02_fila_dinamica

September 17, 2021

1 Filas

1.1 Implementação Fila Dinâmica

[1]:
```
#include <iostream>
using namespace std;
```

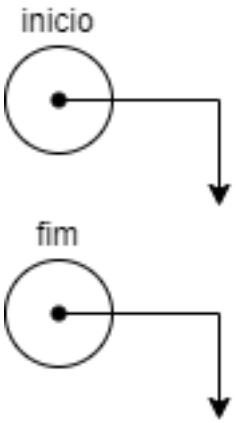
1.1.1 Definição da Classe

[2]:
```
class Fila {
    private:
        struct elemento {
            int valor;
            elemento *proximoElemento; // ligação próximo nó
        };
        typedef elemento *PonteiroElemento;
        PonteiroElemento inicio;
        PonteiroElemento fim;
    public:
        Fila();
        bool vazia();
        bool cheia();
        bool inserir(int x);
        bool remover(int &x);
};
```

1.1.2 Método Construtor

[3]:
```
Fila::Fila() {
    inicio = nullptr;
    fim = nullptr;
}
```

[4]:
```
Fila minhaFila;
```



1.1.3 Verifica se fila está vazia

[5]:
```
bool Fila::vazia() {
    return inicio == nullptr;
}
```

[6]:
```
if (minhaFila.vazia()) {
    cout << "Está vazia!";
}
```

```
Está vazia!
```

1.1.4 Verifica se fila está cheia

[7]:
```
bool Fila::cheia() {
    return false;
}
```

[8]:
```
if (! minhaFila.cheia()) {
    cout << "Não está cheia!";
}
```

```
Não está cheia!
```

1.1.5 Inserção elemento na fila

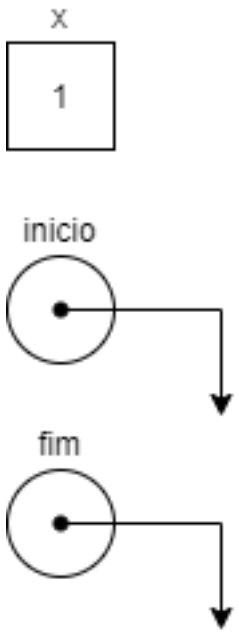
[9]:
```
bool Fila::inserir(int x) {
    PonteiroElemento p;
    p = new elemento;
    if (p == nullptr) {
        return 0;
    }
    p->valor = x;
    if (vazia()) {
```

```
        inicio = p;
        fim = p;
    } else {
        fim->proximoElemento = p;
        fim = p;
    }
    p->proximoElemento = nullptr;
    return true;
}
```

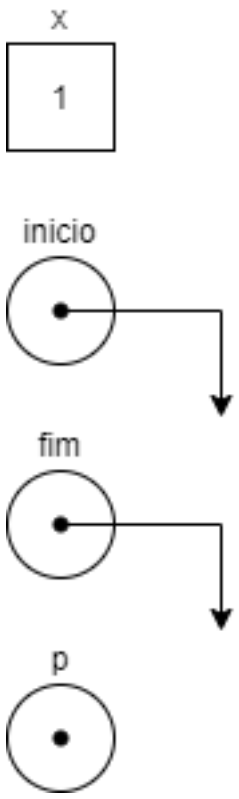
[10]:
```
minhaFila.inserir(1)
```

[10]: true

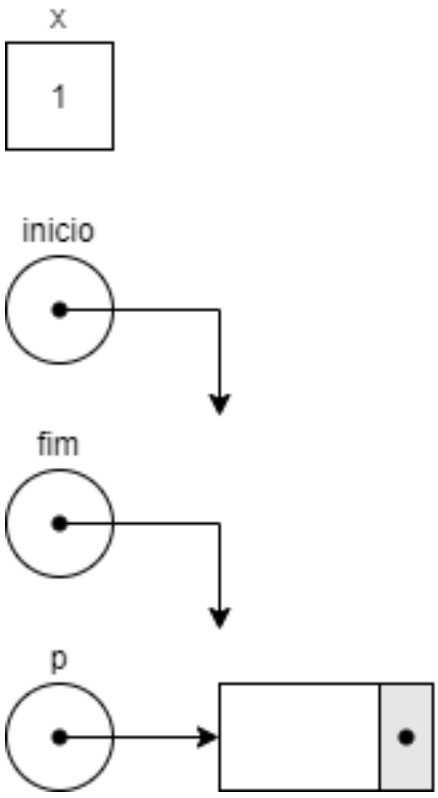
minhaFila.inserir(1)



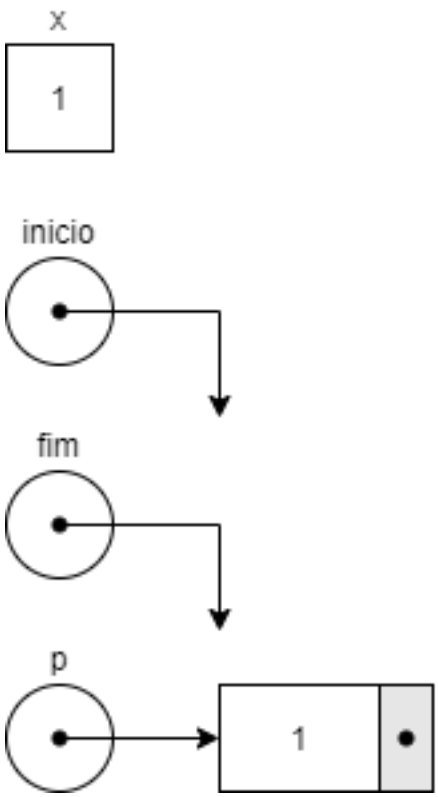
PonteiroElemento p;



