

listaDinamica

October 22, 2021

1 Listas Dinâmica

1.1 Implementação

```
[15]: #include <iostream>
using namespace std;
```

1.2 Definição da classe

```
[16]: class Lista {
private:
    struct elemento {
        int valor;
        elemento *proximoElemento;
    };
    typedef elemento *PonteiroElemento;
    PonteiroElemento inicio;
    int contador;
public:
    Lista();
    bool vazia();
    bool cheia();
    bool inserir(int posicao, int x);
    bool remover(int posicao, int &x);
    //implementar
    bool substituir(int posicao, int x);
    string listar();
    int tamanho();
    bool retornar(int posicao, int &x);
    bool localizar(int &posicao, int x);
    bool localizarUltimo(int &posicao, int x);
private:
    bool setaPosicao(int posicao, PonteiroElemento &atual);
};
```

1.2.1 Método Construtor

```
[17]: Lista::Lista() {  
    inicio = nullptr;  
    contador = 0;  
}
```

```
[18]: Lista minhaLista;
```



1.2.2 Verifica se lista está vazia

```
[19]: bool Lista::vazia() {  
    return inicio == nullptr;  
}
```

```
[20]: if (minhaLista.vazia()) {  
    cout << "Está vazia!";  
}
```

Está vazia!

1.2.3 Verifica se lista está cheia

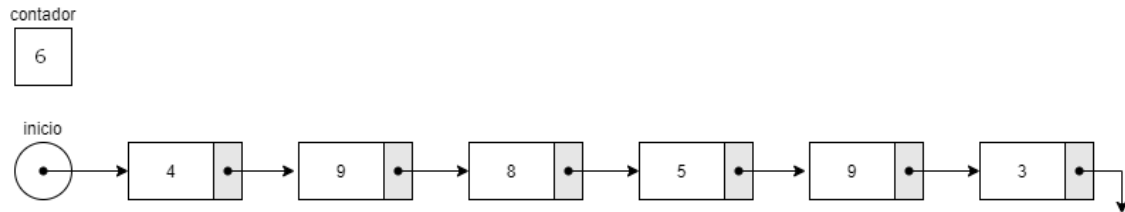
```
[21]: bool Lista::cheia() {  
    return false;  
}
```

```
[22]: if (!minhaLista.cheia()) {  
    cout << "Não está cheia!";  
}
```

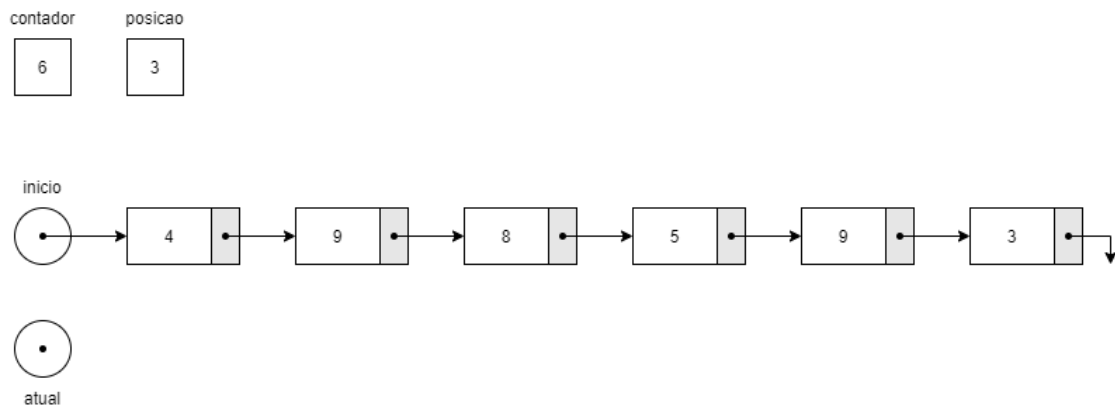
Não está cheia!

1.2.4 Método Auxiliar: setaPosição

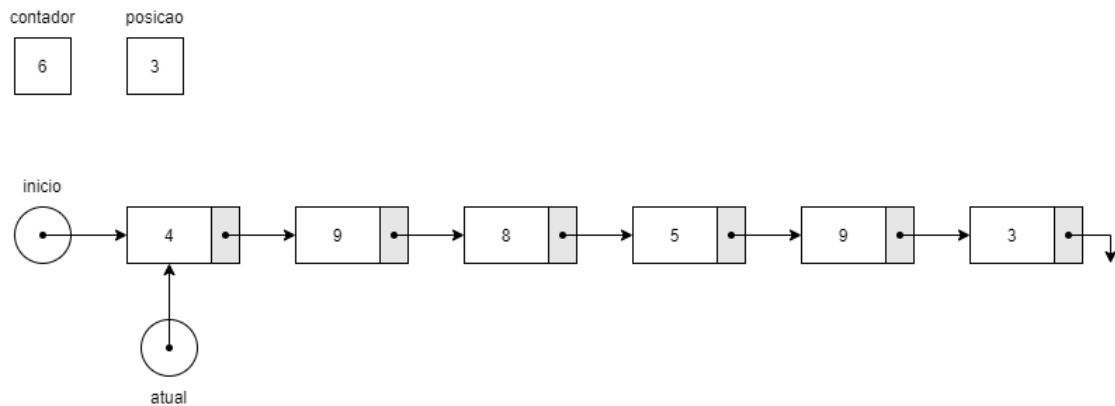
```
[23]: bool Lista::setaPosicao(int posicao, PonteiroElemento &atual) {  
    atual = inicio;  
    for(int i = 1 ; i < posicao ; i++) {  
        atual = atual->proximoElemento;  
    }  
    return true;  
}
```



```
PonteiroElemento p;  
minhaLista.setaPosicao(3, p);
```

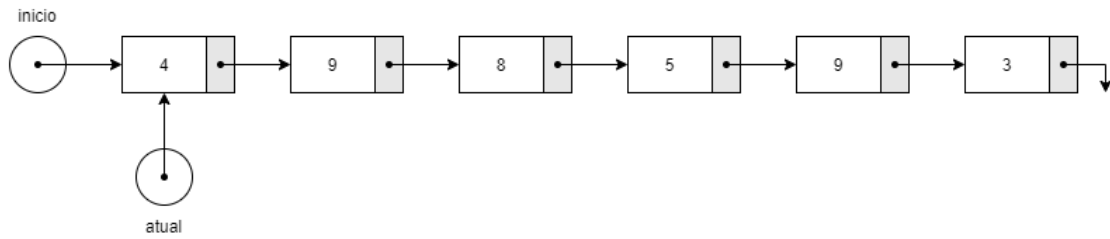


```
atual = inicio;
```



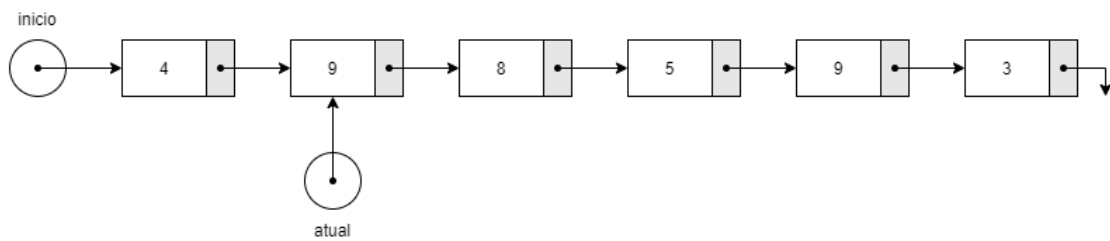
```
for(int i = 1 ; i < posicao ; i++) {
```

