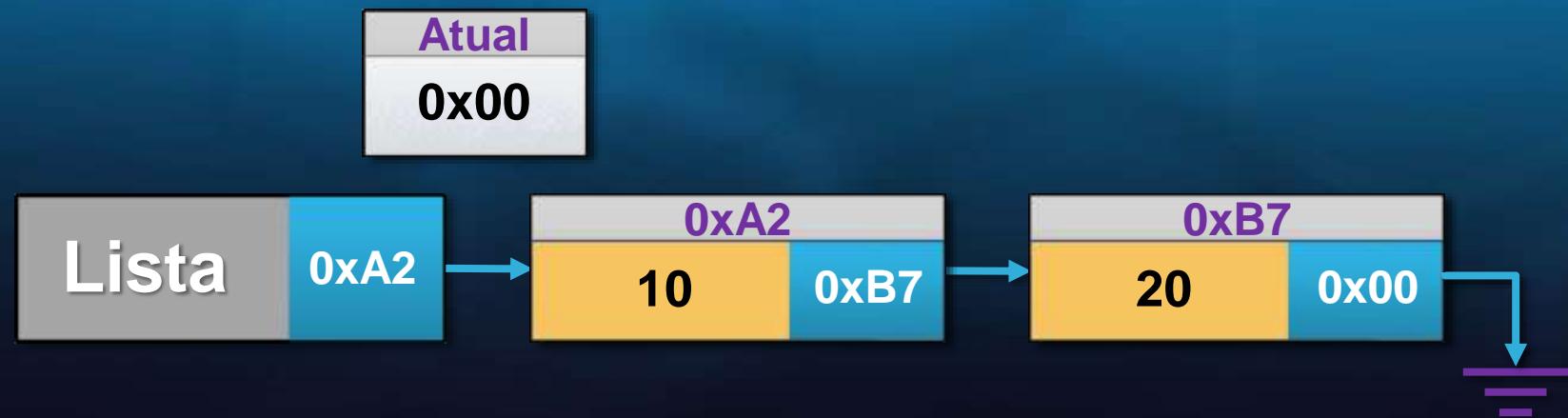


```
// Ajusta o INÍCIO  
ptrNoAtual = ptrLista->inicio;  
ptrLista->inicio = ptrNoAtual->proxNo;  
  
// Exclui o primeiro nó  
delete ptrNoAtual;  
  
// Decrementa o quantidade de Nós  
ptrLista->qtdNo--;
```



EXCLUIR NO INÍCIO DA LISTA

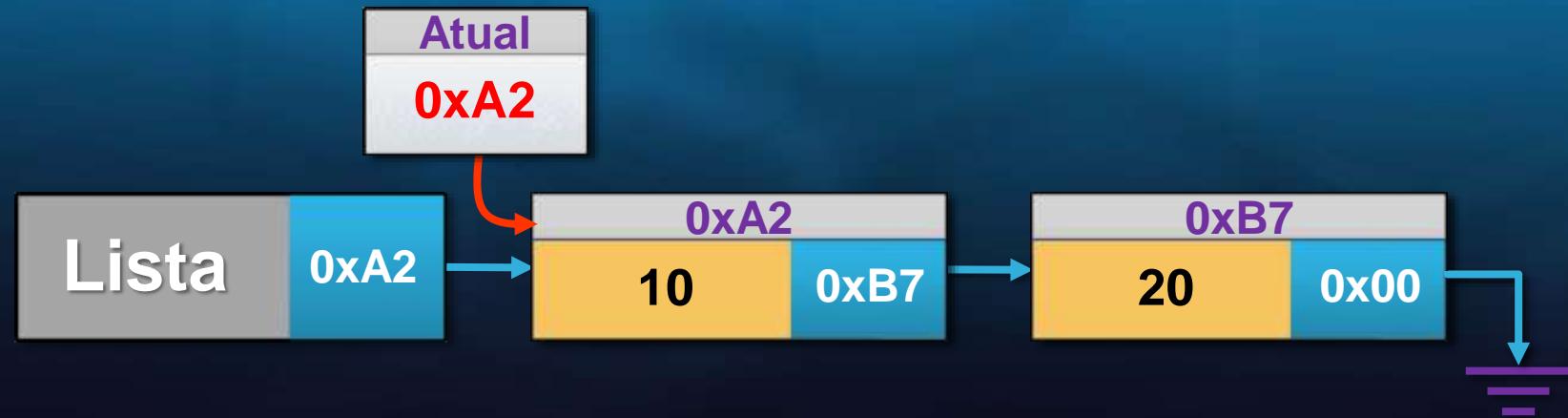
SIMULAÇÃO 2



```
// Ajusta o INÍCIO
ptrNoAtual = ptrLista->inicio;
ptrLista->inicio = ptrNoAtual->proxNo;

// Exclui o primeiro nó
delete ptrNoAtual;

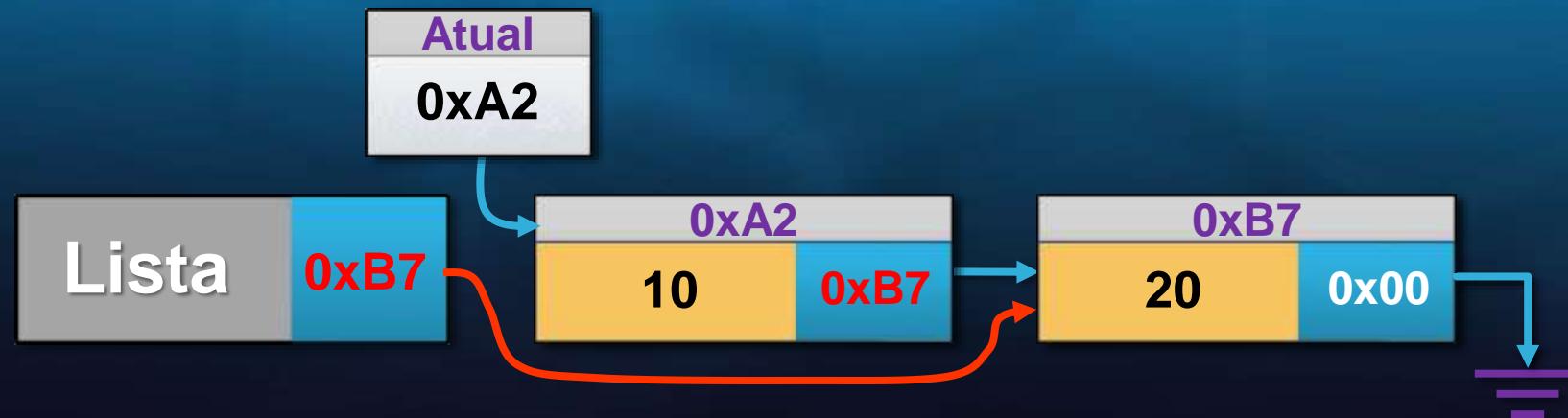
// Decrementa o quantidade de Nós
ptrLista->qtdNo--;
```



```
// Ajusta o INÍCIO  
ptrNoAtual = ptrLista->inicio;  
ptrLista->inicio = ptrNoAtual->proxNo;
```

```
// Exclui o primeiro nó  
delete ptrNoAtual;
```

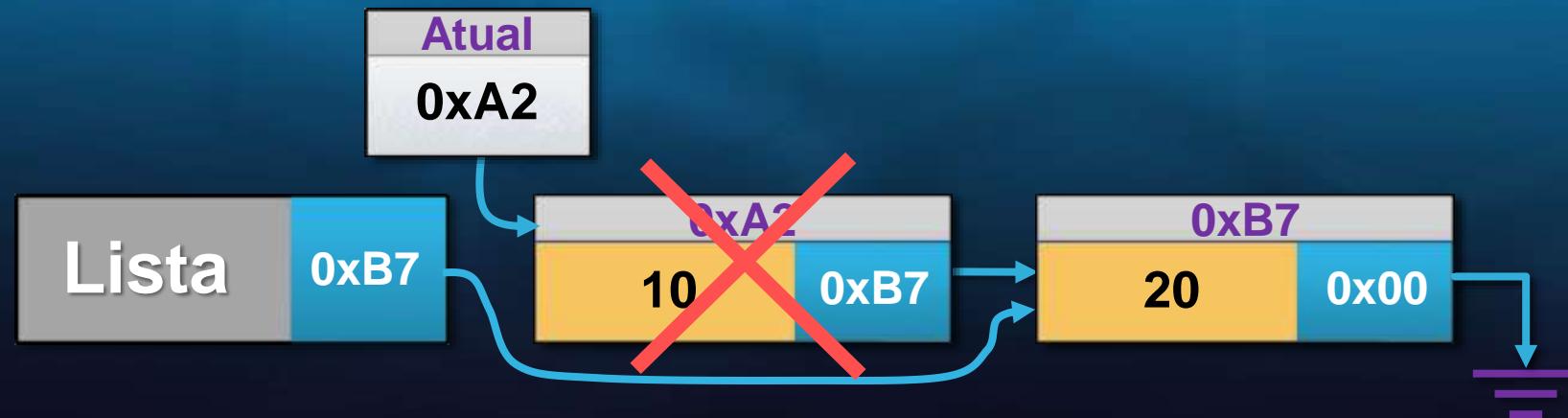
```
// Decrementa o quantidade de Nós  
ptrLista->qtdNo--;
```



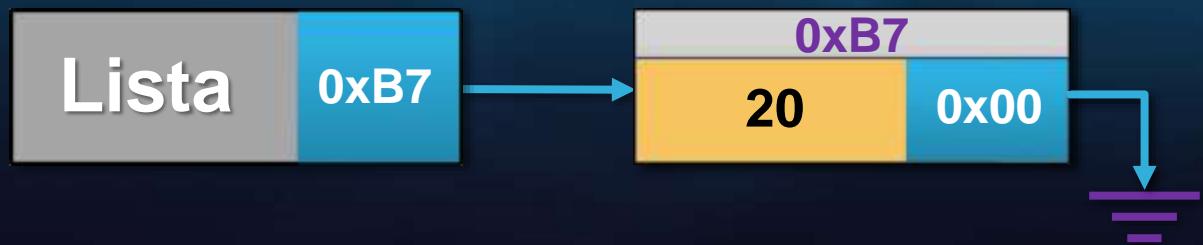
```
// Ajusta o INÍCIO  
ptrNoAtual = ptrLista->inicio;  
ptrLista->inicio = ptrNoAtual->proxNo;
```

```
// Exclui o primeiro nó  
delete ptrNoAtual;
```

```
// Decrementa o quantidade de Nós  
ptrLista->qtdNo--;
```

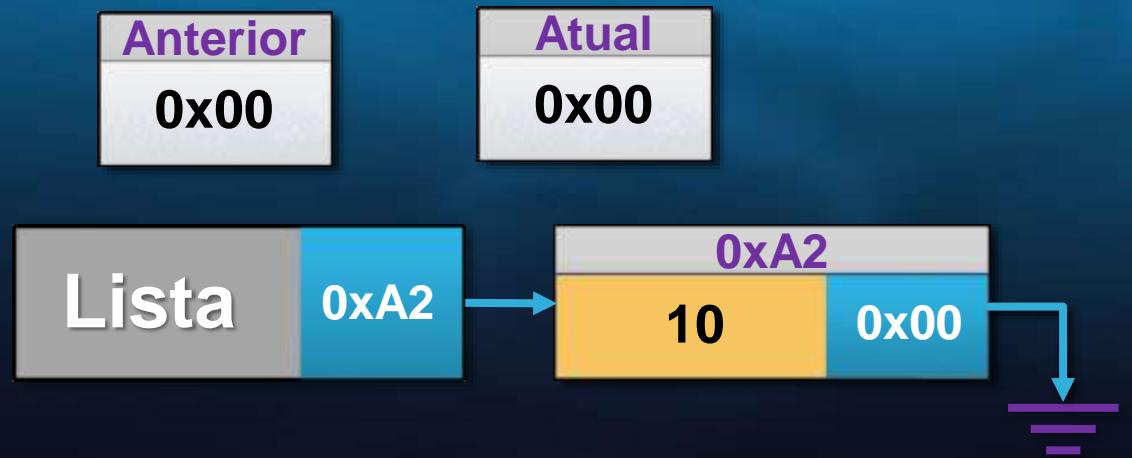


```
// Ajusta o INÍCIO  
ptrNoAtual = ptrLista->inicio;  
ptrLista->inicio = ptrNoAtual->proxNo;  
  
// Exclui o primeiro nó  
delete ptrNoAtual;  
  
// Decrementa o quantidade de Nós  
ptrLista->qtdNo--;
```

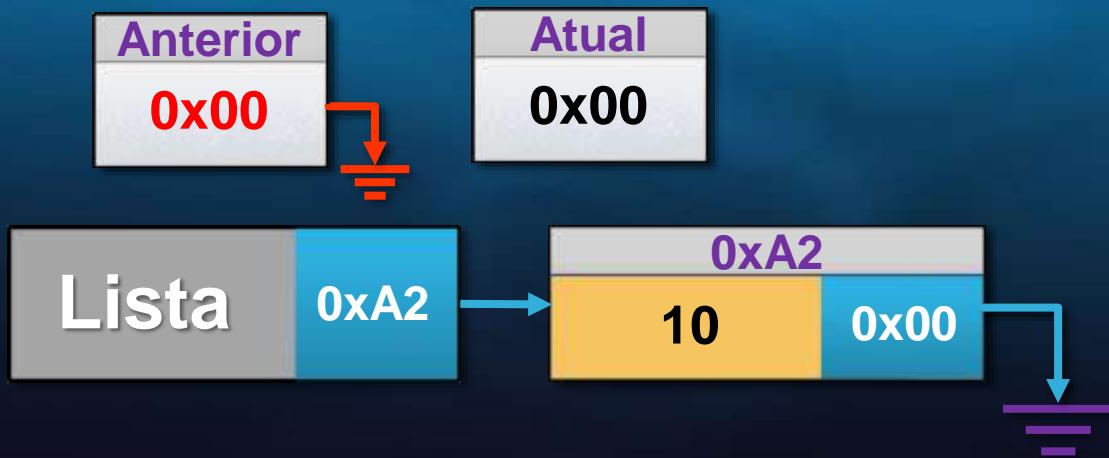


EXCLUIR NO FINAL DA LISTA

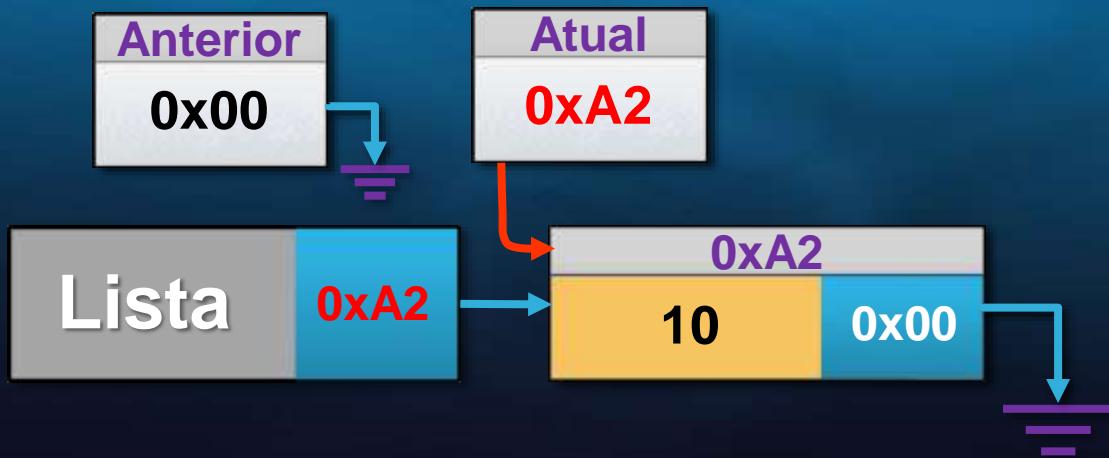
SIMULAÇÃO 1



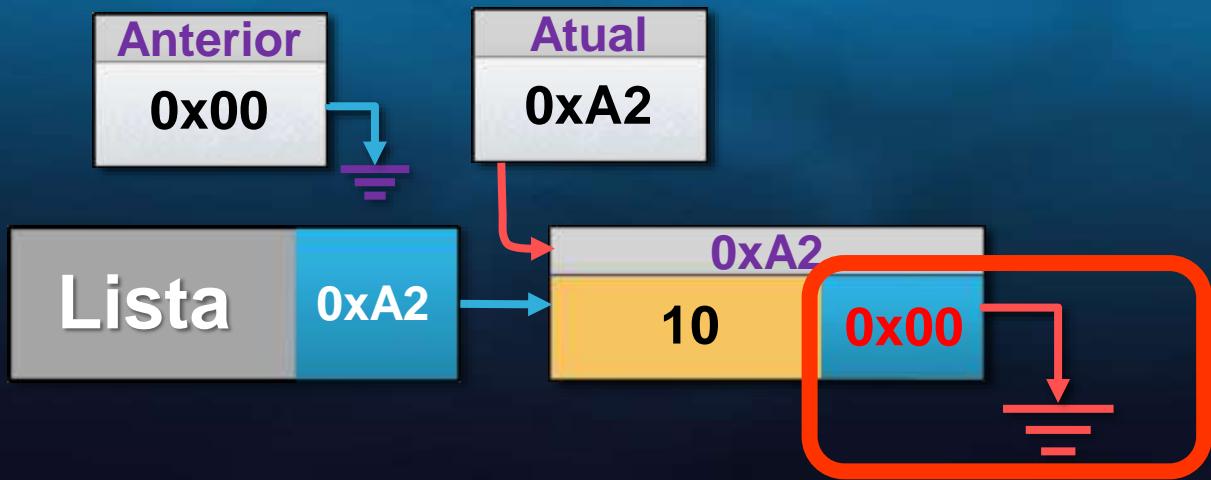
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```