contador	posicao	i
6	3	1



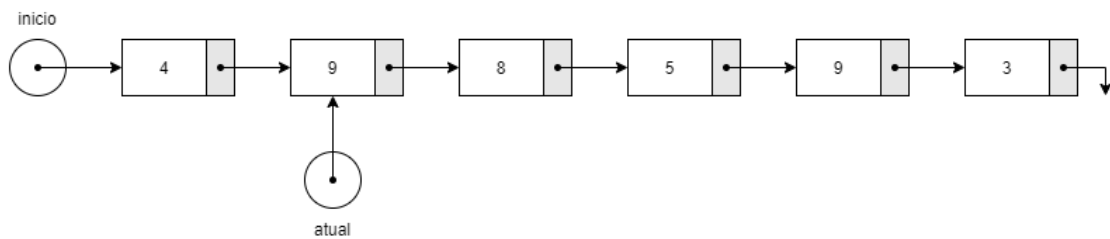
```
atual = atual->proximoElemento;
```

contador	posicao	i
6	3	1

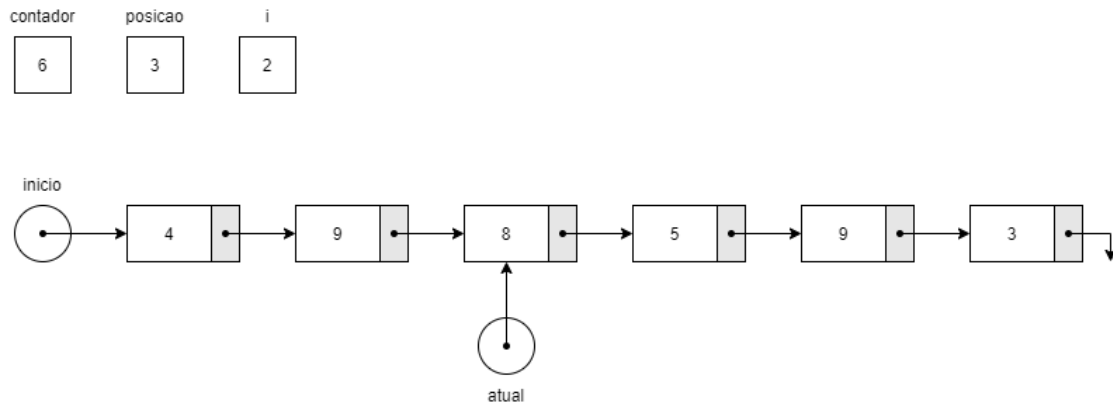


```
for(int i = 1 ; i < posicao ; i++) {
```

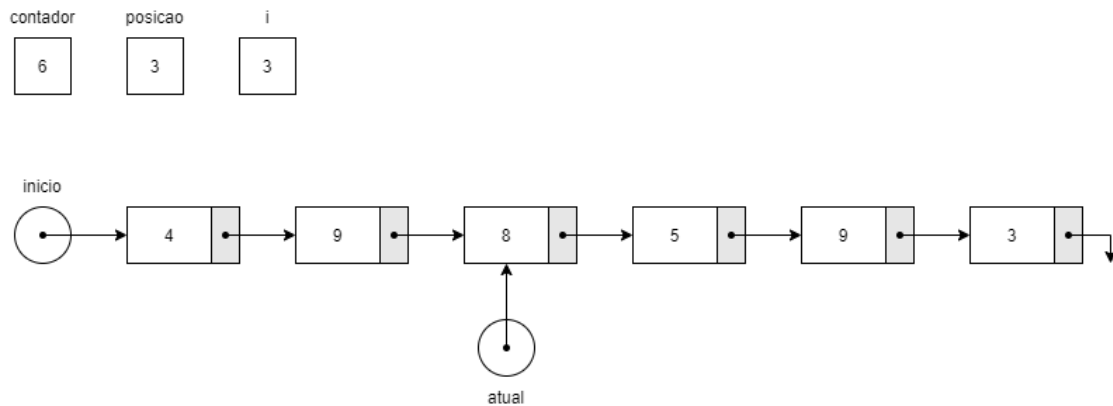
contador	posicao	i
6	3	2



```
atual = atual->proximoElemento;
```



```
for(int i = 1 ; i < posicao ; i++) {
```



```
[24]: #include "listar.cpp"
```

```
[25]: #include "tamanho.cpp"
```

1.2.5 Inserção elemento na fila

```
[26]: bool Lista::inserir(int posicao, int x) {
    PonteiroElemento p, atual;

    if (posicao < 1 || posicao > contador + 1) {
        return false;
    }

    p = new elemento;
    p->valor = x;

    if (posicao == 1) {
        p->proximoElemento = inicio;
        inicio = p;
    }
}
```

```

    } else {
        setaPosicao(posicao - 1, atual);
        p->proximoElemento = atual->proximoElemento;
        atual->proximoElemento = p;
    }
    contador++;
    return true;
}

```

```

[27]: if (! minhaLista.inserir(2, 3))
        cout << "Não inseriu!";
    else
        cout << minhaLista.listar()

```

Não inseriu!

```

[28]: if (! minhaLista.inserir(-1, 3))
        cout << "Não inseriu!";
    else
        cout << minhaLista.listar()

```

Não inseriu!

```

[29]: if (! minhaLista.inserir(0, 3))
        cout << "Não inseriu!";
    else
        cout << minhaLista.listar()

```

Não inseriu!

```

[30]: if (! minhaLista.inserir(1, 3))
        cout << "Não inseriu!";
    else
        cout << minhaLista.listar();

```

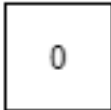
[3]

```


// minhaLista.inserir(1, 3)
PonteiroElemento p, atual;
p = new elemento;
p->valor = x;

```

posicao



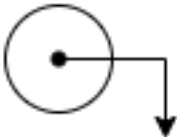
x



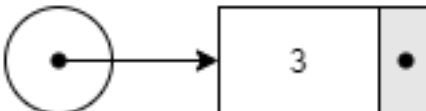
contador



inicio



p

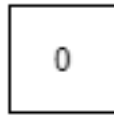


atual



```
if (posicao == 1) {  
    p->proximoElemento = inicio;  
    inicio = p;  
}
```

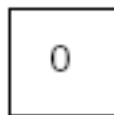
posicao



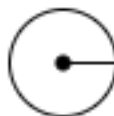
x



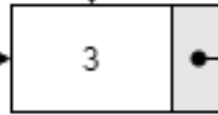
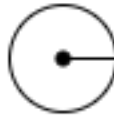
contador



inicio



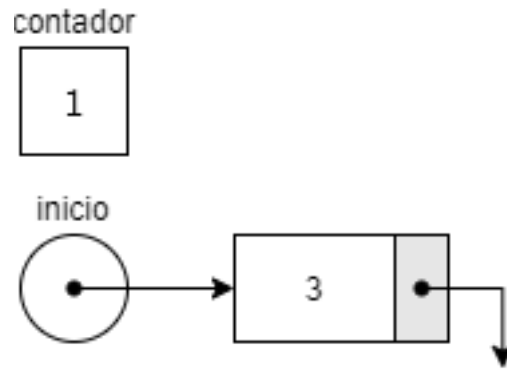
p



atual



```
else {  
    setaPosicao(posicao - 1, atual);  
    p->proximoElemento = atual->proximoElemento;  
    atual->proximoElemento = p;  
}  
contador++;  
return true;
```

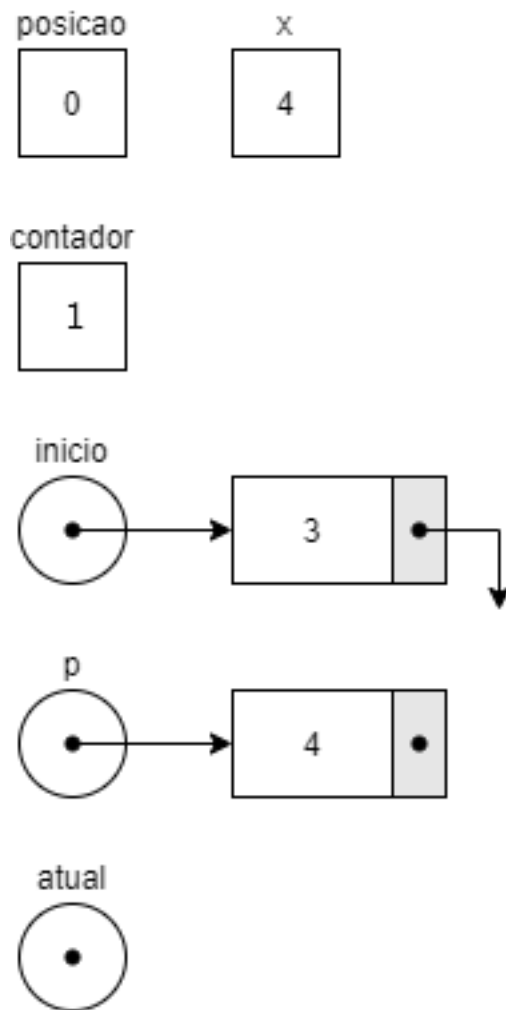
```
[31]: cout << minhaLista.listar();
```

[3]

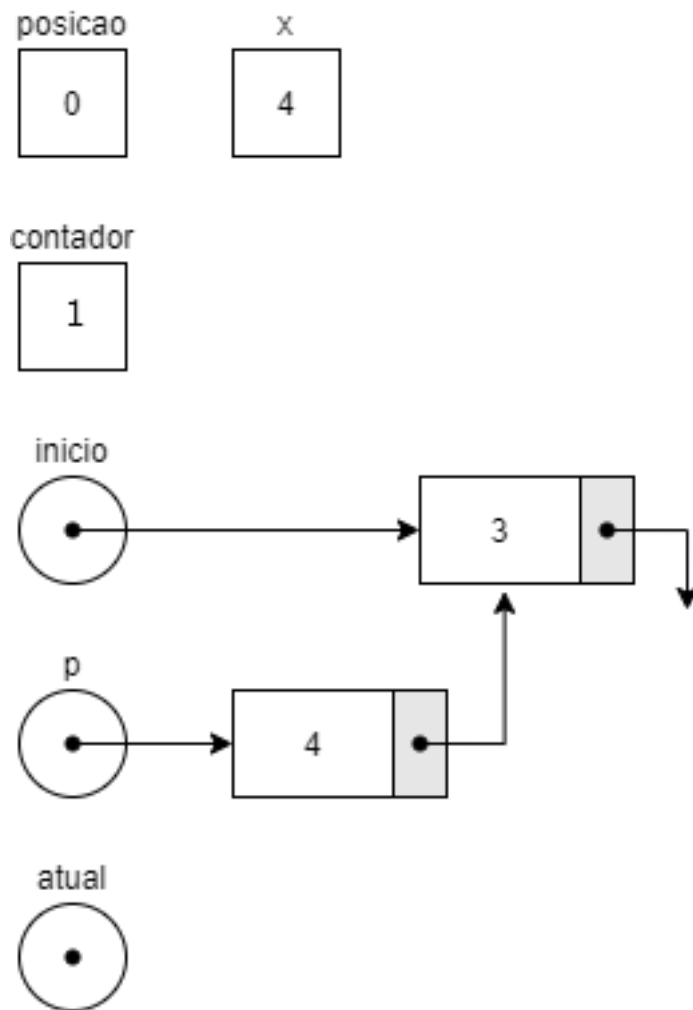
```
[32]: if (! minhaLista.inserir(1, 4))
        cout << "Não inseriu!";
    else
        cout << minhaLista.listar();
```