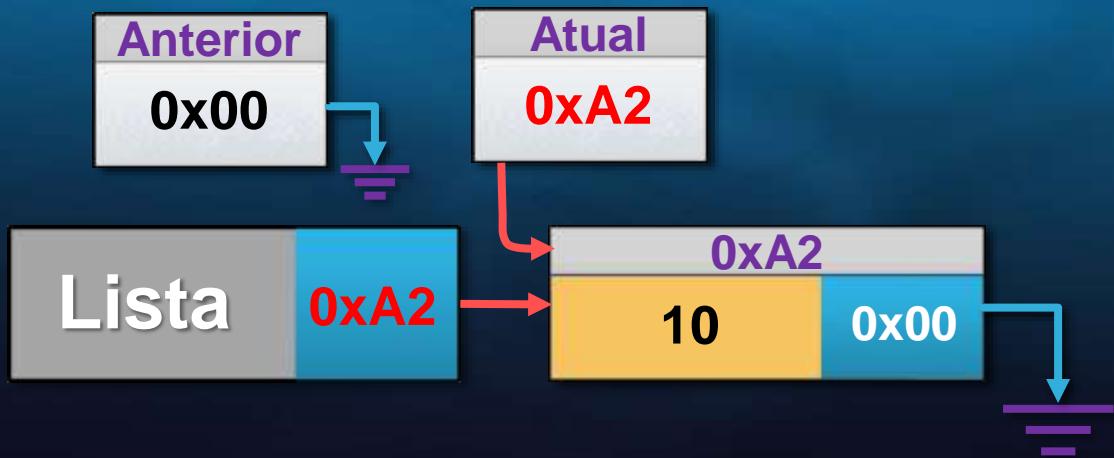


```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



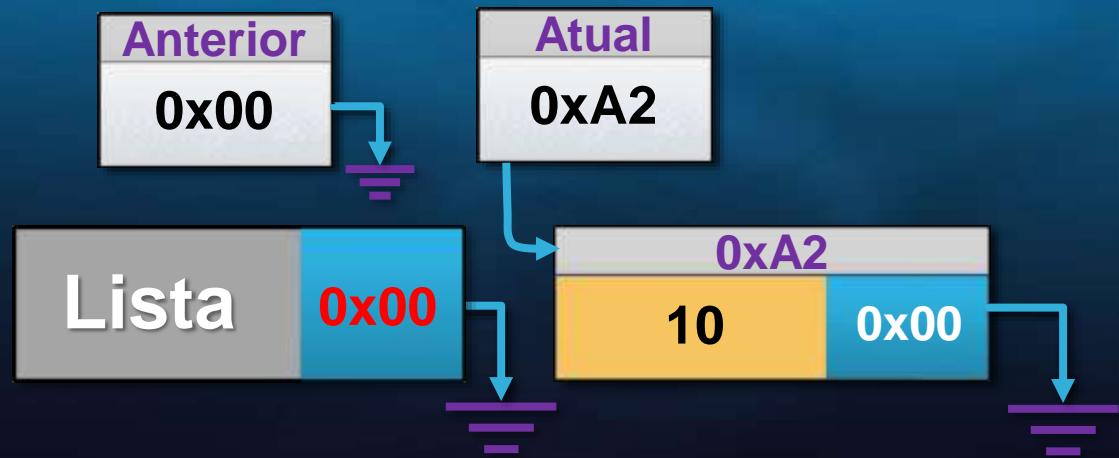
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) [
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```



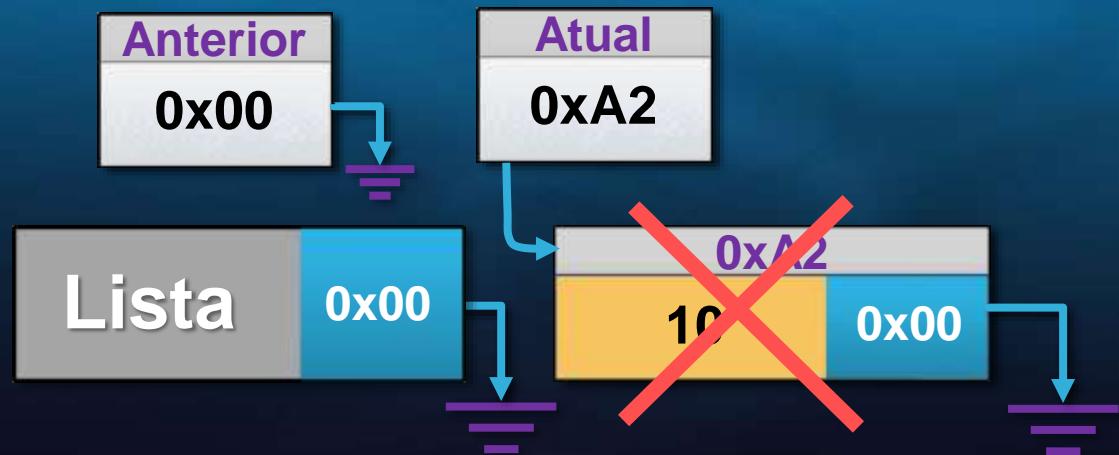
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```

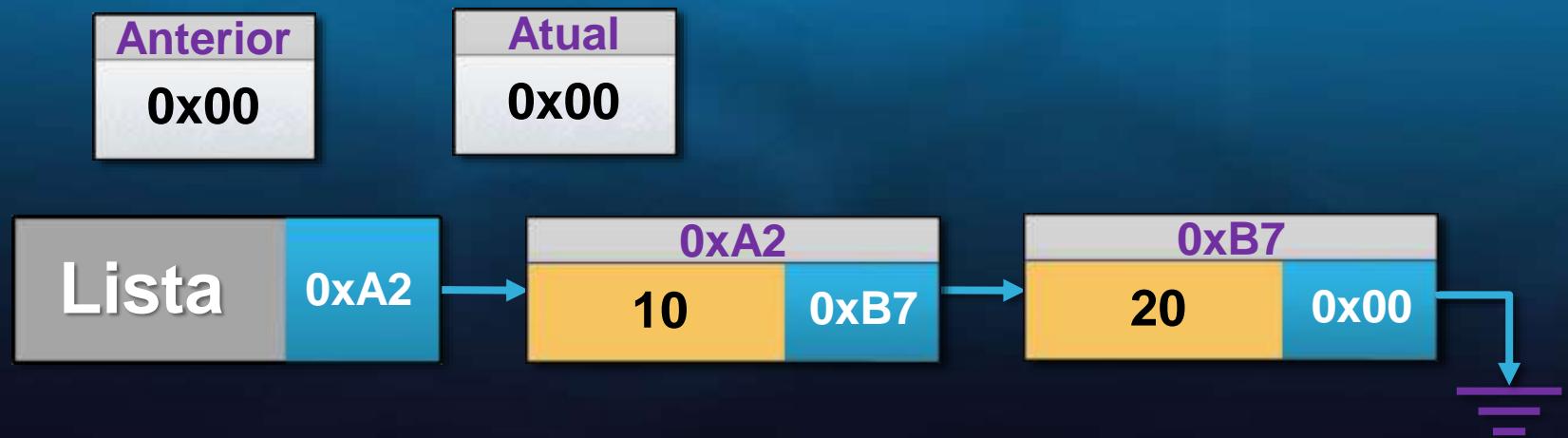
Lista

0x00



EXCLUIR NO FINAL DA LISTA

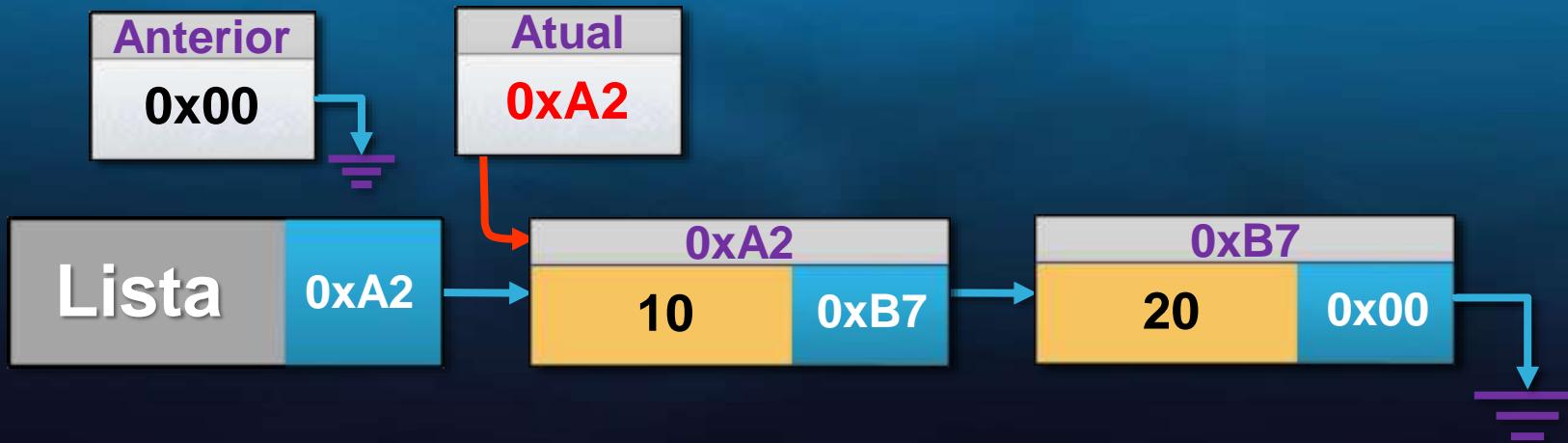
SIMULAÇÃO 2



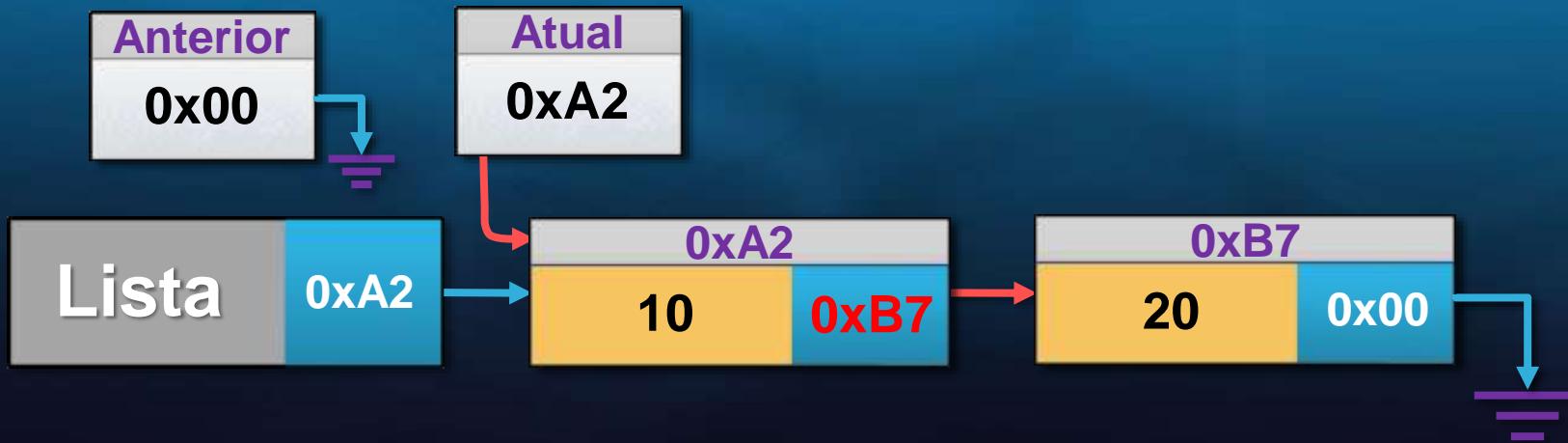
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



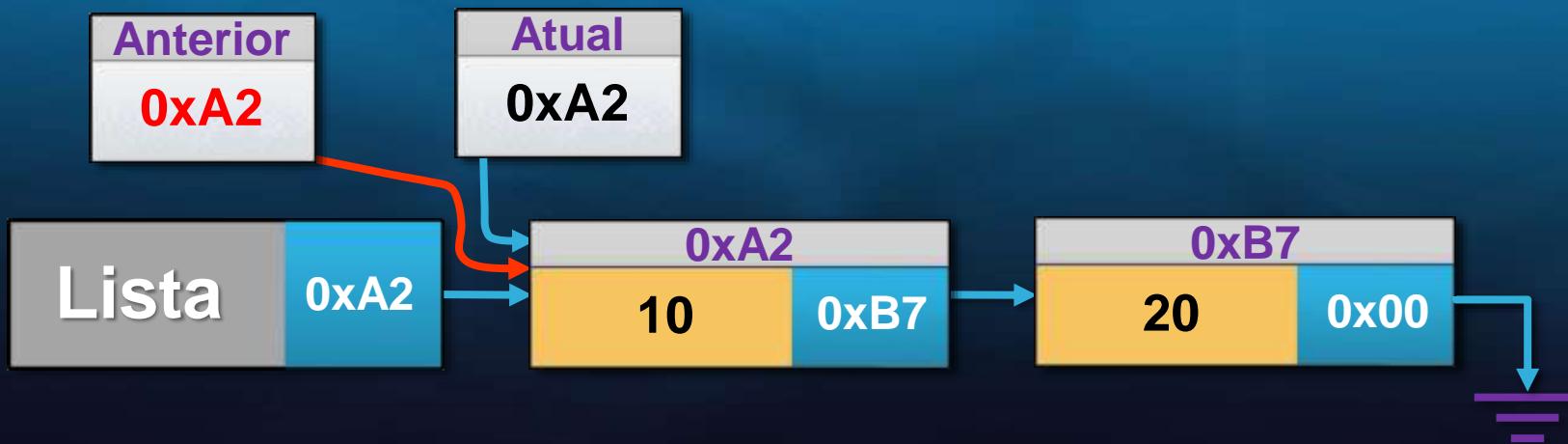
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



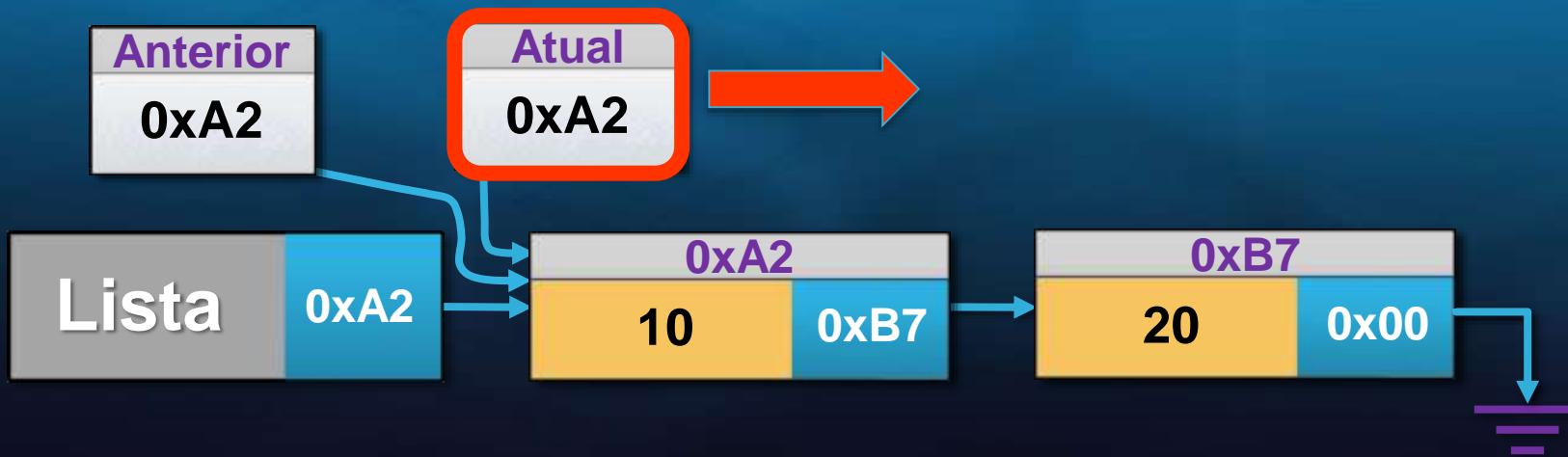
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



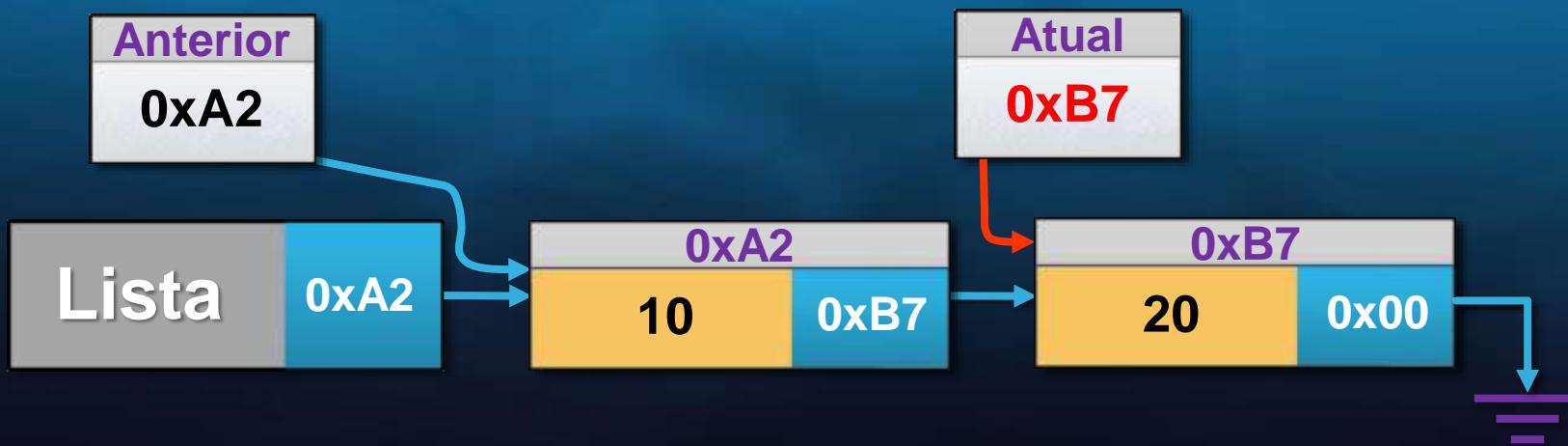
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



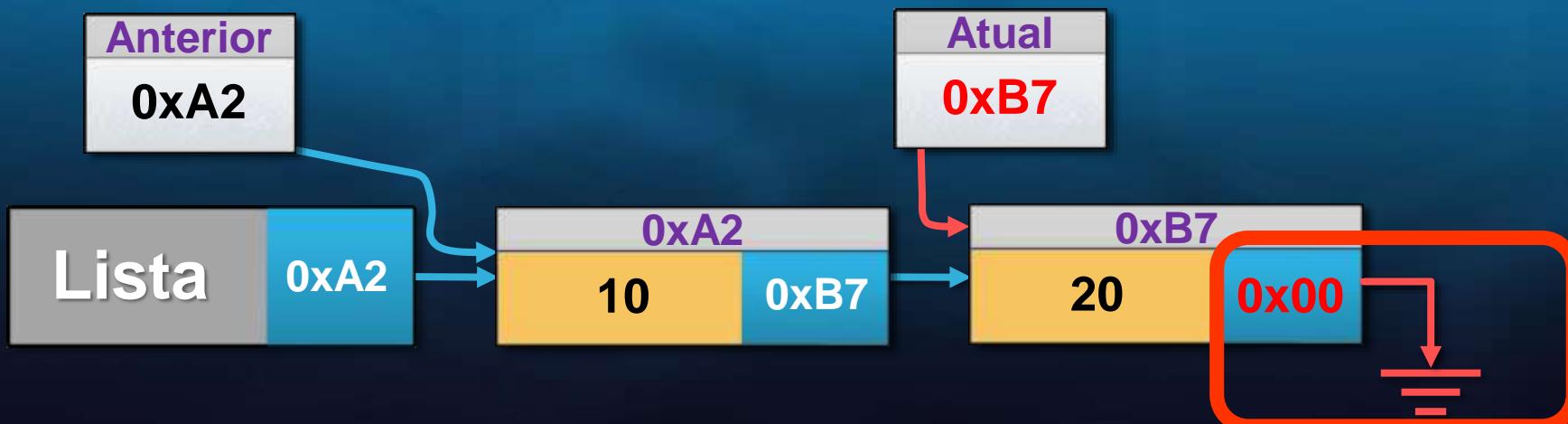
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```

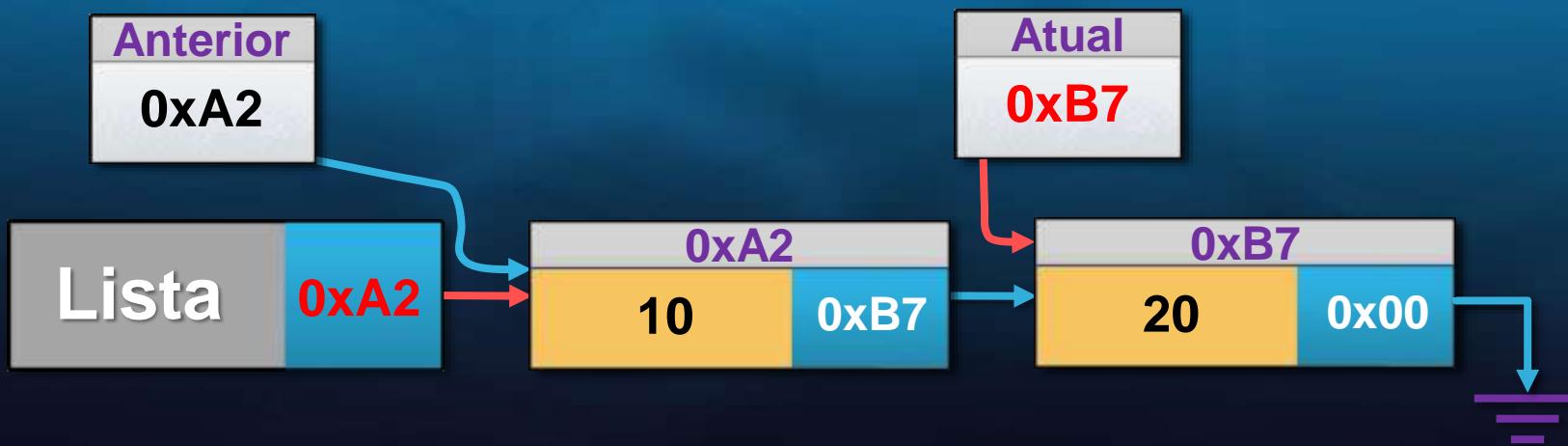