[4] [3]

```
PonteiroElemento p, atual;
p = new elemento;
p->valor = x;
```



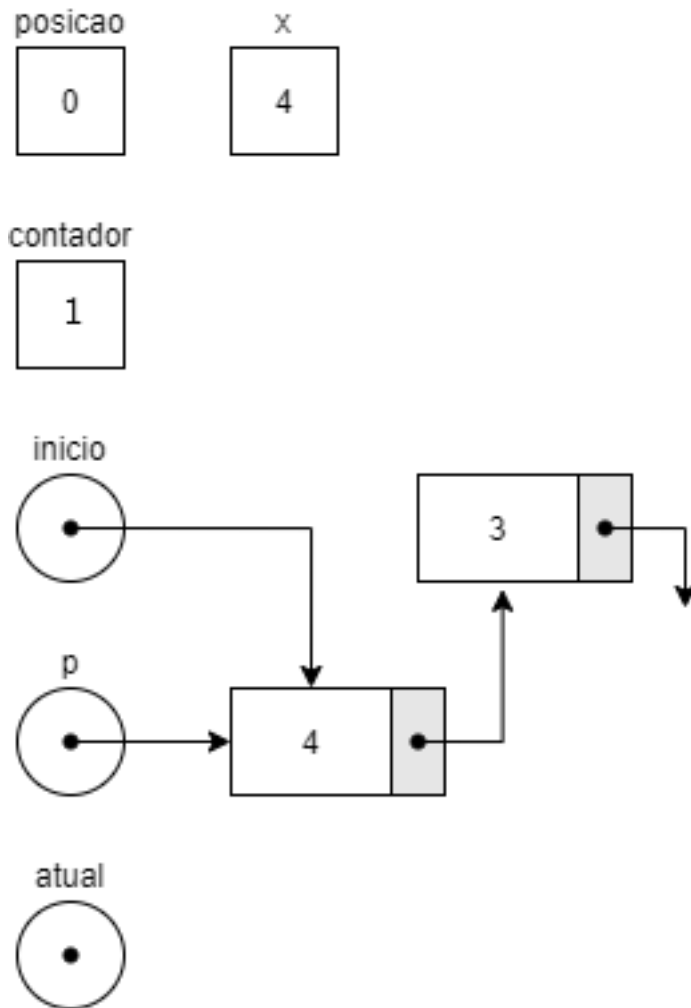
```
if (posicao == 1) {  
    p->proximoElemento = inicio;  
}
```



```

    inicio = p;
}

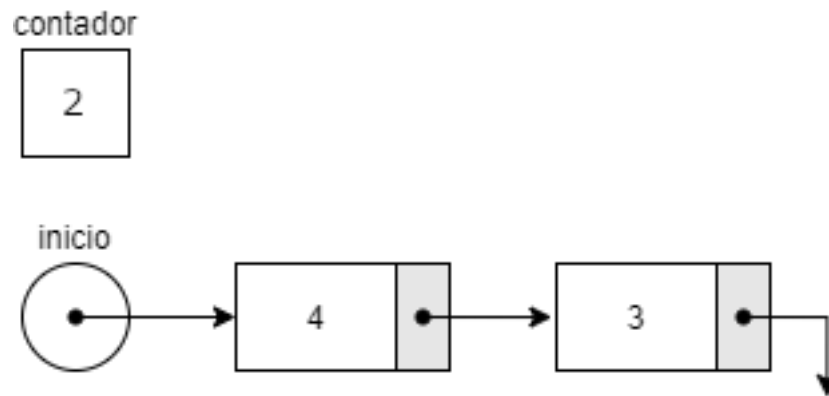
```



```

else {
    setaPosicao(posicao - 1, atual);
    p->proximoElemento = atual->proximoElemento;
    atual->proximoElemento = p;
}
contador++;
return true;

```



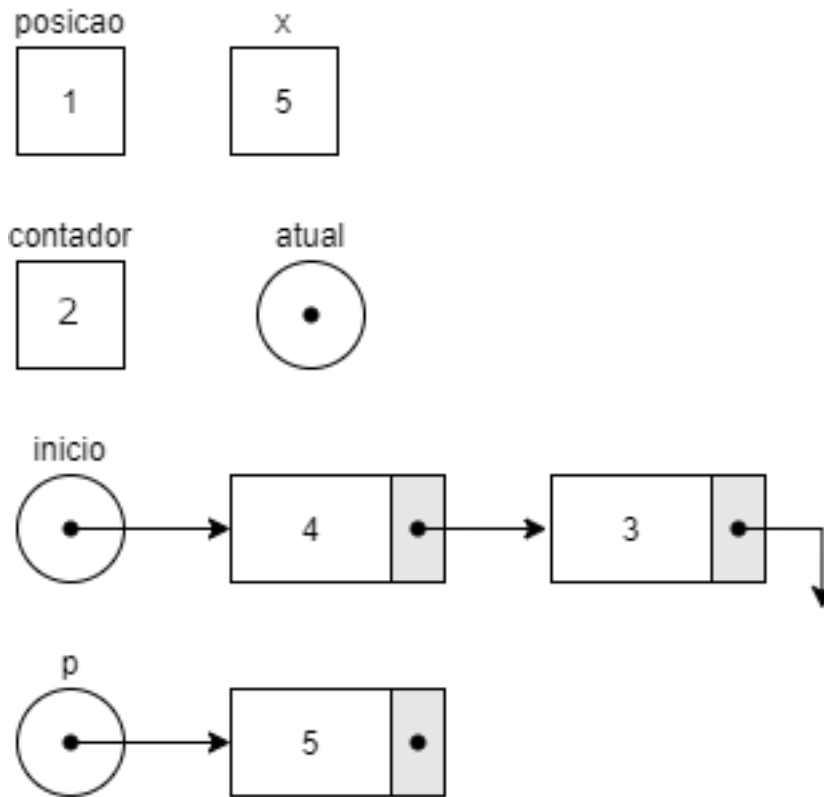
```
[33]: cout << minhaLista.listar();
```

[4] [3]

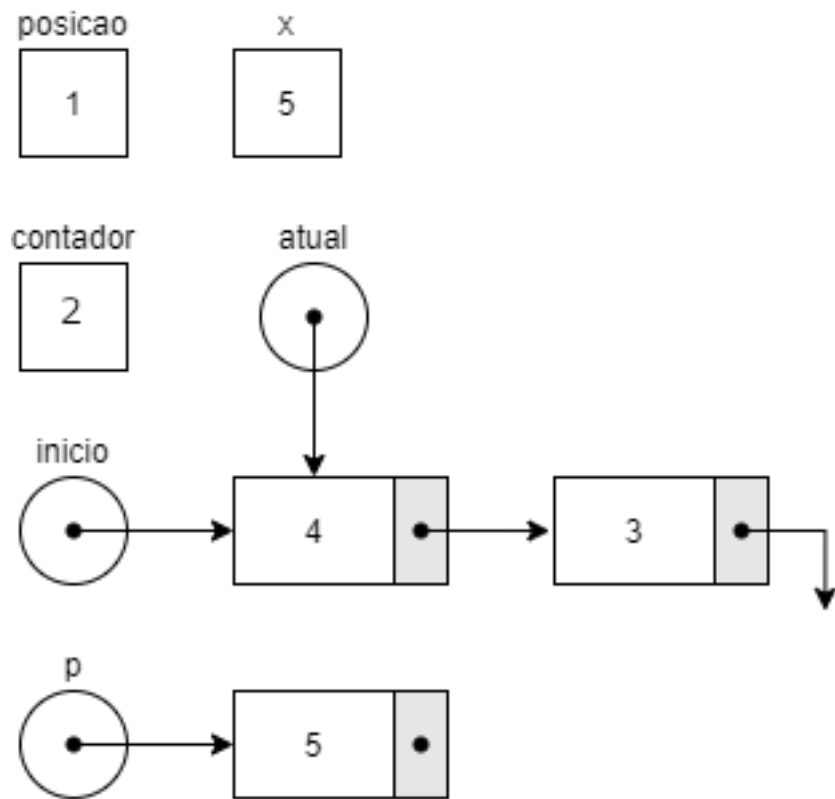
```
[35]: if (! minhaLista.inserir(2, 5))
        cout << "Não inseriu!";
    else
        cout << minhaLista.listar();
```

[4] [5] [3]