```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localiza o nó final da lista  
while (ptrNoAtual->proxNo != NULL) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```



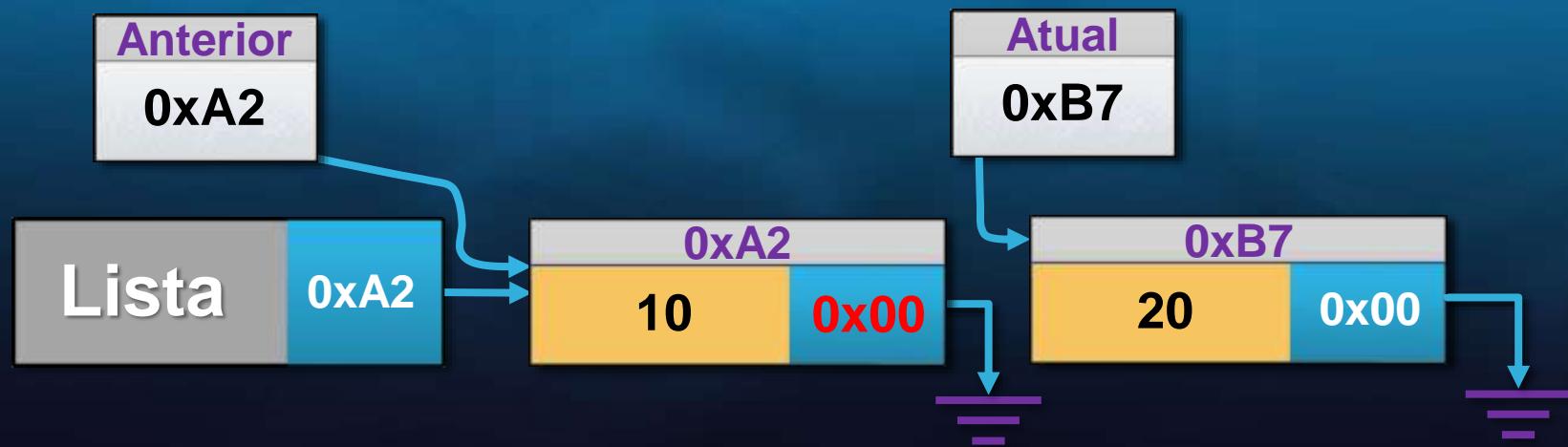
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) [
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```



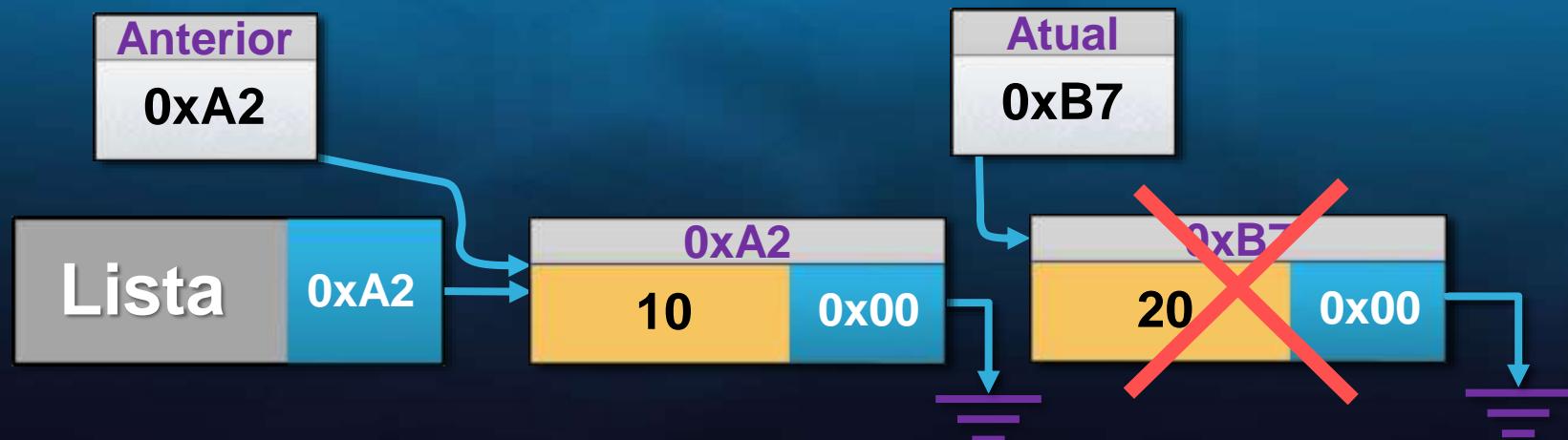
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```



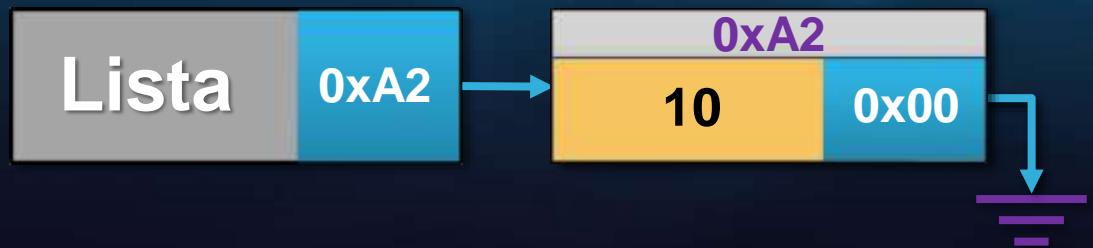
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

// Exclui o primeiro nó
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = NULL;
}
else {
    ptrNoAnterior->proxNo = NULL;
}

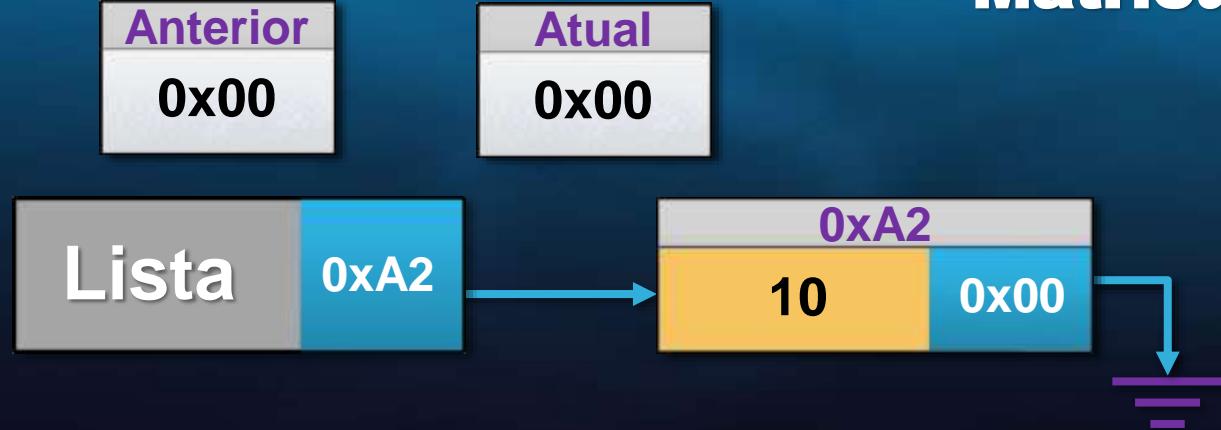
// Exclui o primeiro nó
delete ptrNoAtual;
```



EXCLUIR ORDENADO NA LISTA

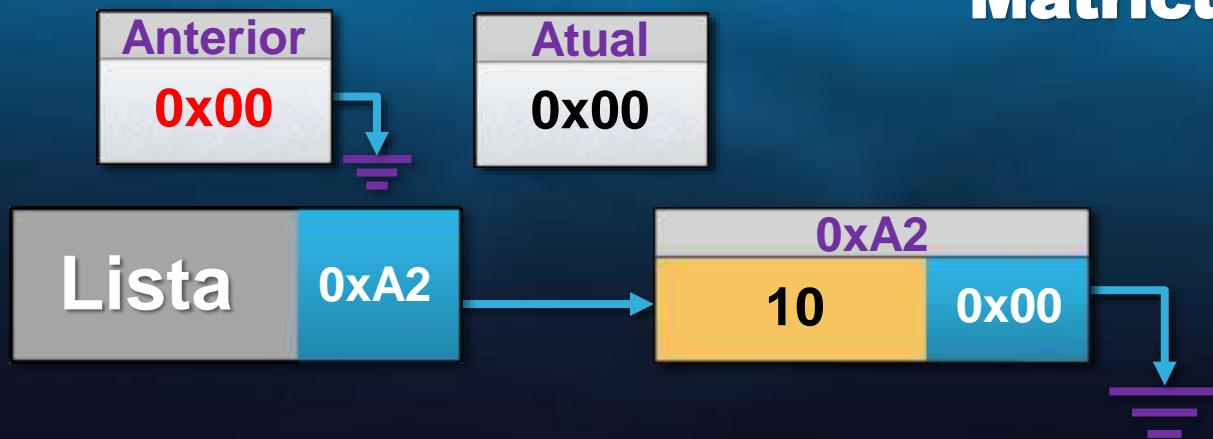
SIMULAÇÃO 1

Matrícula = 10



```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localizao nó que será excluído  
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}  
  
if (ptrNoAtual == NULL) {  
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;  
    return false;  
}
```

Matrícula = 10



```

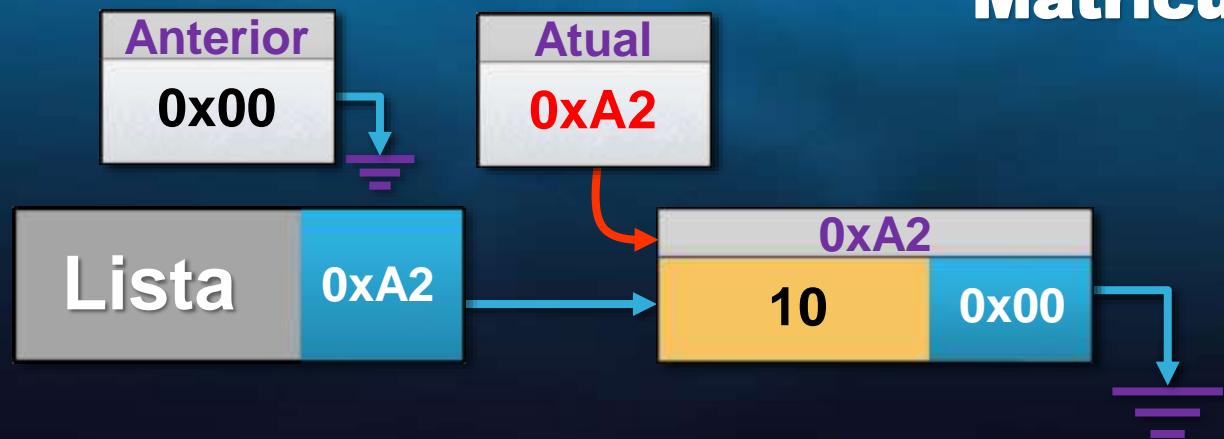
ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```

Matrícula = 10



```

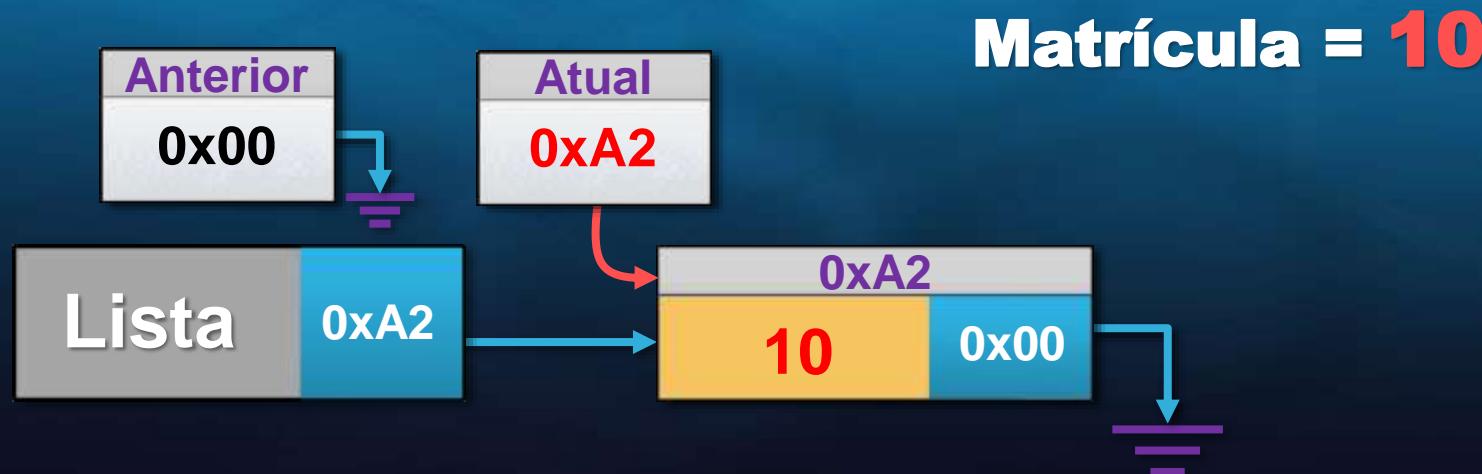
ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localiza o nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```

$$10 \neq 10$$



```

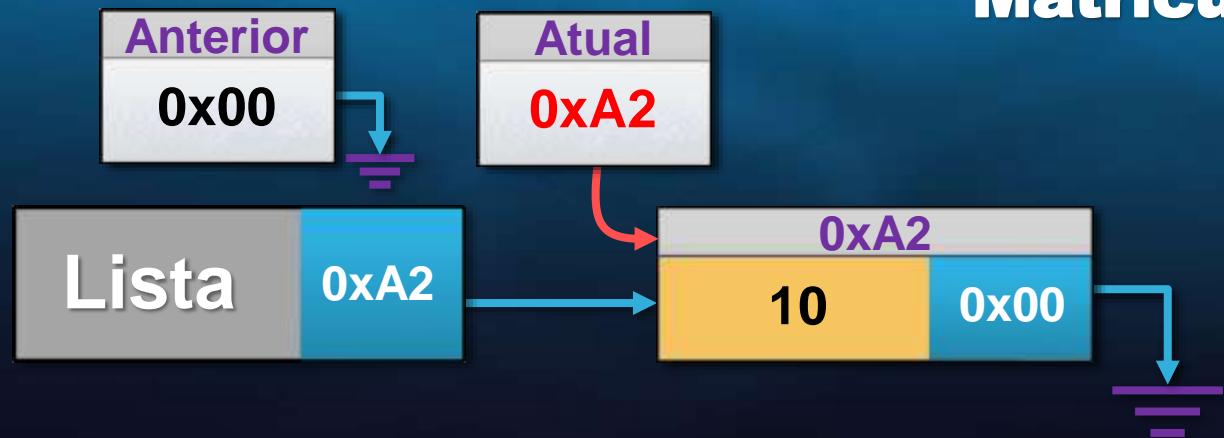
ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```