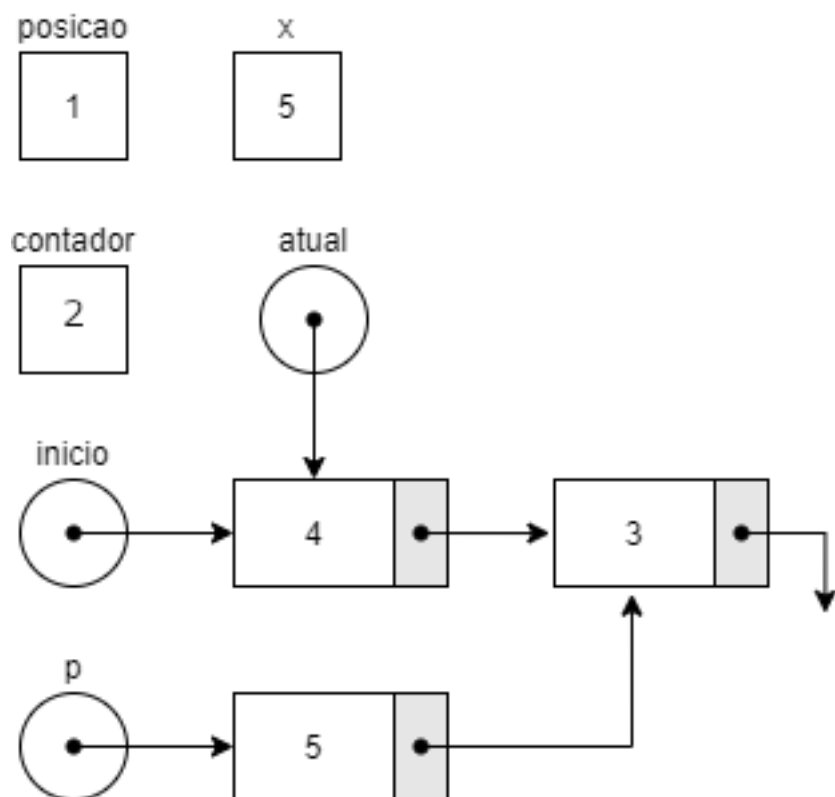
```
PonteiroElemento p, atual;
p = new elemento;
p->valor = x;
```



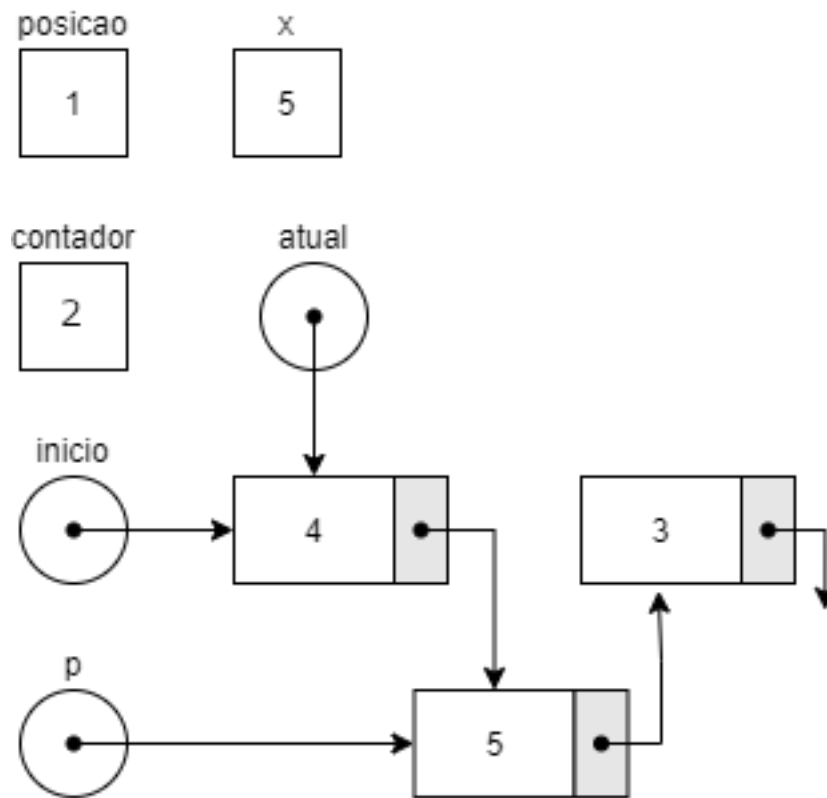
```
if (posicao == 1) {  
    p->proximoElemento = inicio;  
    inicio = p;  
}  
else {  
    setaPosicao(posicao - 1, atual);  
}
```



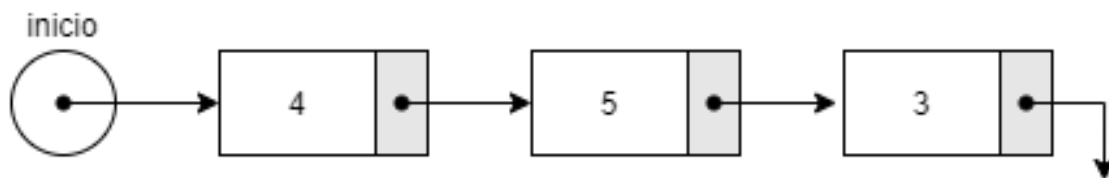
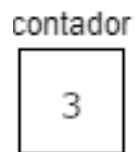
`p->proximoElemento = atual->proximoElemento;`



```
atual->proximoElemento = p;
```



```
}  
contador++;  
return true;
```



```
[34]: cout << minhaLista.listar();
```

[4] [3]


```
[36]: if (! minhaLista.inserir(1, 7))
      cout << "Não inseriu!";
      else
      cout << minhaLista.listar()
```

[7] [4] [5] [3]

```
[37]: if (! minhaLista.inserir(3, 9))
      cout << "Não inseriu!";
      else
      cout << minhaLista.listar()
```

[7] [4] [9] [5] [3]

```
[38]: if (! minhaLista.inserir(4, 8))
      cout << "Não inseriu!";
      else
      cout << minhaLista.listar()
```

[7] [4] [9] [8] [5] [3]

```
[39]: if (! minhaLista.inserir(6, 9))
      cout << "Não inseriu!";
      else
      cout << minhaLista.listar()
```

[7] [4] [9] [8] [5] [9] [3]

1.2.6 Remoção de Elemento da Fila

```
[40]: bool Lista::remover(int posicao, int &x) {
      PonteiroElemento p, atual;

      if (posicao < 1 or posicao > contador) {
          return false;
      }

      if(posicao == 1) {
          p = inicio;
          inicio = inicio->proximoElemento;
      } else {
          setaPosicao(posicao - 1, atual);
          p = atual->proximoElemento;
          atual->proximoElemento = p->proximoElemento;
      }
      x = p->valor;
      delete p;
      contador --;
```

```
    return true;
}
```

```
[41]: int y;
```

```
[42]: cout << minhaLista.listar();
```

[7] [4] [9] [8] [5] [9] [3]

```
[43]: if (! minhaLista.remover(-1, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

Não removeu!