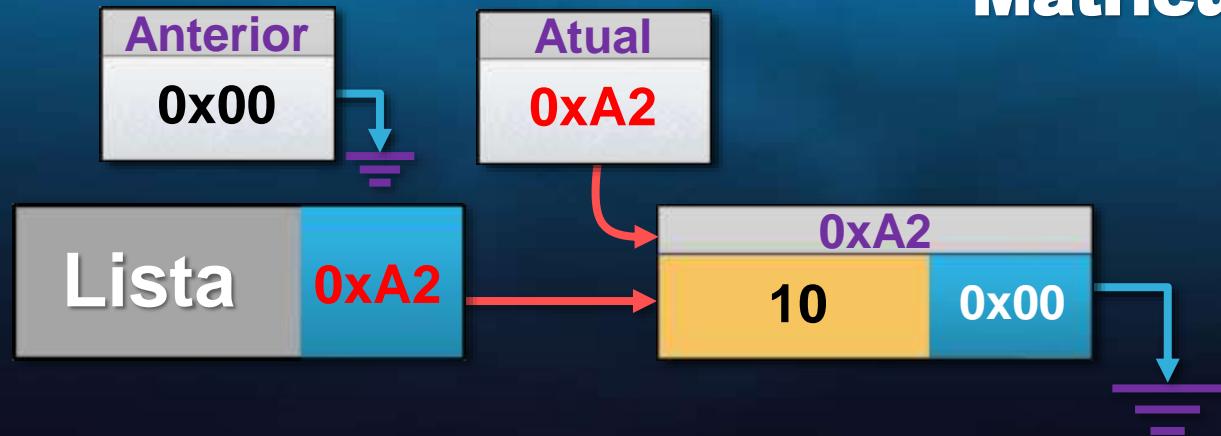
Matrícula = 10



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```

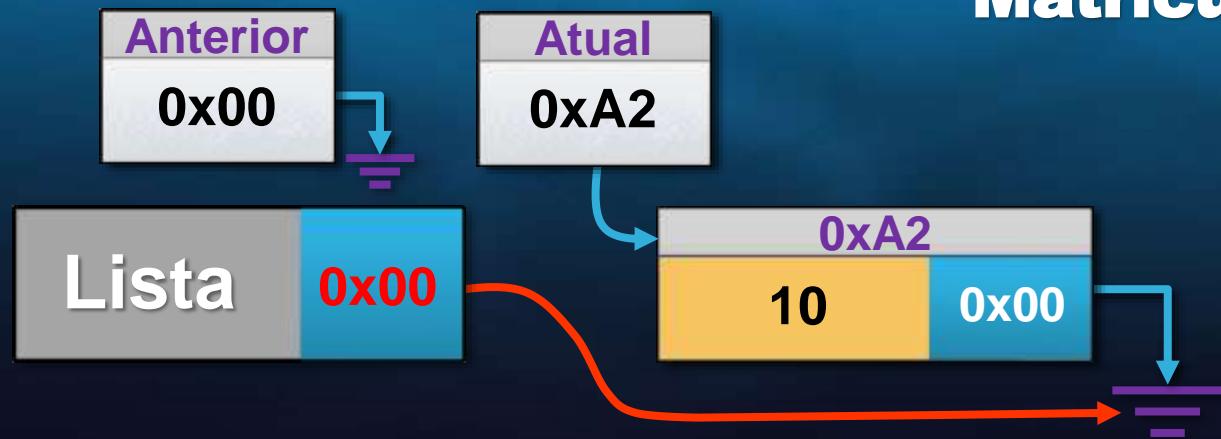
Matrícula = 10



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

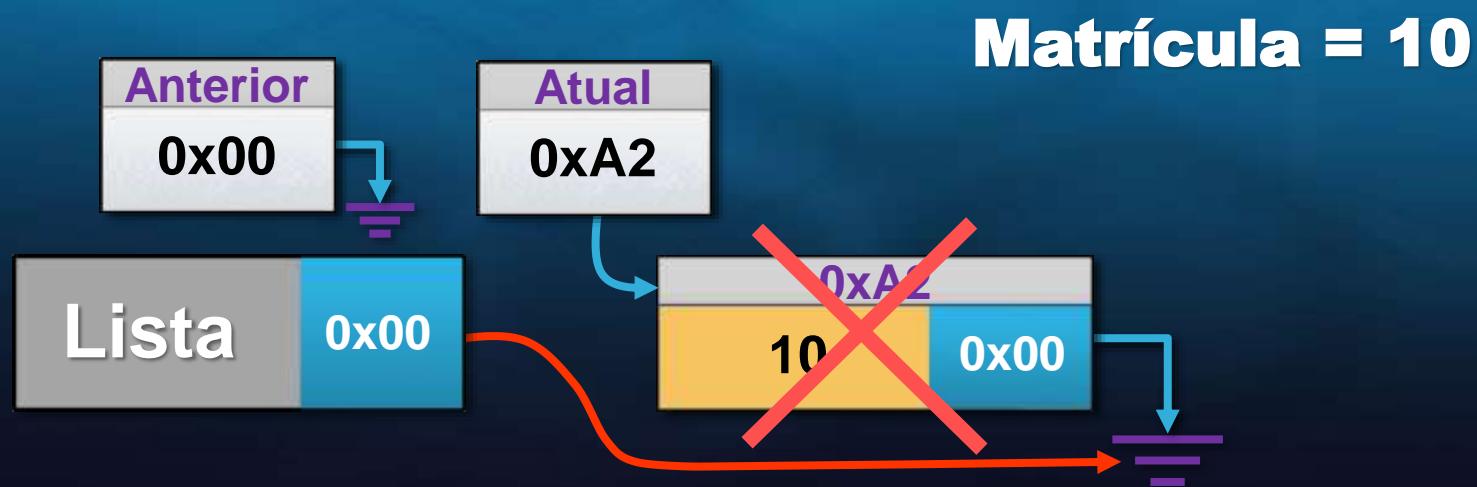
// Exclui o nó atual
delete ptrNoAtual;
```

Matrícula = 10



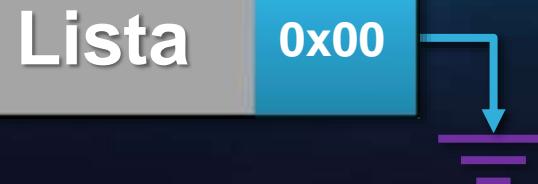
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

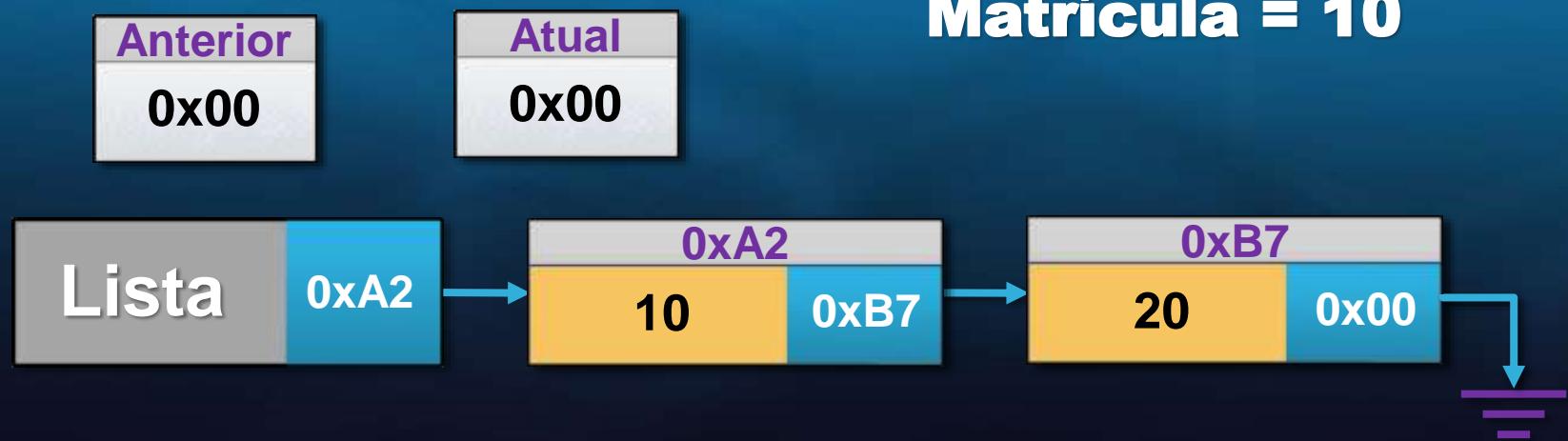
// Exclui o nó atual
delete ptrNoAtual;
```



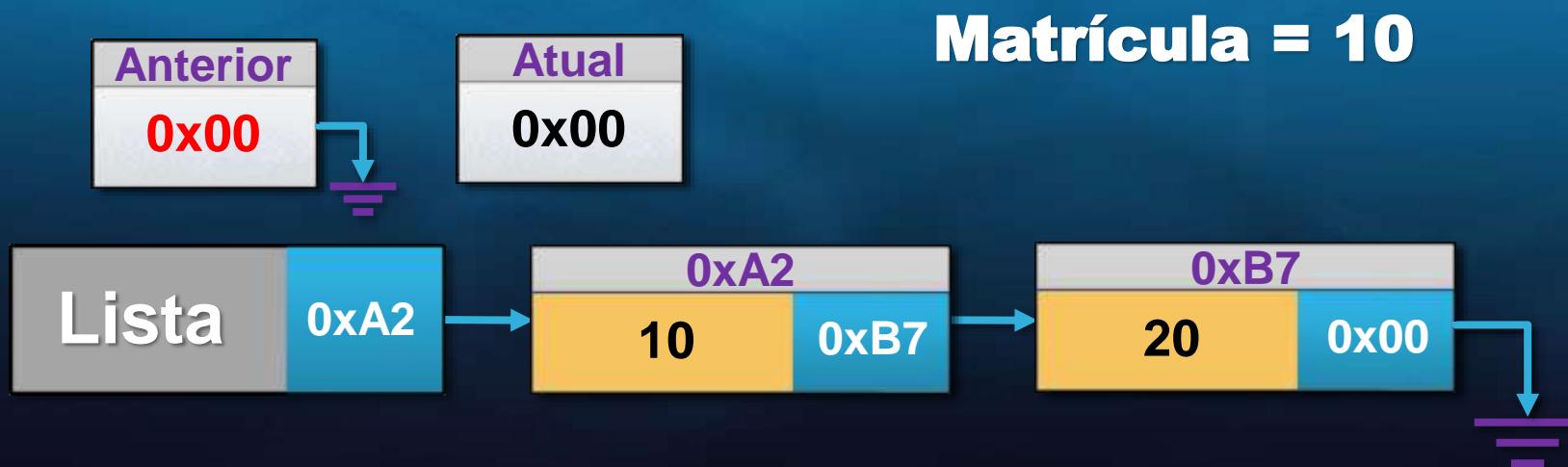
EXCLUIR ORDENADO NA LISTA

SIMULAÇÃO 2

Matrícula = 10



```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;  
  
// Localizao nó que será excluído  
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}  
  
if (ptrNoAtual == NULL) {  
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;  
    return false;  
}
```



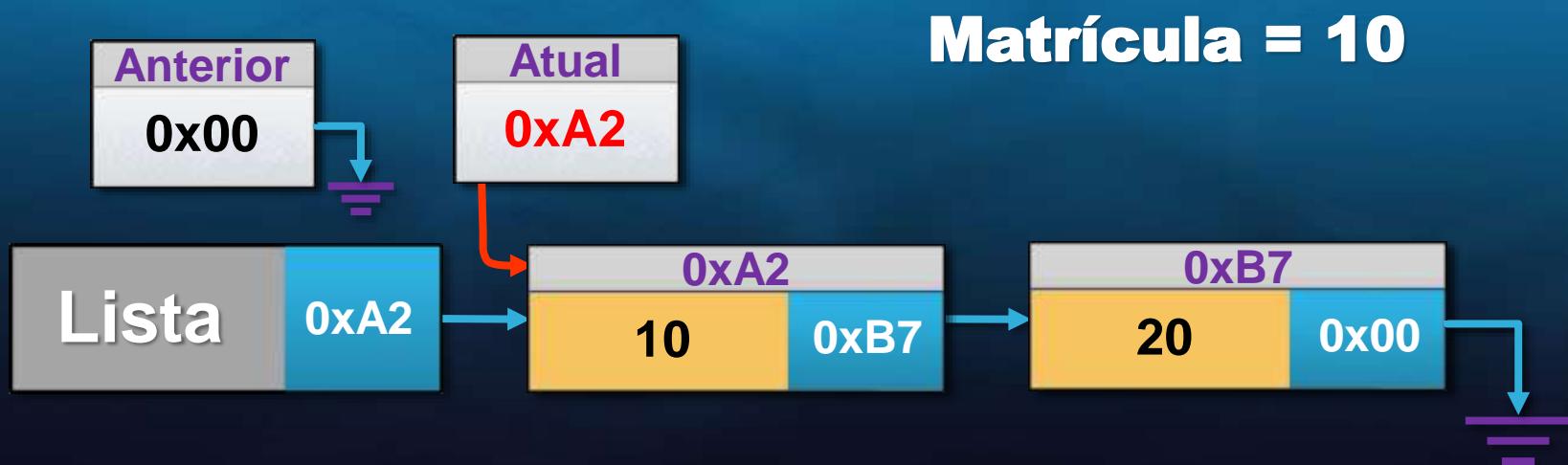
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



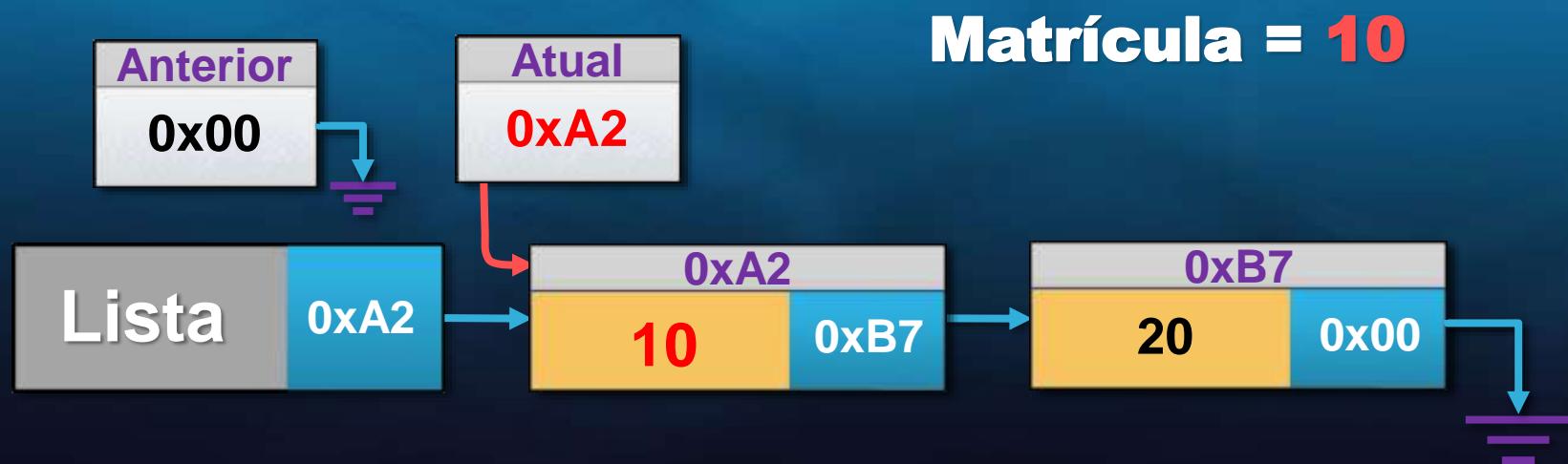
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;
```

```
// Localiza o nó que será excluído
```

```
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```

```
if (ptrNoAtual == NULL) {  
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;  
    return false;  
}
```

10 != 10



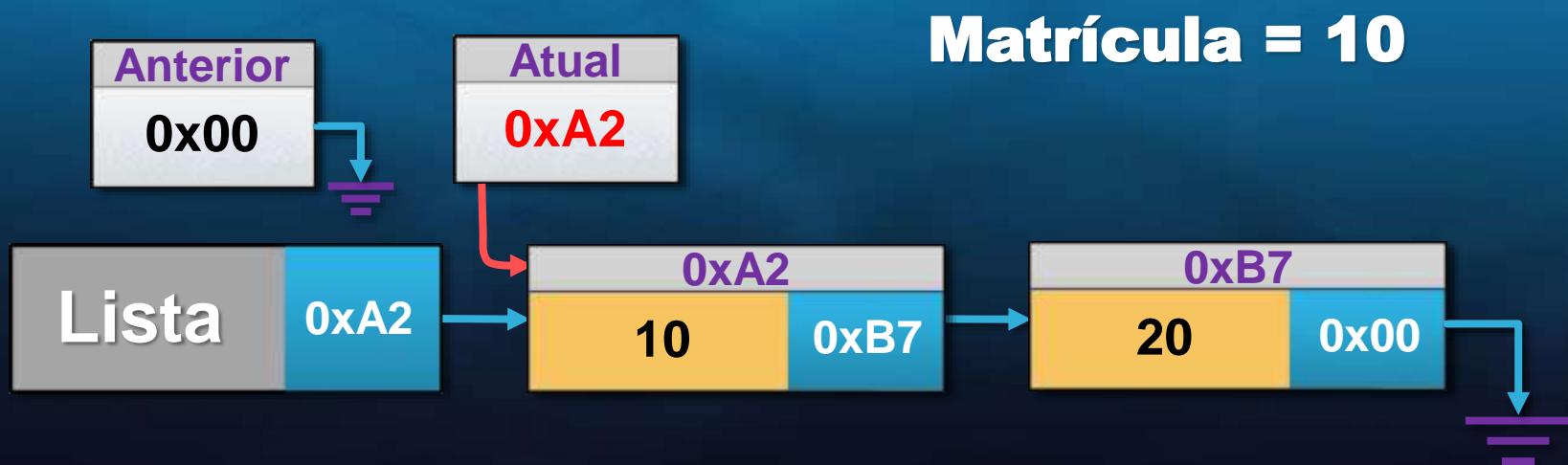
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

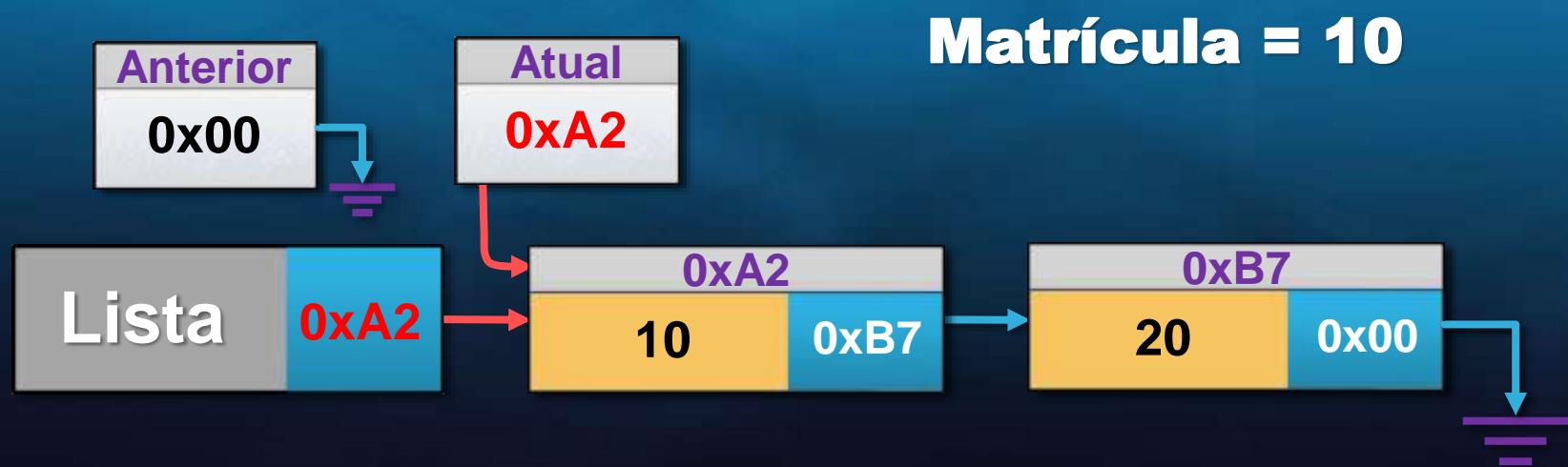
if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