```
[44]: if (! minhaLista.remover(8, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

Não removeu!

```
[45]: if (! minhaLista.remover(0, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

Não removeu!

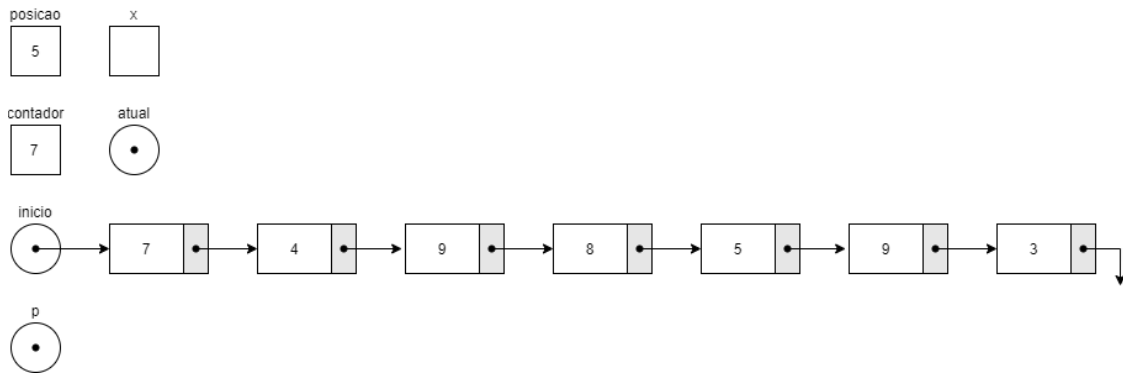
```
[46]: cout << minhaLista.listar();
```

[7] [4] [9] [8] [5] [9] [3]

```
[47]: if (! minhaLista.remover(1, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

y: 7 - [4] [9] [8] [5] [9] [3]

```
// minhaLista.remover(1, y)
PonteiroElemento p, atual;
```



```

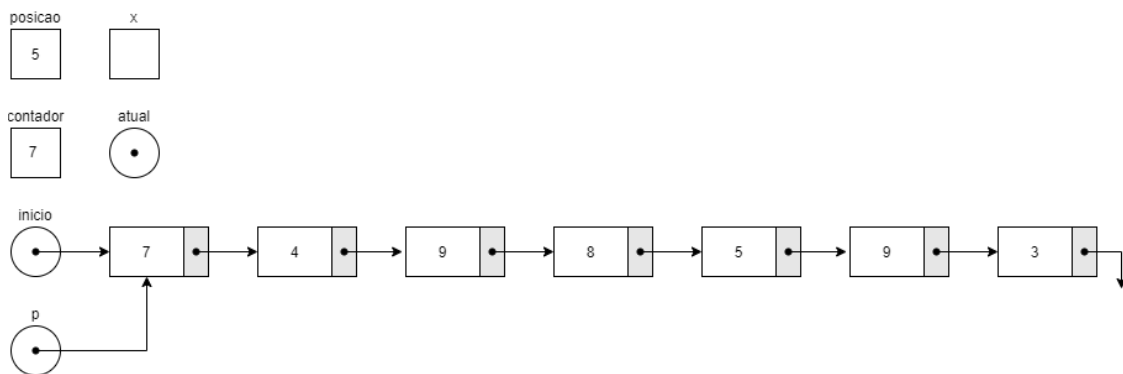
if (posicao < 1 or posicao > contador) {
    return false;
}

```

```

if(posicao == 1) {
    p = inicio;

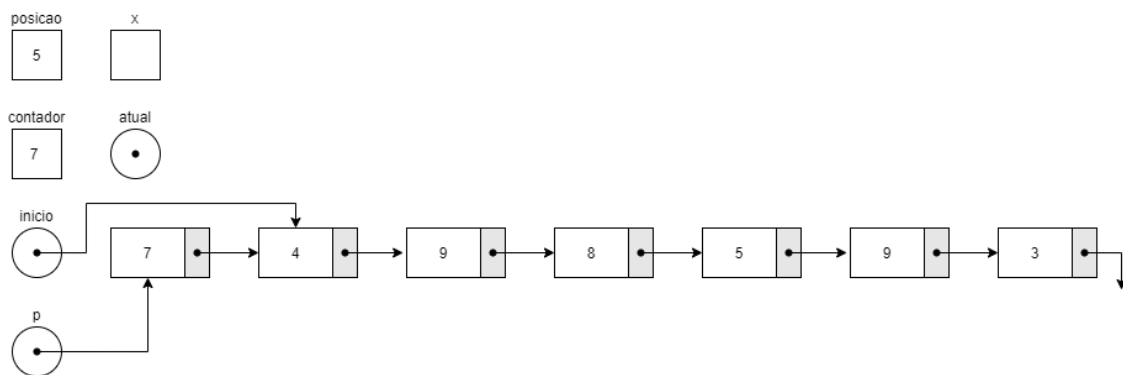
```



```

    inicio = inicio->proximoElemento;

```

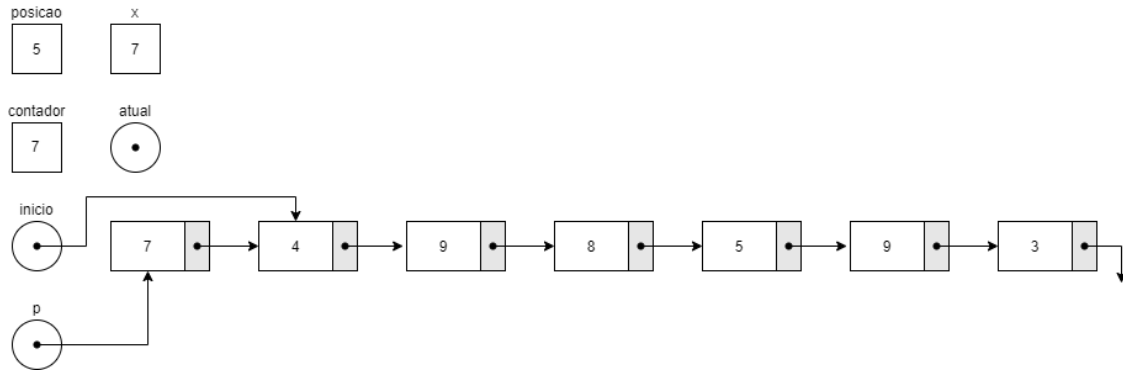


```

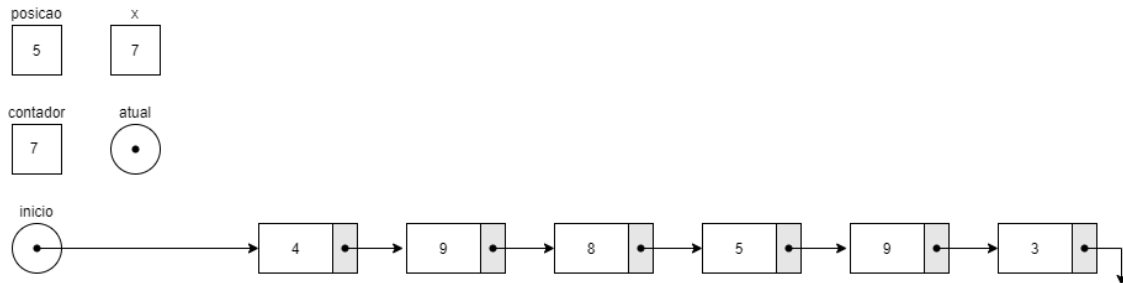
} else {
    setaPosicao(posicao - 1, atual);
    p = atual->proximoElemento;
    atual->proximoElemento = p->proximoElemento;
}

```

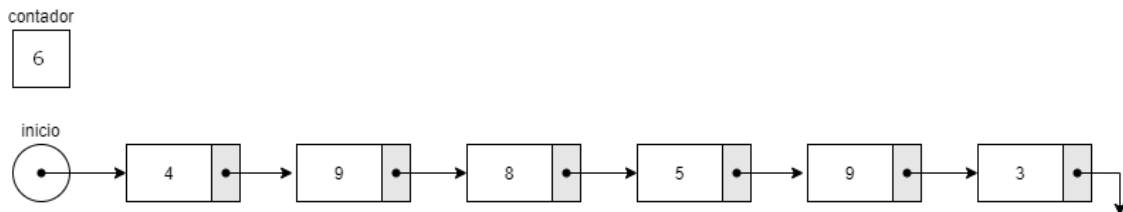
```
x = p->valor;
```



```
delete p;
```



```
contador --;  
return true;
```



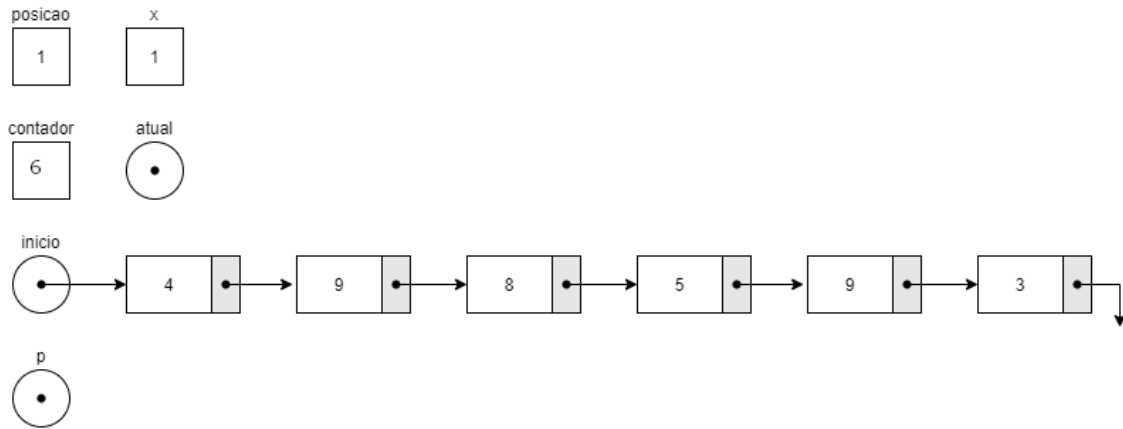
```
[48]: cout << minhaLista.listar();
```

```
[4] [9] [8] [5] [9] [3]
```

```
[49]: if (! minhaLista.remover(5, y))  
    cout << "Não removeu!";  
else  
    cout << "y: " << y << " - " << minhaLista.listar()
```

```
y: 9 - [4] [9] [8] [5] [3]
```

```
// minhaLista.remover(5, y)  
PonteiroElemento p, atual;
```

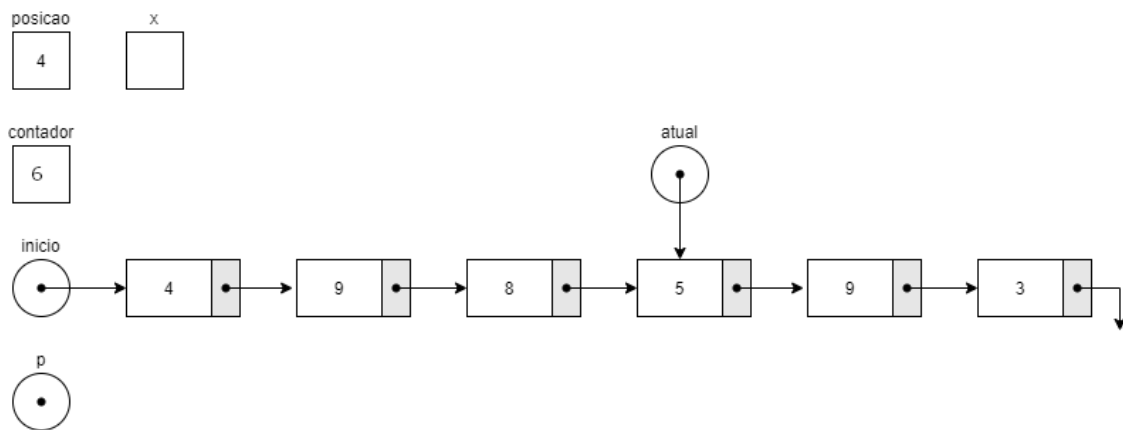


```

if (posicao < 1 or posicao > contador) {
    return false;
}

if(posicao == 1) {
    p = inicio;
    inicio = inicio->proximoElemento;
} else {
    setaPosicao(posicao - 1, atual);

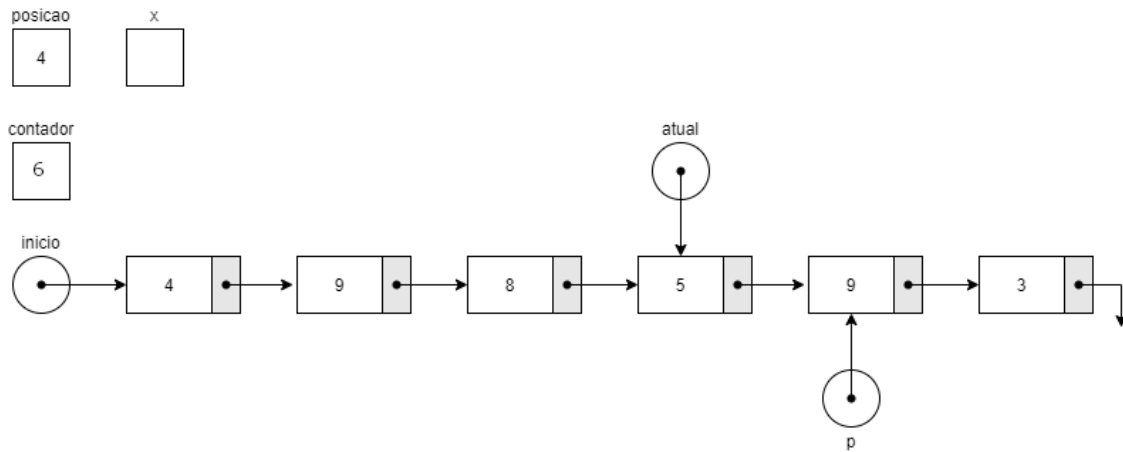
```