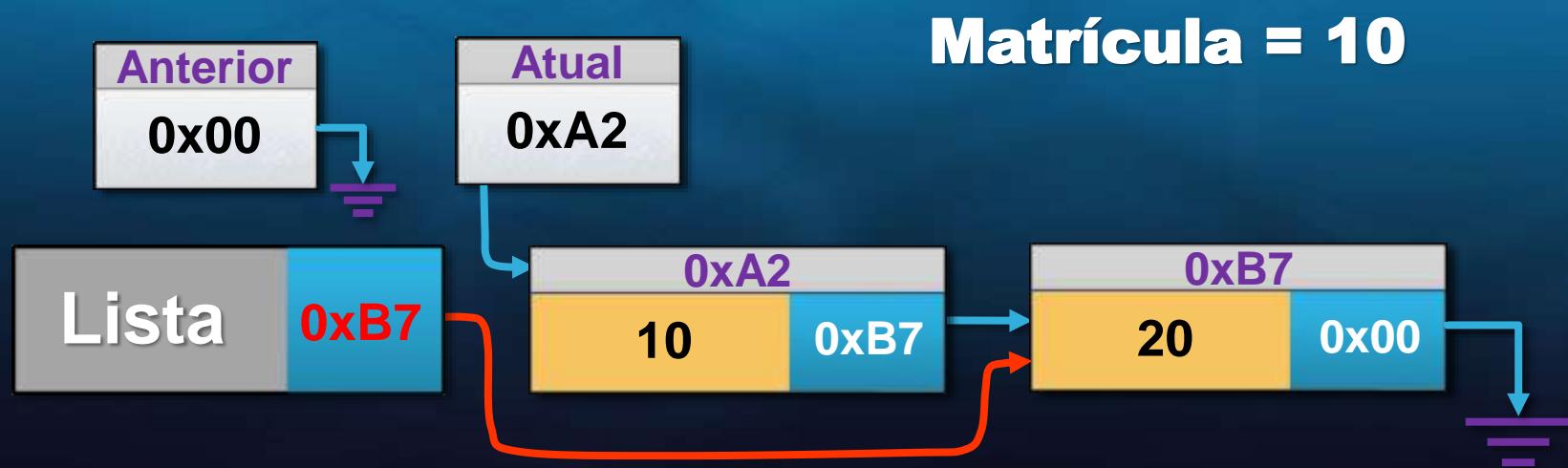
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```



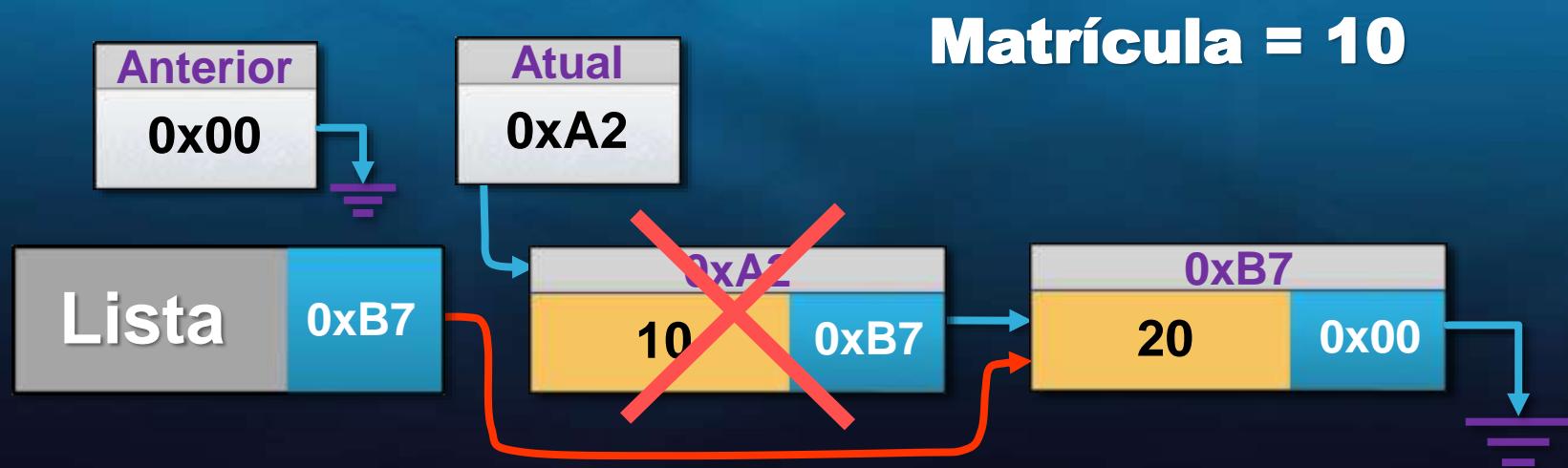
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```



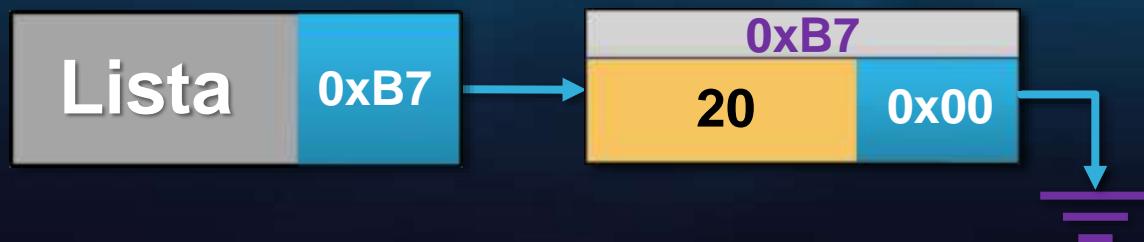
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

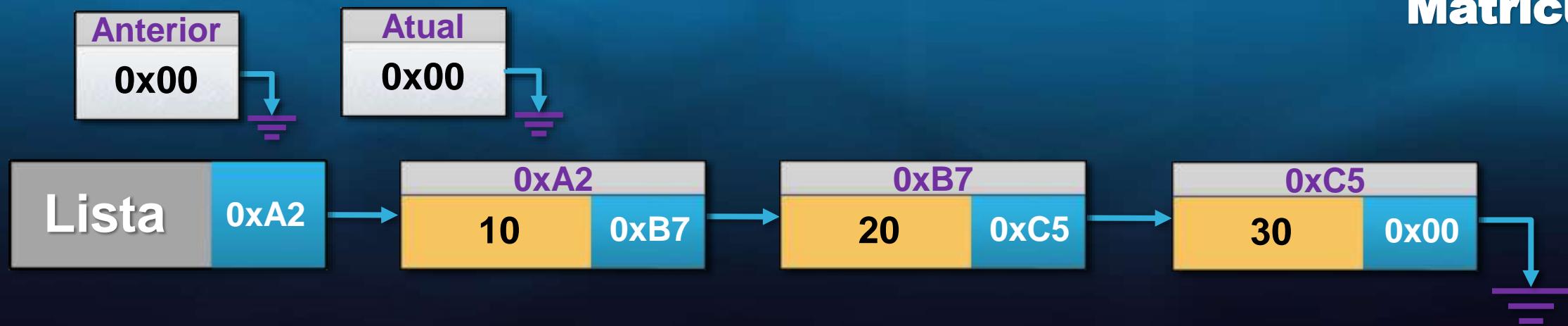
// Exclui o nó atual
delete ptrNoAtual;
```



EXCLUIR ORDENADO NA LISTA

SIMULAÇÃO 3

Matrícula = 20



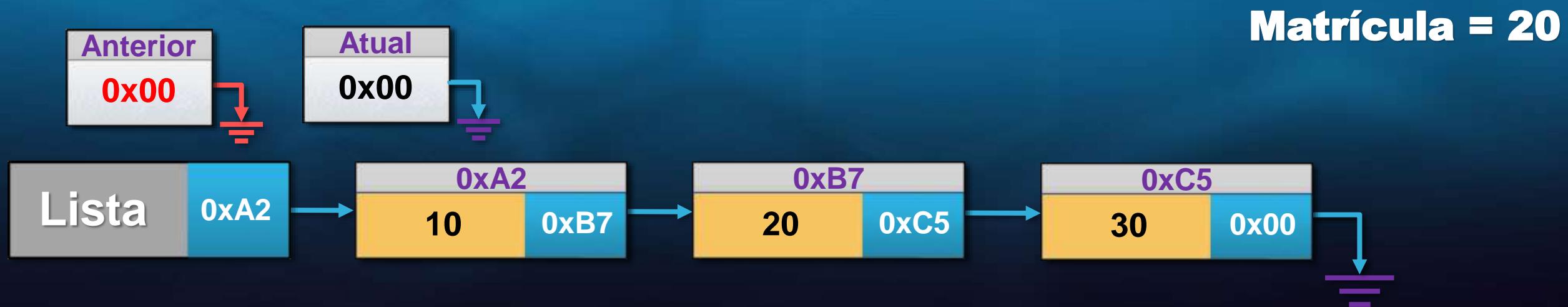
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



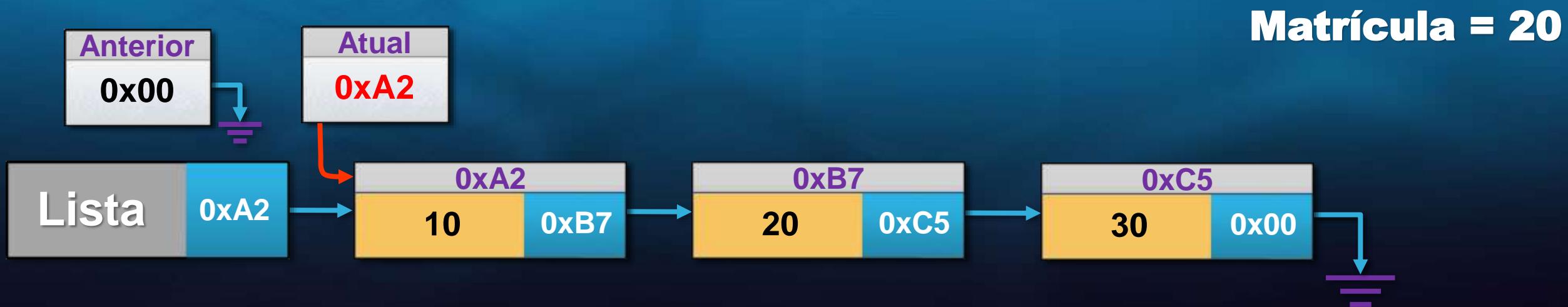
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;
```

// Localiza o nó que será excluído

```
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```

```
if (ptrNoAtual == NULL) {  
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;  
    return false;  
}
```

10 != 20



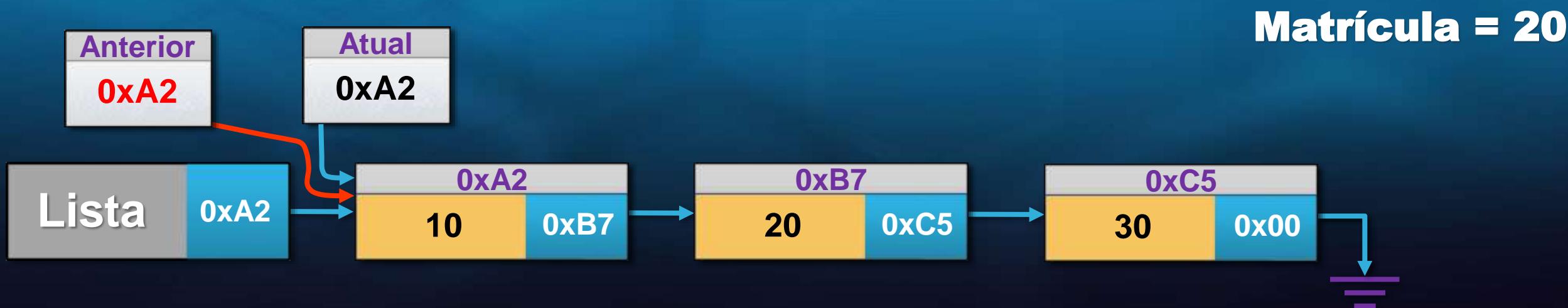
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



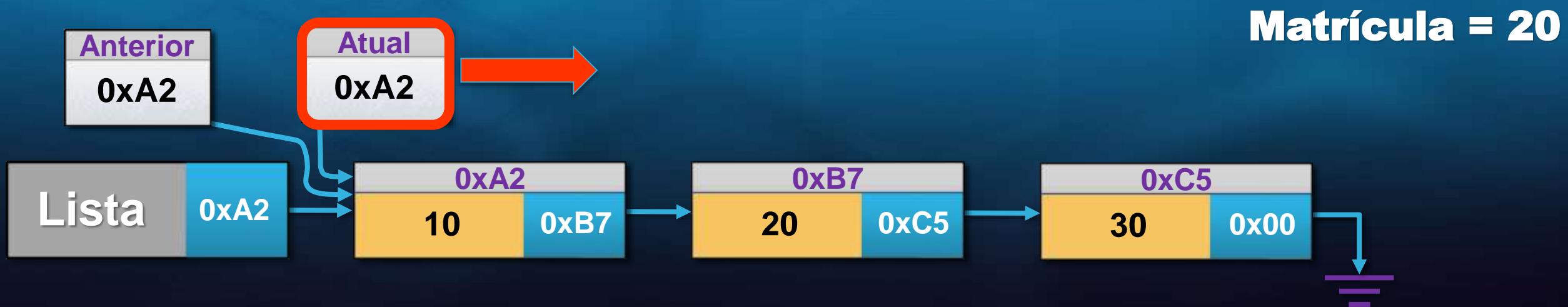
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



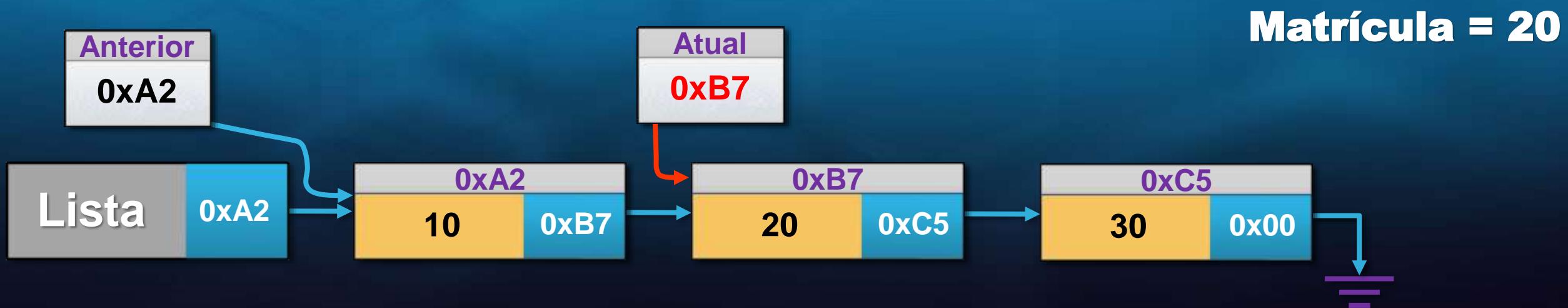
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



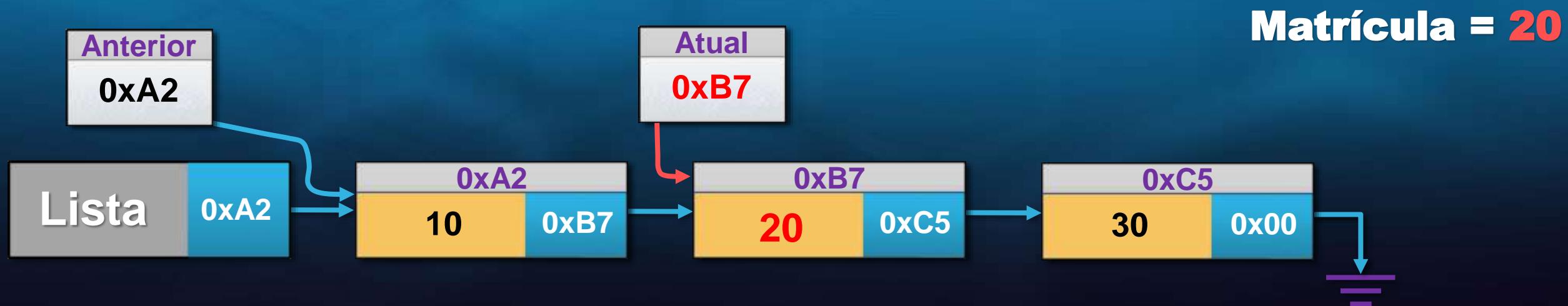
```
ptrNoAnterior = NULL;  
ptrNoAtual = ptrLista->inicio;
```

// Localiza o nó que será excluído

```
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {  
    ptrNoAnterior = ptrNoAtual;  
    ptrNoAtual = ptrNoAtual->proxNo;  
}
```

```
if (ptrNoAtual == NULL) {  
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;  
    return false;  
}
```

20 != 20



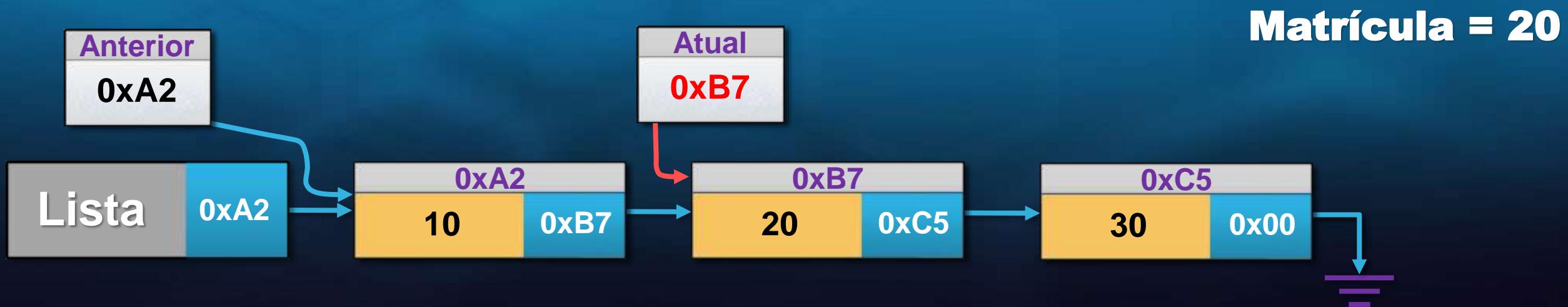
```

ptrNoAnterior = NULL;
ptrNoAtual = ptrLista->inicio;