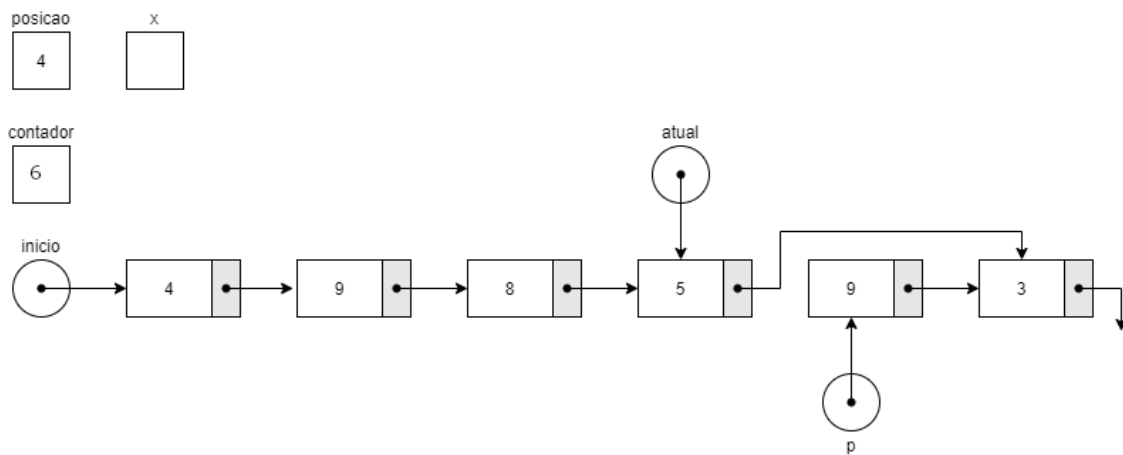
```

p = atual->proximoElemento;

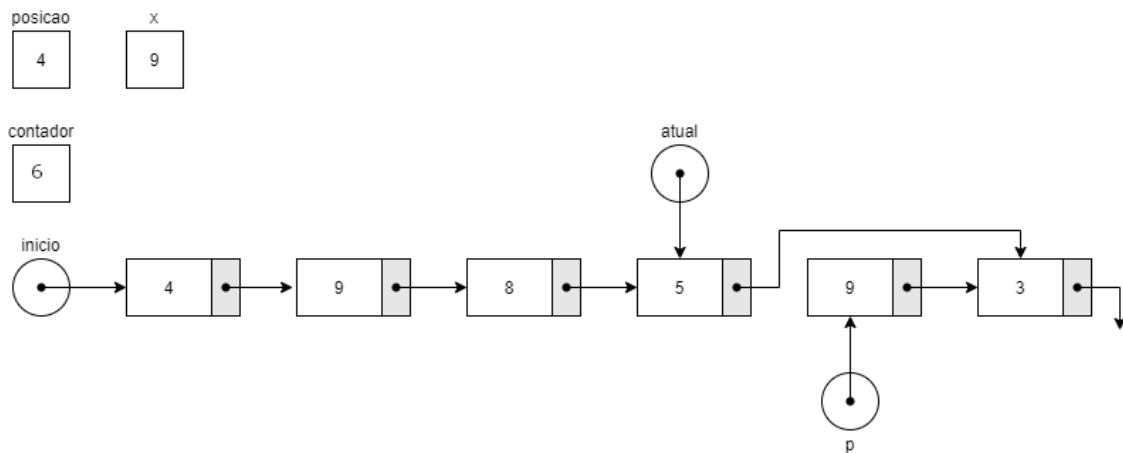
```



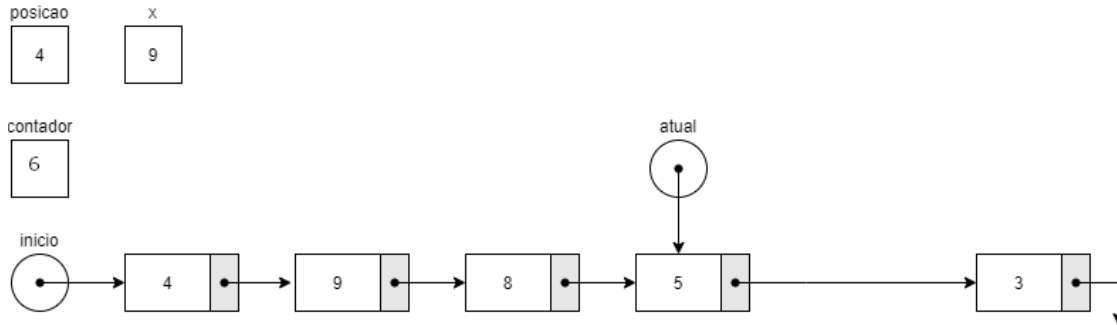
```
atual->proximoElemento = p->proximoElemento;
```



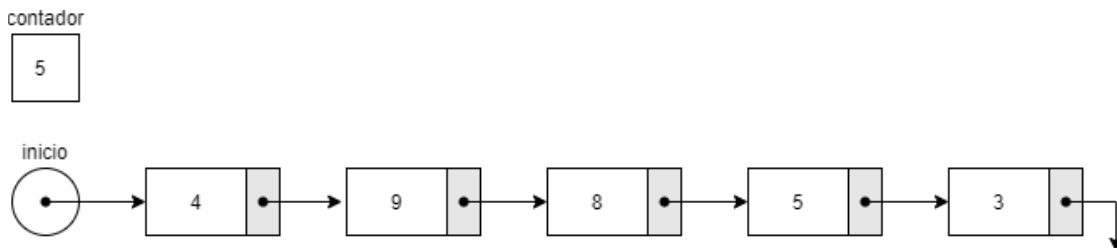
```
}
x = p->valor;
```



```
delete p;
```



```
contador --;
return true;
```



```
[50]: cout << minhaLista.listar();
```

```
[4] [9] [8] [5] [3]
```

```
[51]: if (! minhaLista.remover(2, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

```
y: 9 - [4] [8] [5] [3]
```

```
[52]: if (! minhaLista.remover(1, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

```
y: 4 - [8] [5] [3]
```

```
[53]: if (! minhaLista.remover(1, y))
        cout << "Não removeu!";
    else
        cout << "y: " << y << " - " << minhaLista.listar()
```

```
y: 8 - [5] [3]
```

```
[54]: if (! minhaLista.remover(1, y))
      cout << "Não removeu!";
      else
      cout << "y: " << y << " - " << minhaLista.listar();
```

y: 5 - [3]

```
[55]: if (! minhaLista.remover(1, y))
      cout << "Não removeu!";
      else
      cout << "y: " << y << " - " << minhaLista.listar();
```

y: 3 -

```
[56]: if (! minhaLista.remover(1, y))
      cout << "Não removeu!";
      else
      cout << "y: " << y << " - " << minhaLista.listar();
```

Não removeu!

```
[57]: cout << minhaLista.tamanho();
```

0

[]:

[]:

```
[ ]: // listar
      // começando em 0
      for (int i = 0 ; i < contador ; i++)

      // começando em 1
      for (int i = 1 ; i <= contador ; i++)

      // for reverso
      for (int i = 2 ; i >= 1 ; i--)
```