// Localizao nó que será excluído
while (ptrNoAtual != NULL && ptrNoAtual->dadosAluno.matricula != matricula) {
    ptrNoAnterior = ptrNoAtual;
    ptrNoAtual = ptrNoAtual->proxNo;
}

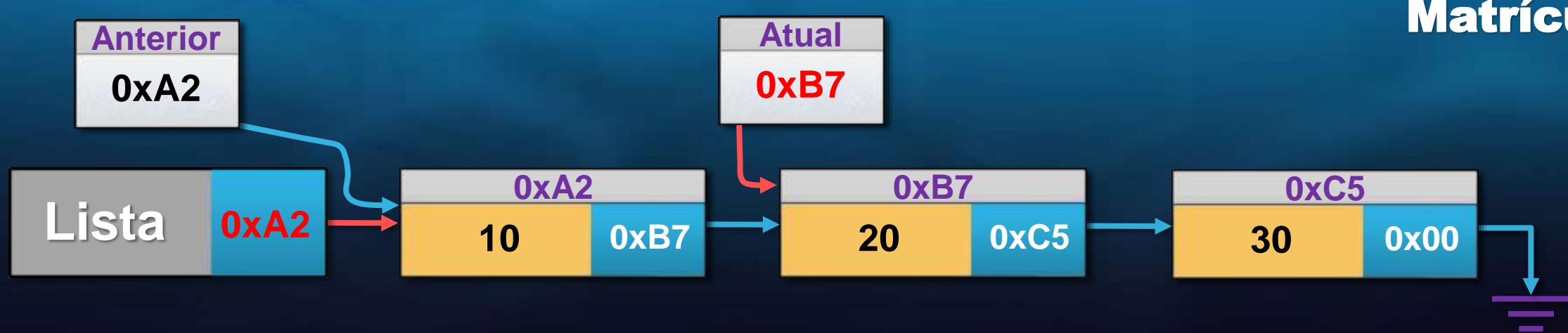
if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return false;
}

```



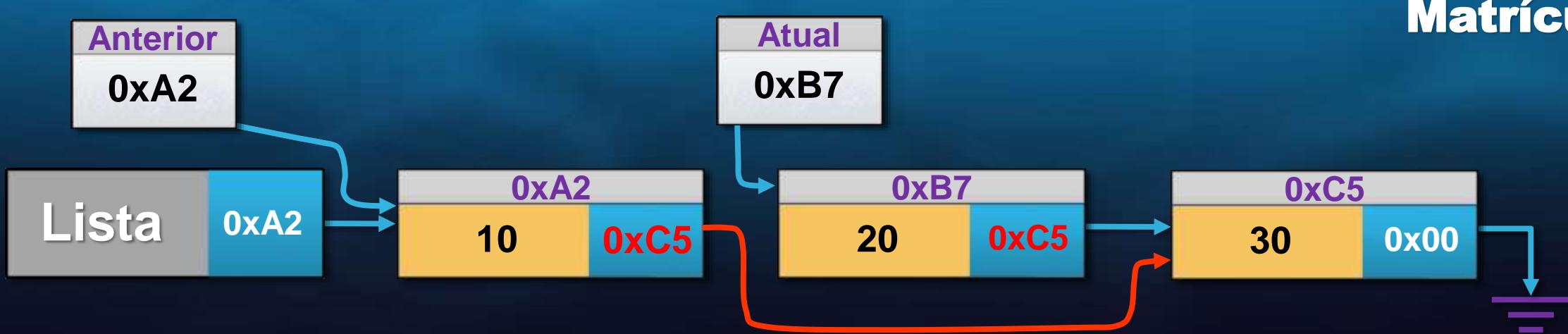
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```



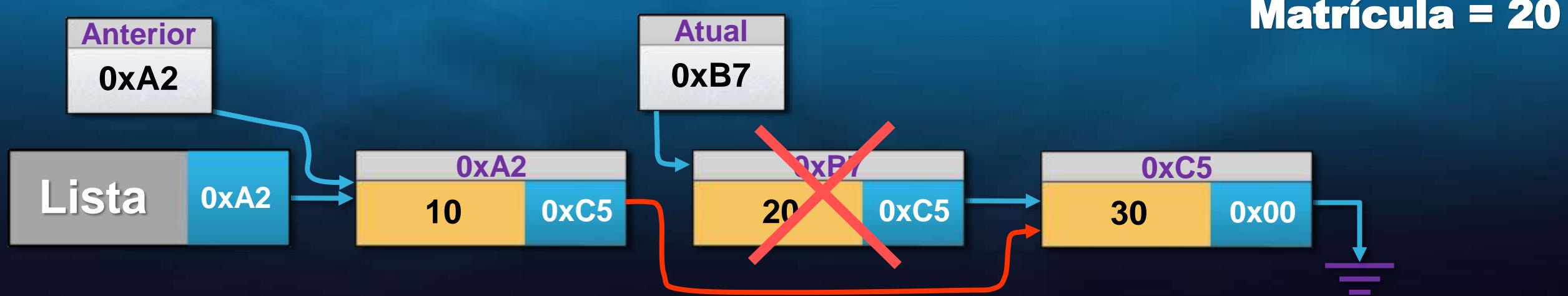
```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

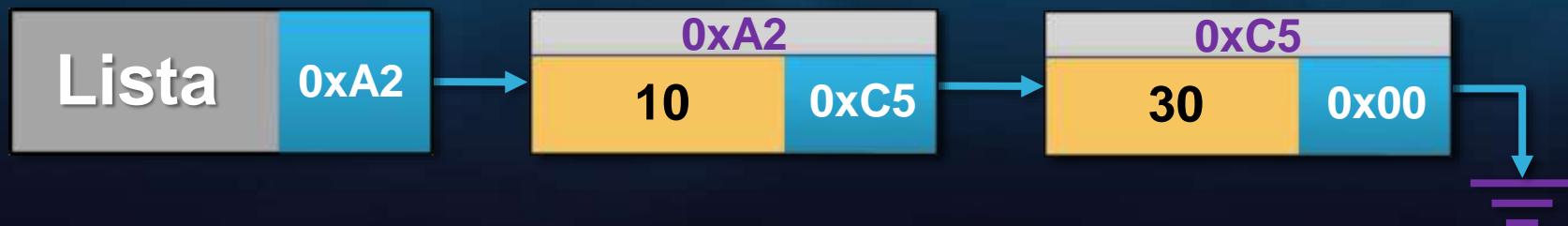
// Exclui o nó atual
delete ptrNoAtual;
```



```
// Se for o primeiro nó da lista
if (ptrNoAtual == ptrLista->inicio) {
    ptrLista->inicio = ptrNoAtual->proxNo;
}
else {
    ptrNoAnterior->proxNo = ptrNoAtual->proxNo;
}

// Exclui o nó atual
delete ptrNoAtual;
```

Matrícula = 20



ALTERAR NÓ DA LISTA

SIMULAÇÃO 1

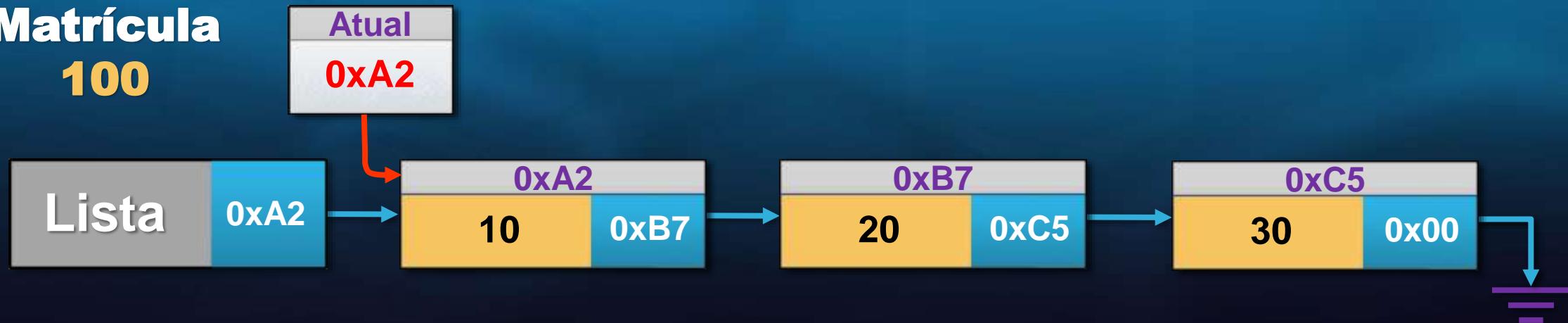
```
ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;
```

Matrícula
100



```

ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

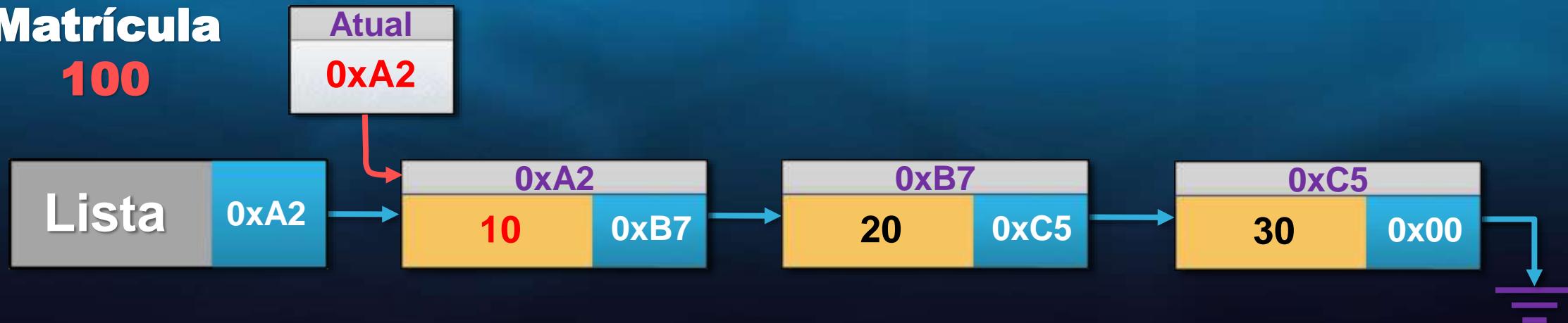
if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

10 != 100

**Matrícula
100**



```

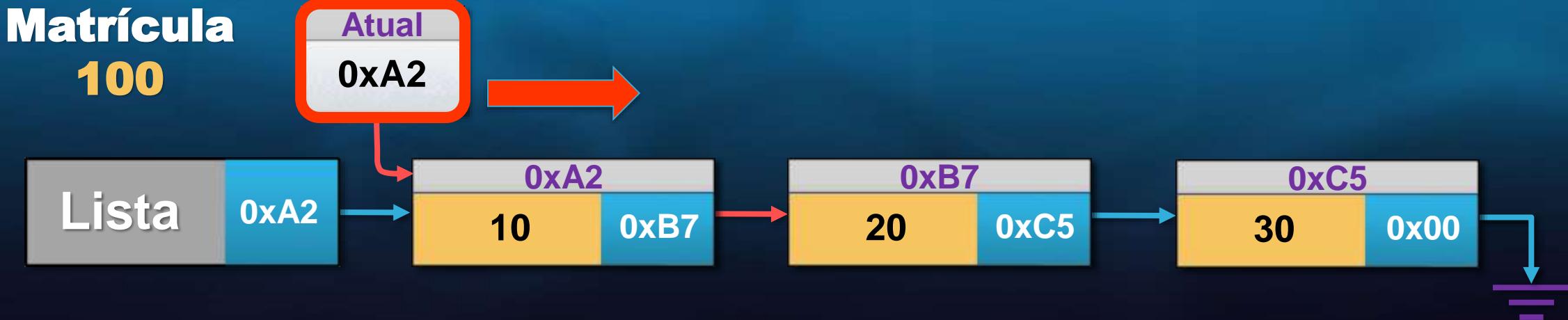
ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```



```

ptrNoAtual = ptrLista->inicio;

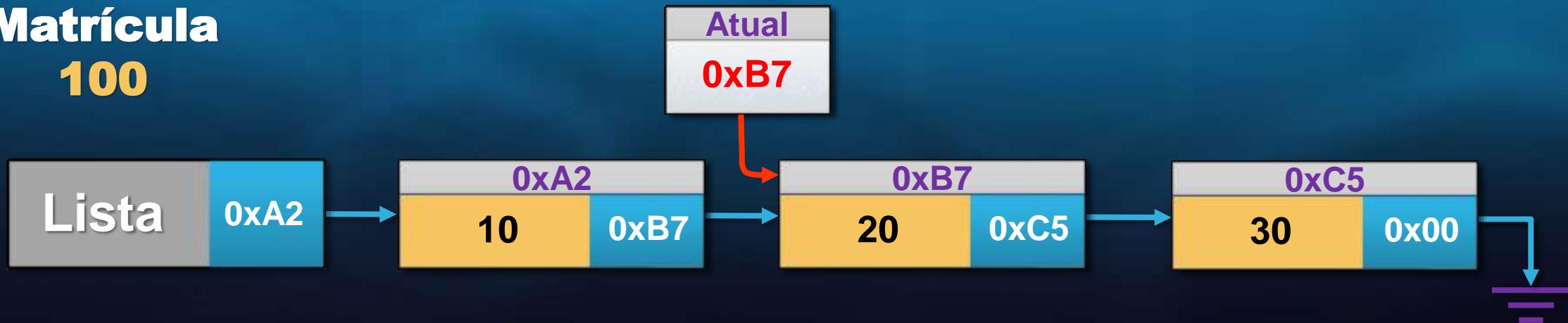
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

**Matrícula
100**



```

ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

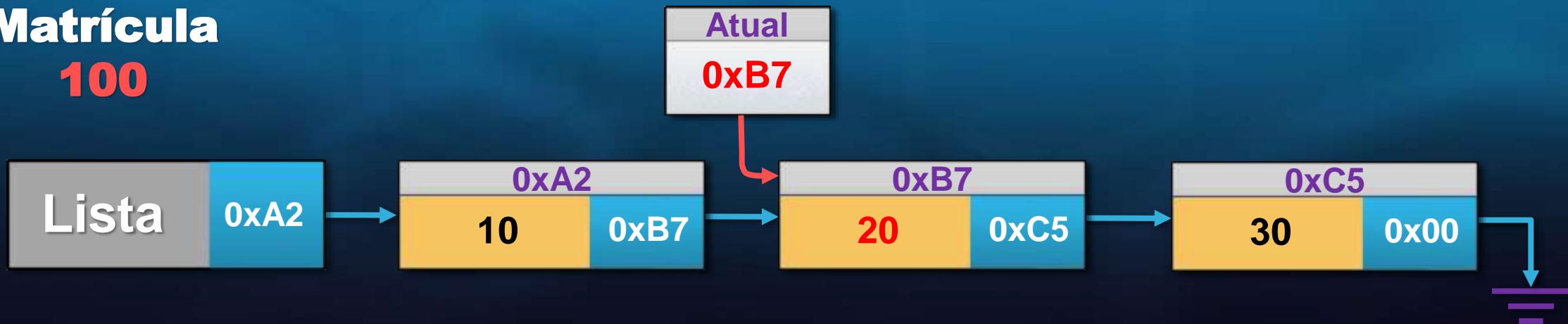
if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

20 != 100

**Matrícula
100**



```

ptrNoAtual = ptrLista->inicio;

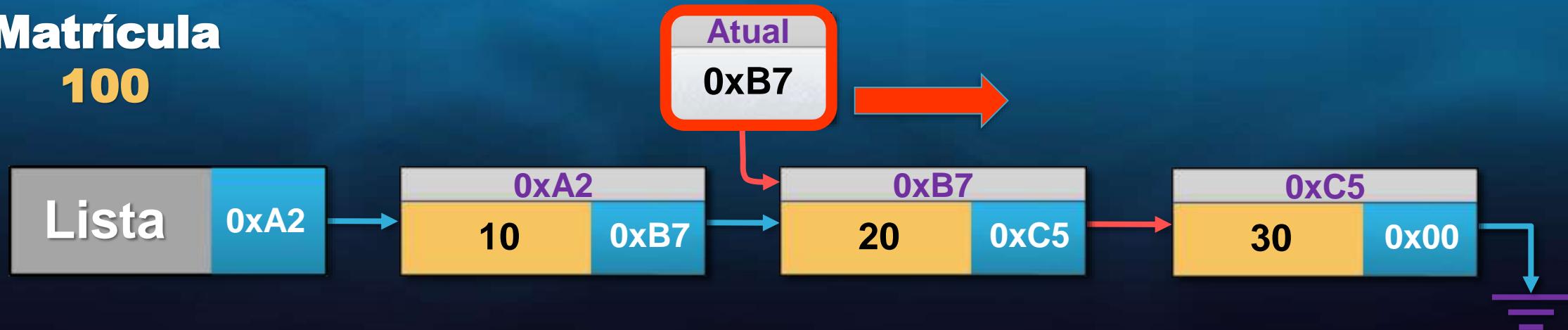
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

**Matrícula
100**



```

ptrNoAtual = ptrLista->inicio;

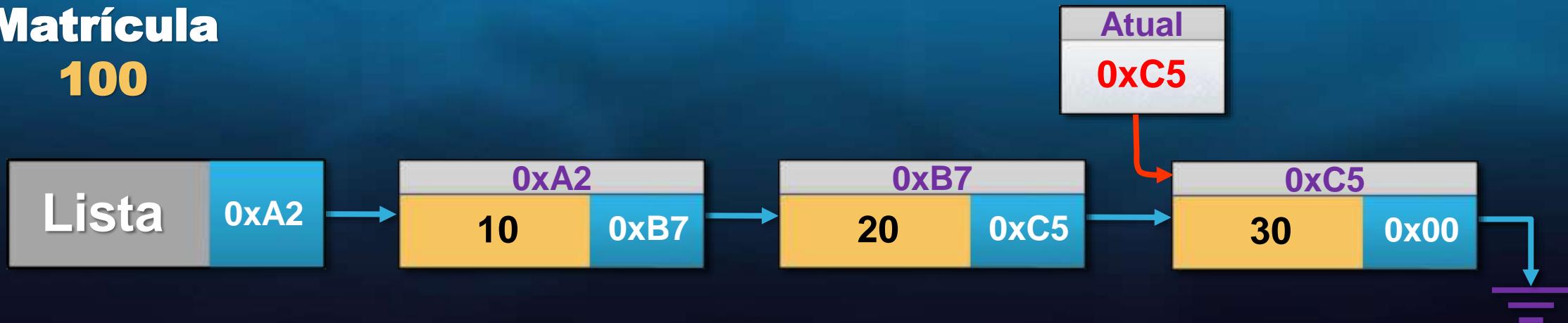
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

Matrícula 100



```

ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

30 != 100

**Matrícula
100**



```

ptrNoAtual = ptrLista->inicio;

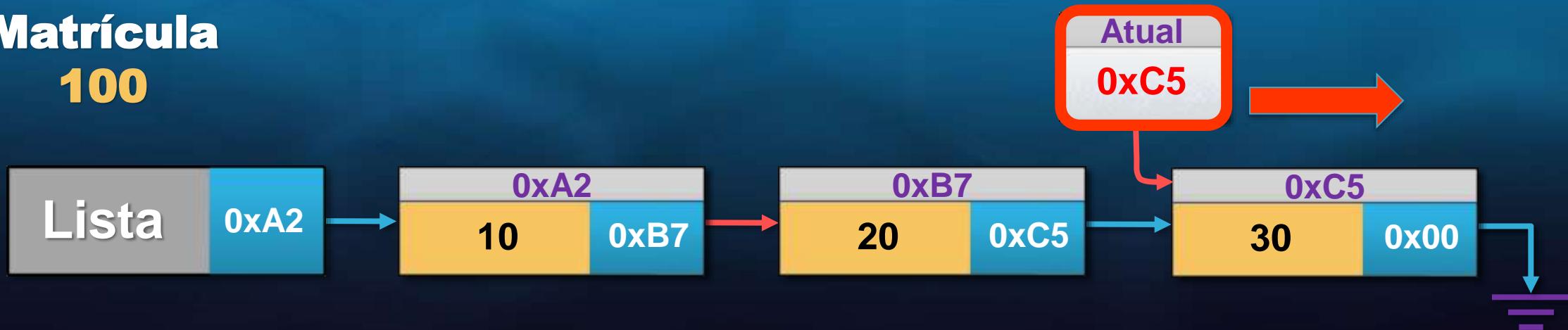
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

**Matrícula
100**



```

ptrNoAtual = ptrLista->inicio;

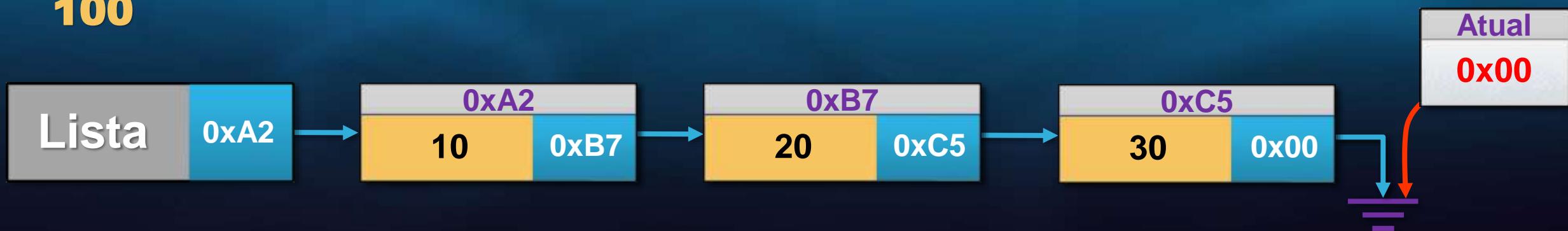
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

Matrícula 100

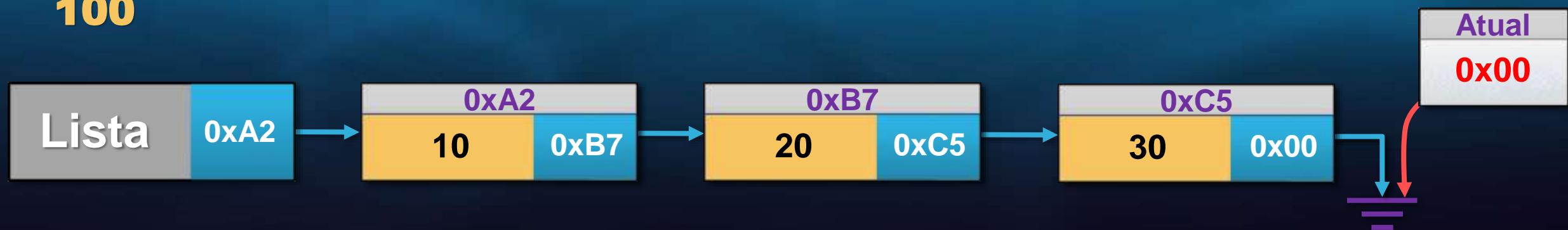


```
ptrNoAtual = ptrLista->inicio;
```

FALSO

```
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {  
    ptrNoAtual = ptrNoAtual->proxNo;  
}  
  
if (ptrNoAtual == NULL) {  
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;  
    return NULL;  
}  
  
return ptrNoAtual;
```

**Matrícula
100**



```

ptrNoAtual = ptrLista->inicio;

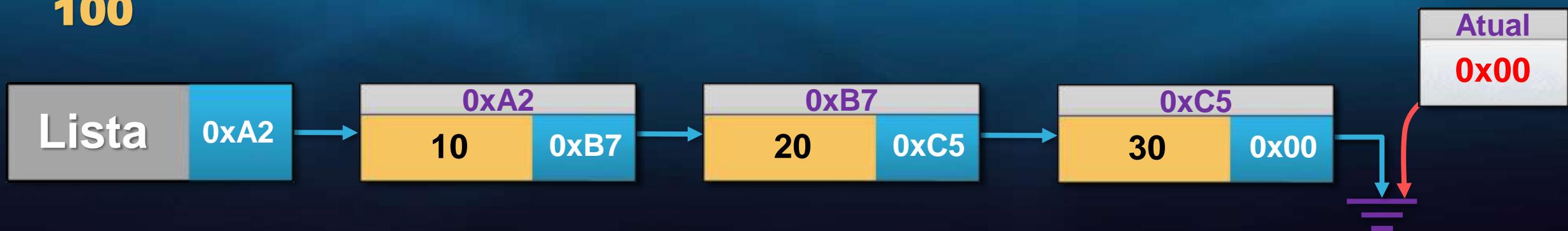
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

Matrícula 100



```

ptrNoAtual = ptrLista->inicio;

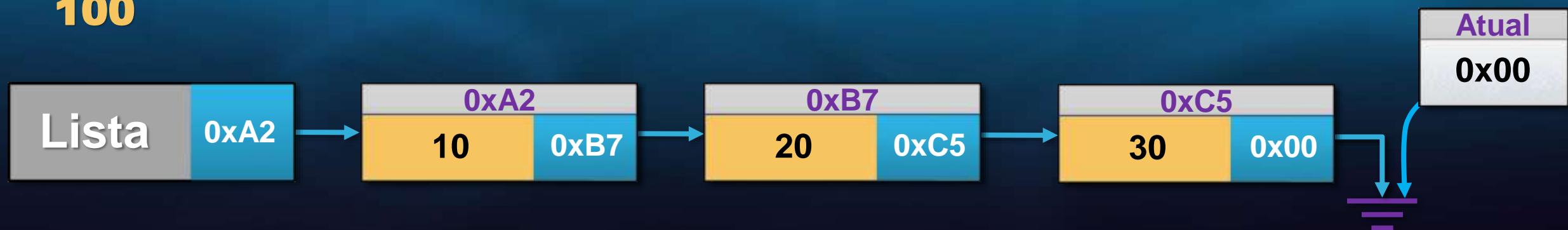
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

Matrícula 100



```

ptrNoAtual = ptrLista->inicio;

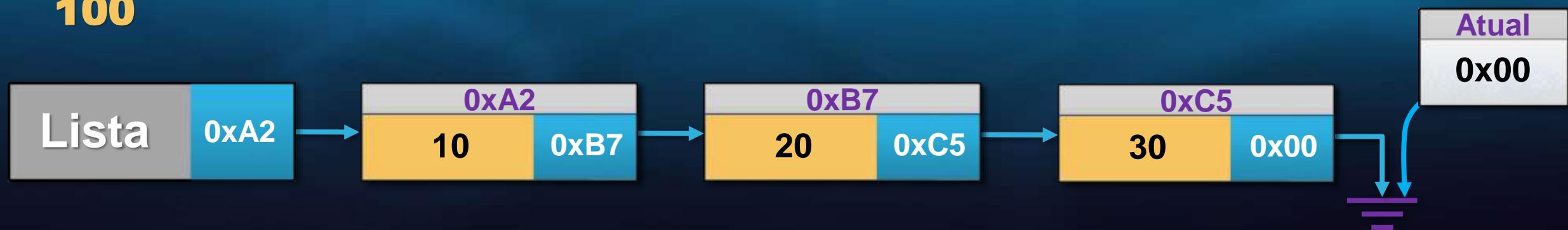
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

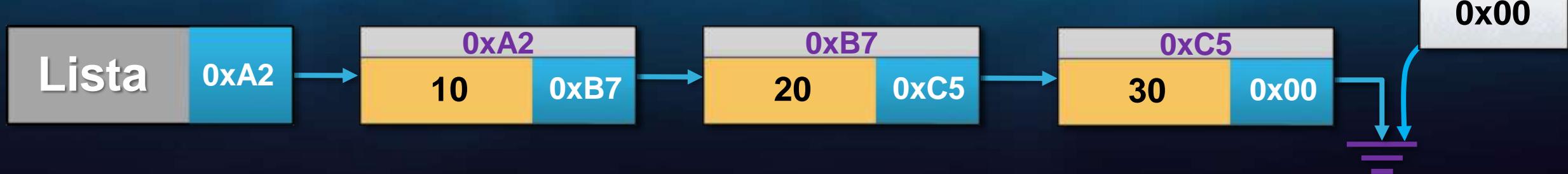
Matrícula 100



```
No *pNoAluno;  
NULL  
pNoAluno = pesquisarAluno(pLista, 20);
```

```
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

**Matrícula
100**

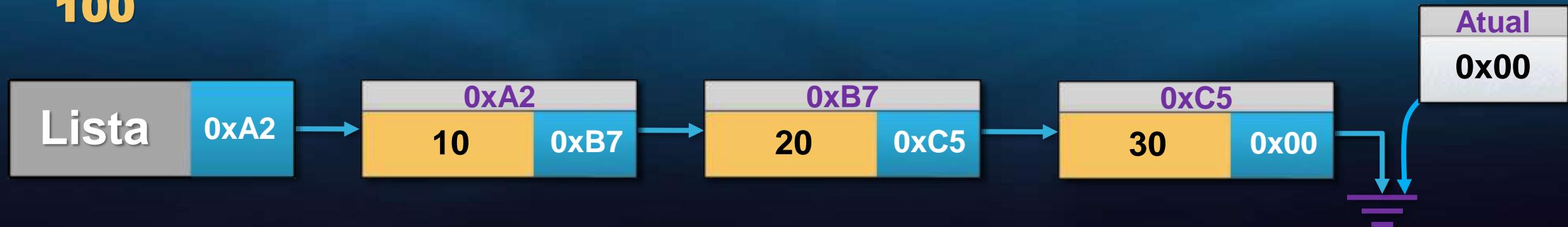


```
No *pNoAluno;
```

```
pNoAluno = pesquisarAluno(pLista, 20);
```

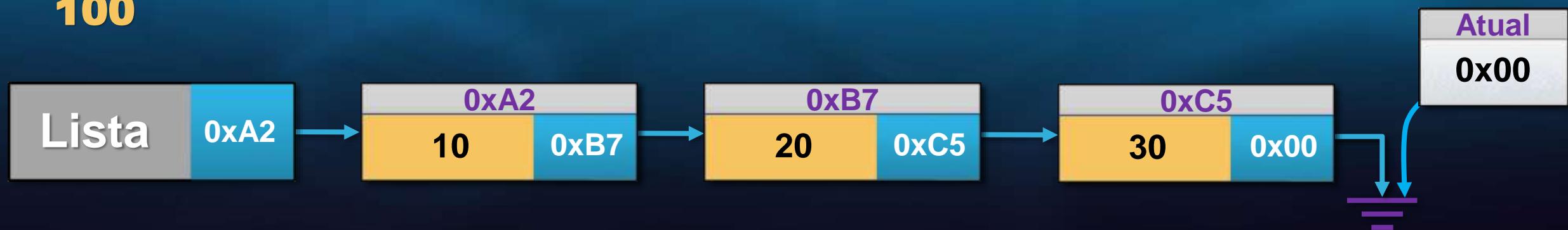
```
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

Matrícula 100



```
No *pNoAluno;  
  
pNoAluno = pesquisarAluno(pLista, 20);  
  
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

Matrícula 100



ALTERAR NÓ DA LISTA

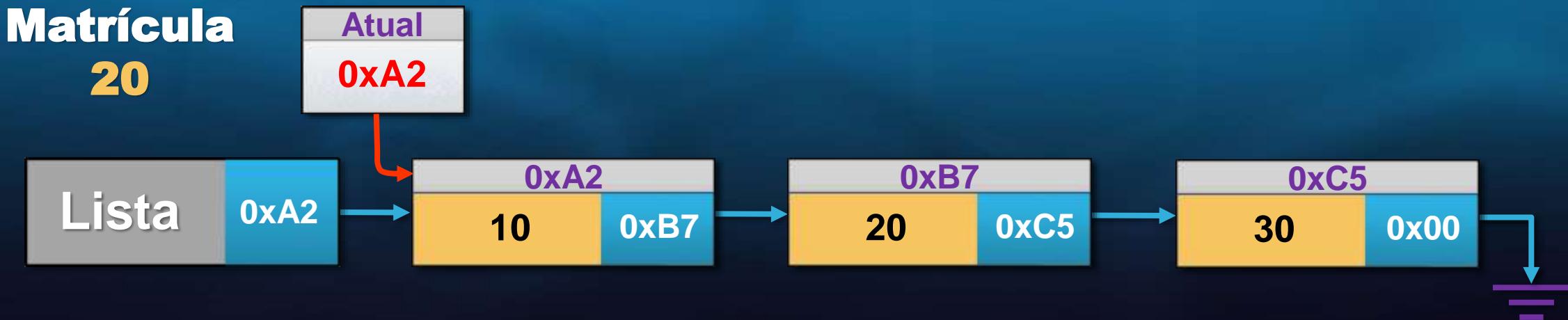
SIMULAÇÃO 2

```
ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;
```



```

ptrNoAtual = ptrLista->inicio;

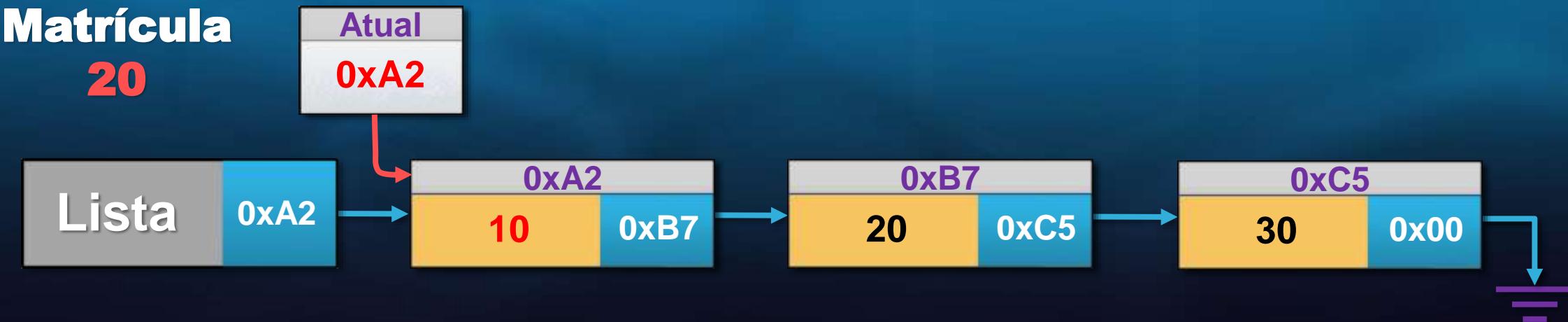
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

10 != 100



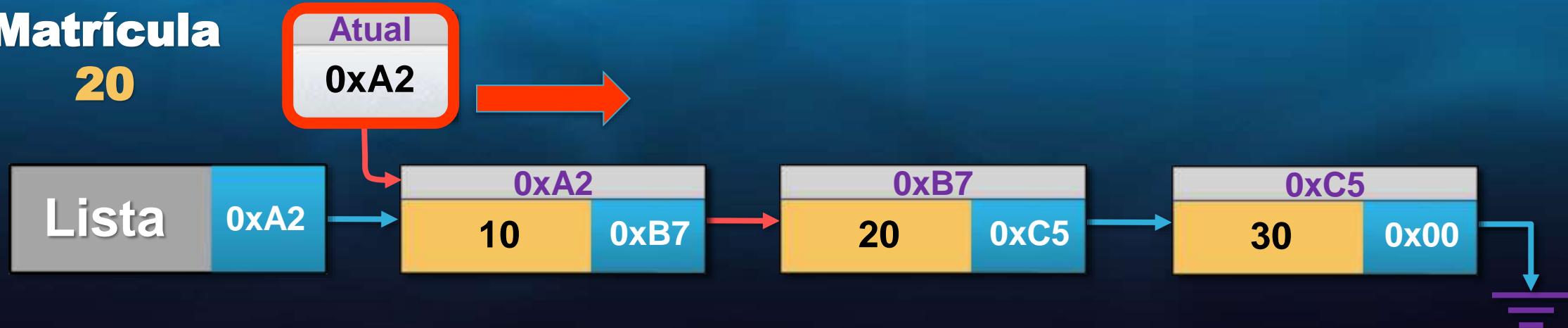
```
ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;
```

Matrícula
20



```

ptrNoAtual = ptrLista->inicio;

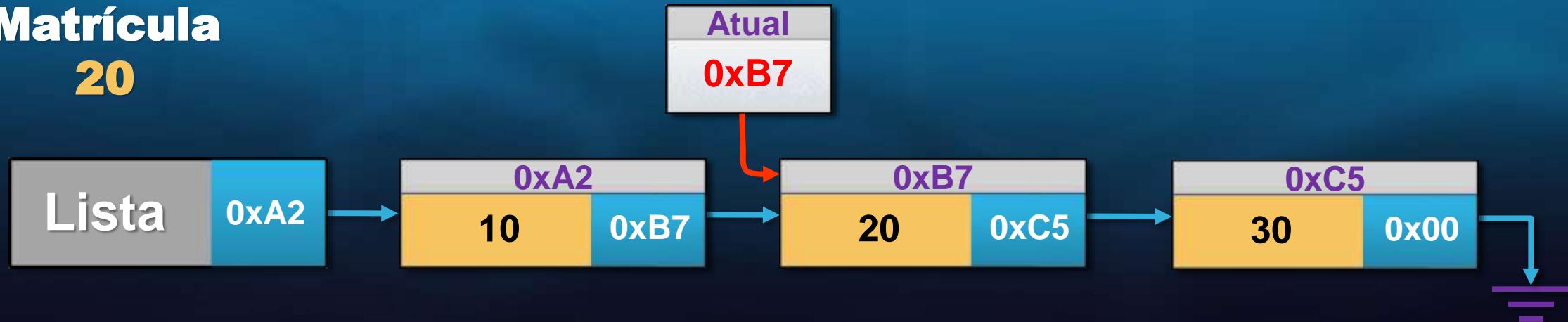
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

**Matrícula
20**



```

ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

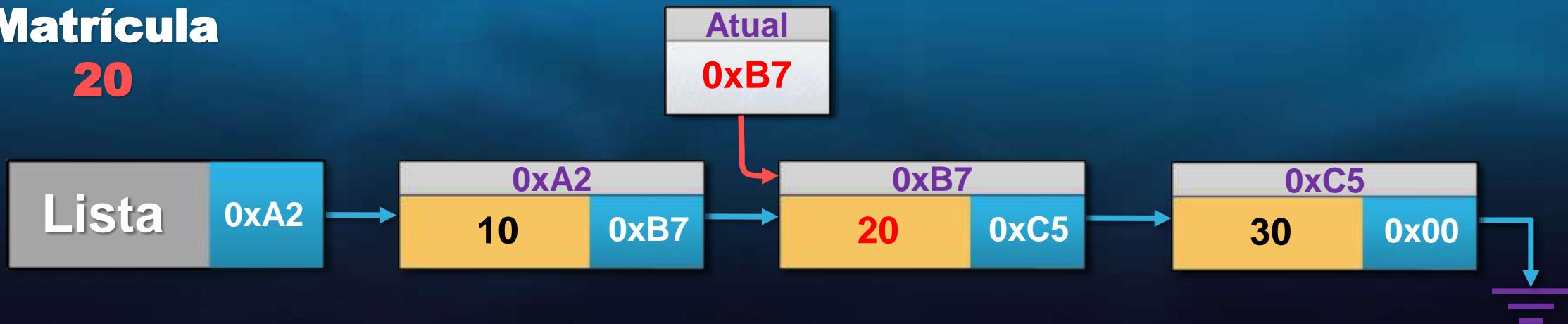
if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

20 != 20

**Matrícula
20**



```

ptrNoAtual = ptrLista->inicio;

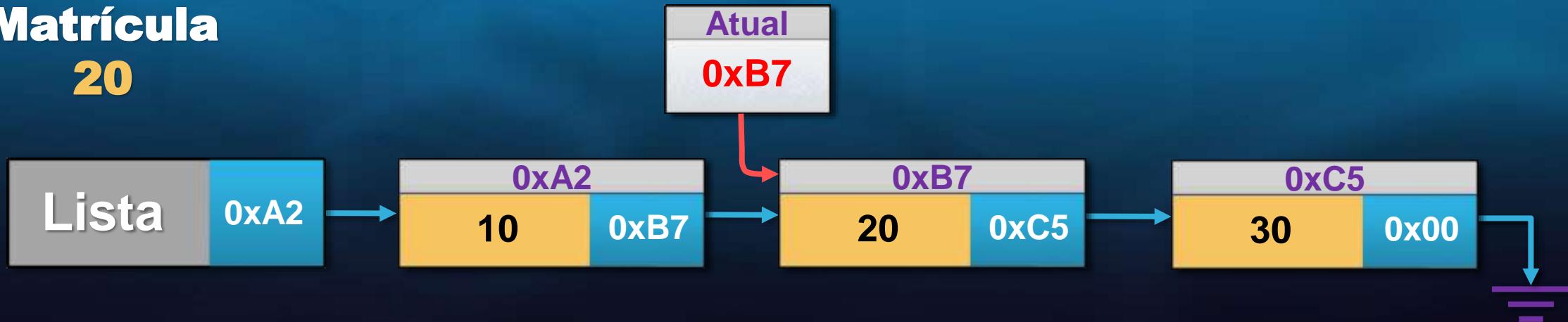
// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

return ptrNoAtual;

```

**Matrícula
20**



```

ptrNoAtual = ptrLista->inicio;

// Localiza o nó a ser alterado
while (ptrNoAtual != NULL && ptrNoAtual->dados.matricula != matricula) {
    ptrNoAtual = ptrNoAtual->proxNo;
}

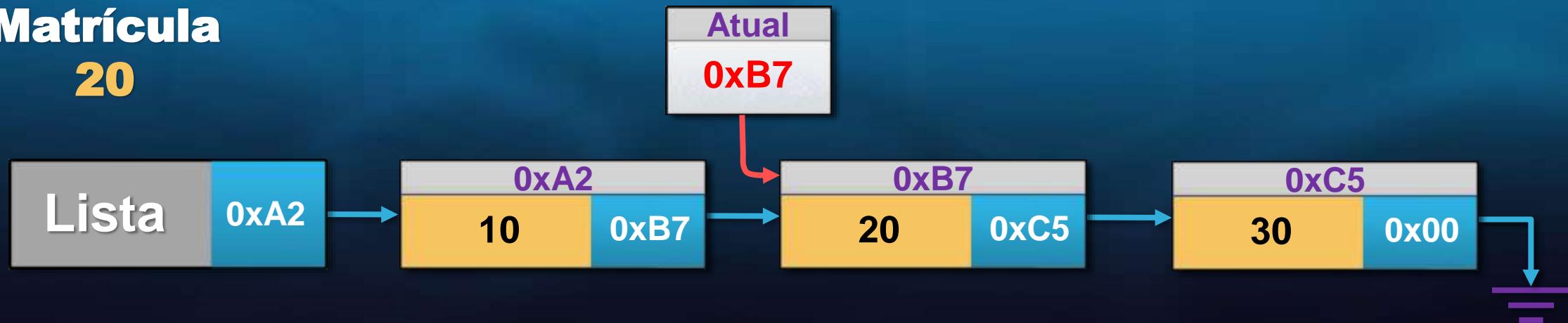
if (ptrNoAtual == NULL) {
    cout << "A matrícula " << matricula << " não foi encontrada!" << endl;
    return NULL;
}

```

0xB7

return ptrNoAtual;

**Matrícula
20**



```
No *pNoAluno;
```



```
pNoAluno = pesquisarAluno(pLista, 20);
```

```
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

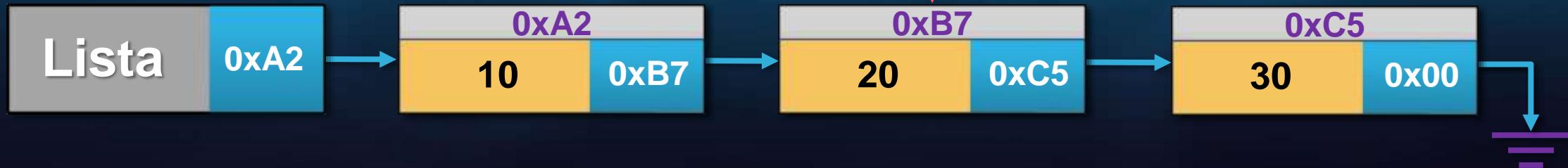
Matrícula
20



```
No *pNoAluno;  
0xB7  
pNoAluno = pesquisarAluno(pLista, 20);
```

```
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

Matrícula
20

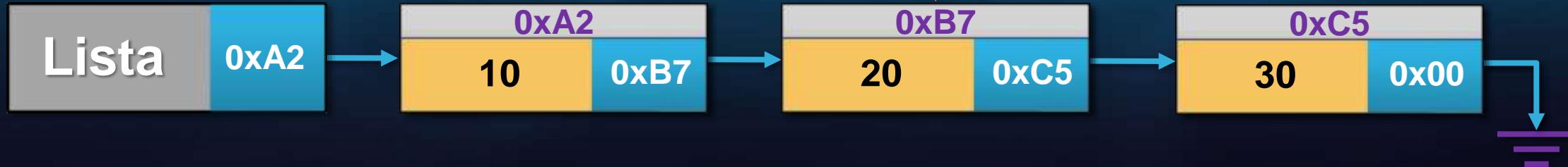


```
No *pNoAluno;
```

```
pNoAluno = pesquisarAluno(pLista, 20);
```

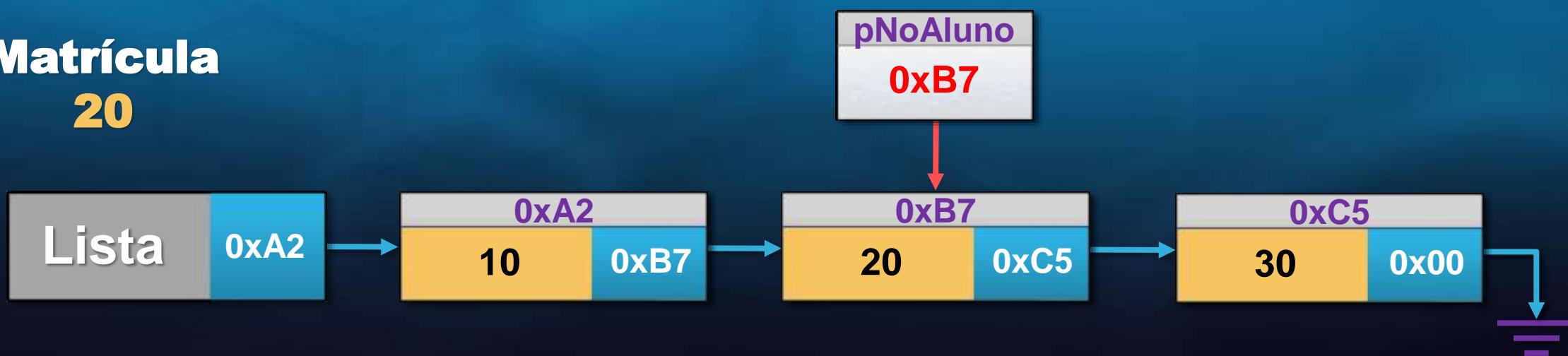
```
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

Matrícula
20



```
No *pNoAluno;  
  
pNoAluno = pesquisarAluno(pLista, 20);  
  
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;  
}
```

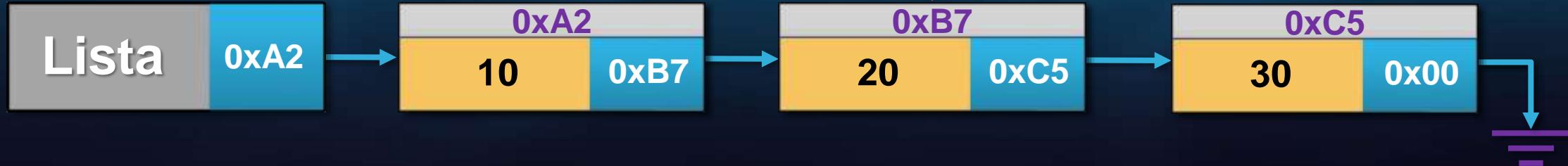
Matrícula
20

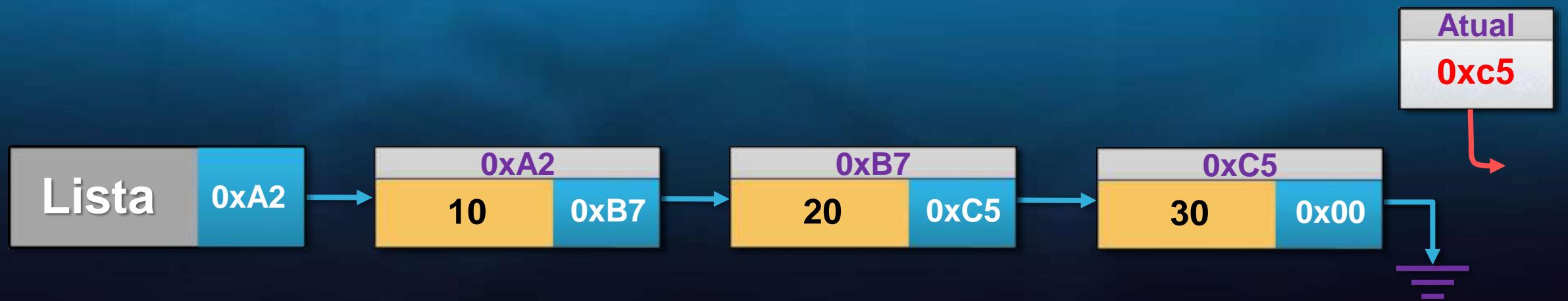


```
No *pNoAluno;  
  
pNoAluno = pesquisarAluno(pLista, 20);  
  
if (pNoAluno != NULL) {  
    pNoAluno->dados.nome = "José Maria";  
    pNoAluno->dados.media = 8.0f;
```

}

Matrícula
20





Exercício – A02

Insira, via programação, 10 alunos a uma lista. Em seguida, crie 3 funções, todas recebem como parâmetro a lista.

- A primeira deve encontrar e imprimir a maior média.
 - A segunda deve encontrar a imprimir a menor média.
 - A terceira deve calcular e imprimir a média geral.
-
- **OBS.:** copie o algoritmo de **lista simplesmente encadeada** para um novo arquivo e altere-o com as solicitações do exercício.

Exercício – A03

- Cria duas listas chamadas DiscBD e DiscED. Insira, programaticamente, 3 alunos na lista DiscBD, e os mesmos 3 em DiscED. Crie uma terceira lista vazia (MediaGeral).
- Crie uma função que receba como parâmetros as 3 listas, calcule a média entre DiscBD e DiscED e coloque o resultado em MediaGeral.
- Após a função, imprima a lista MediaGeral;



Exercício – A04

- Crie uma função que concatene as listas DiscBD e DiscED, do exercício anterior, em uma nova lista (Curso).
- Após a função, imprima a lista MediaGeral;



Exercício – A05

- Crie uma nova lista encadeada para guardar os dados de veículo: Código do veículo, nome do proprietário, ano de fabricação e ano do modelo.

```
// Dados
struct Dados {
    int codVeic;
    string nome;
    int anoFab;
    int anoModelo;
};
```

Exercício – A06

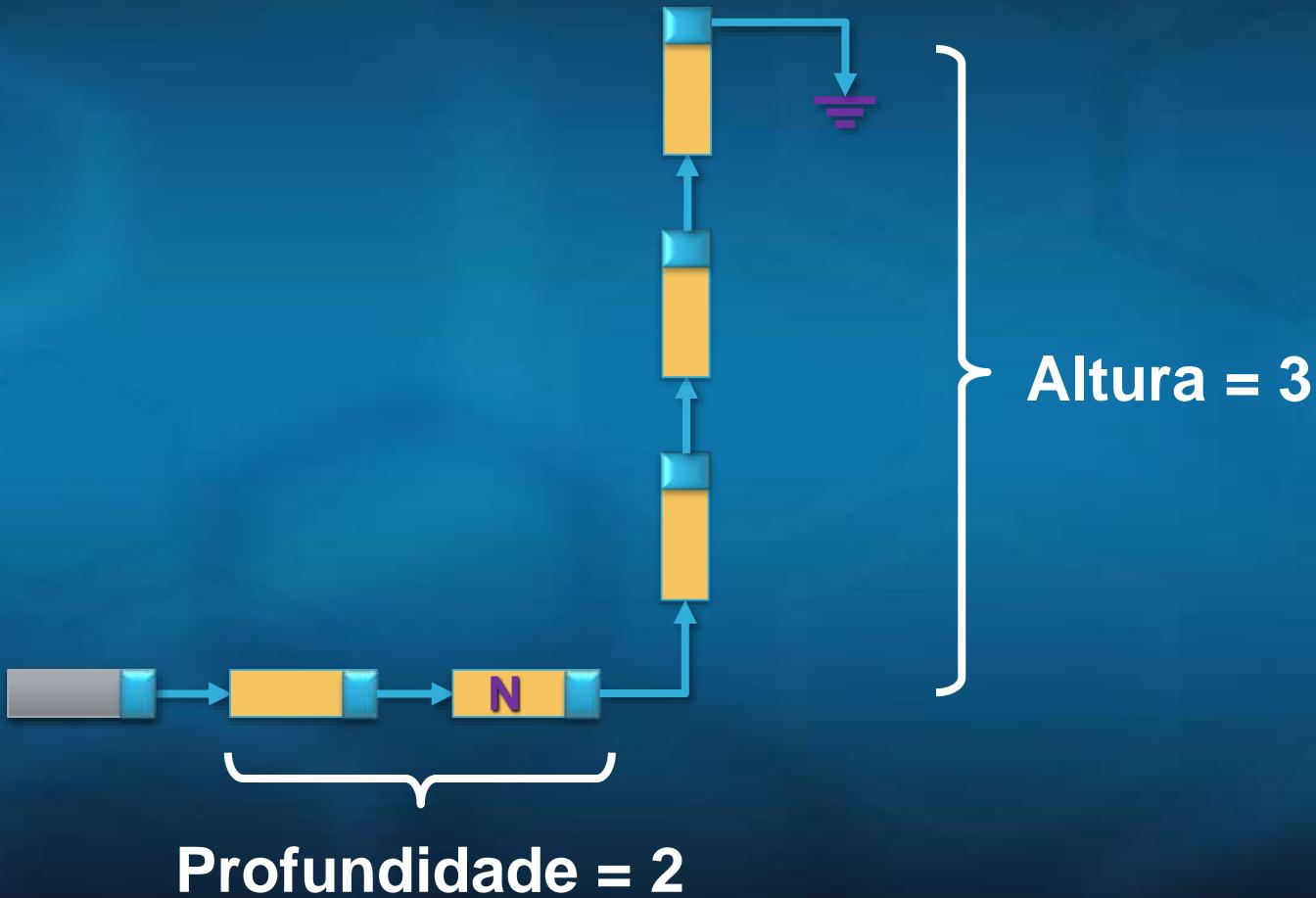
- Acrescente a lista de veículo, criada no exercício anterior, uma variável membro na estrutura de dados para indicar que o venda do veículo.
- Crie uma função chamada de vendas que quando for chamada, deve atribuir true para a variável vendido.
 - `void vender(Lista* ptrLista, int codVeic);`
- Crie duas funções. Uma para exibir os veículos em estoque (`vendido = false`) e outra para exibir os vendidos (`vendido = true`).

```
// Dados
struct Dados {
    int codVeic;
    string nome;
    int anoFab;
    int anoModelo;
    bool vendido = false;
};
```

Trabalho

- Escreva uma função que conte o número de nós de uma lista encadeada.
- **Altura.** A altura de um nó **N** em uma lista encadeada é a distância entre **N** e o **fim** da lista. Mais precisamente, a altura de **N** é o número de passos do caminho que leva de **N** até o último nó da lista. Escreva uma função que calcule a altura de um dado nó.
- **Profundidade.** A profundidade de um nó **N** em uma lista encadeada é o número de passos do único caminho que vai do **primeiro** nó da lista até **N**. Escreva uma função que calcule a profundidade de um dado nó.

Profundidade x Altura



Pensamento

"O pensamento lógico pode levar você de A a B, mas a imaginação te leva a qualquer parte do Universo."

(Albert Einstein)

FIM

Prof. Dr Ricardo Luis Balieiro