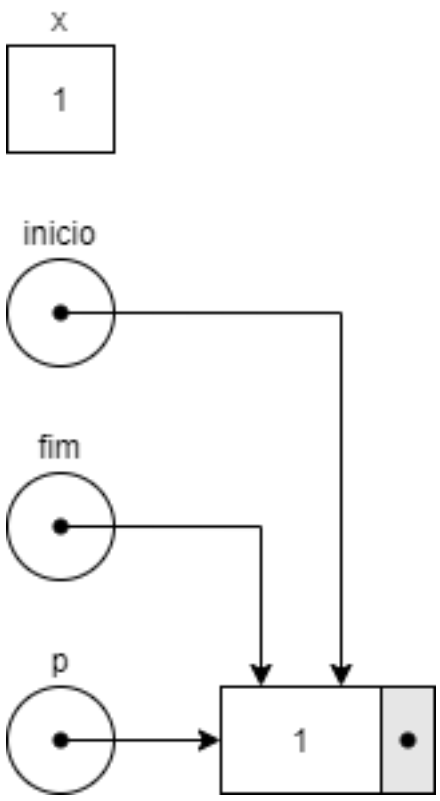

```
p = new elemento;
```



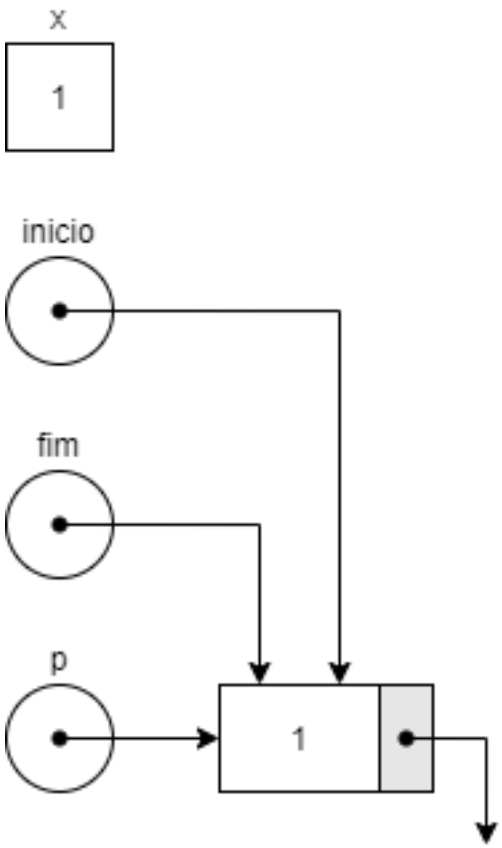
```
p->valor = x;
```



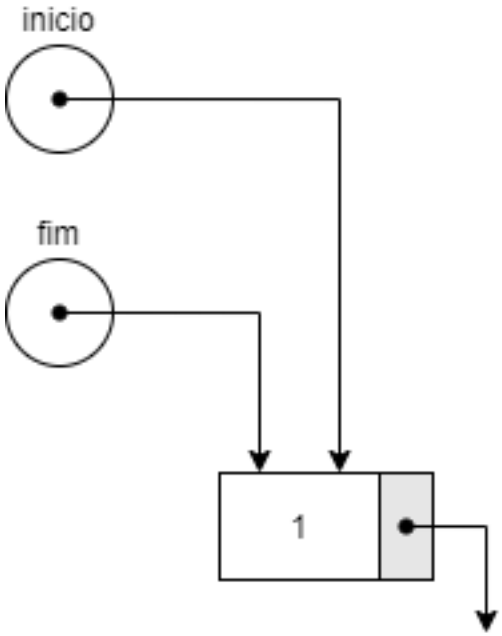

```
if (vazia()) {
    inicio = p;
    fim = p;
} else {
    fim->proximoElemento = p;
    fim = p;
}
```




```
p->proximoElemento = nullptr;
```



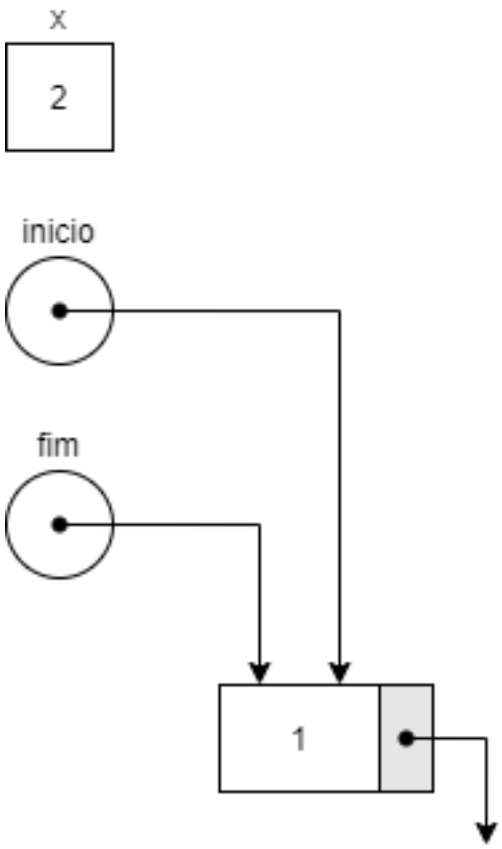
Finalizado o método:



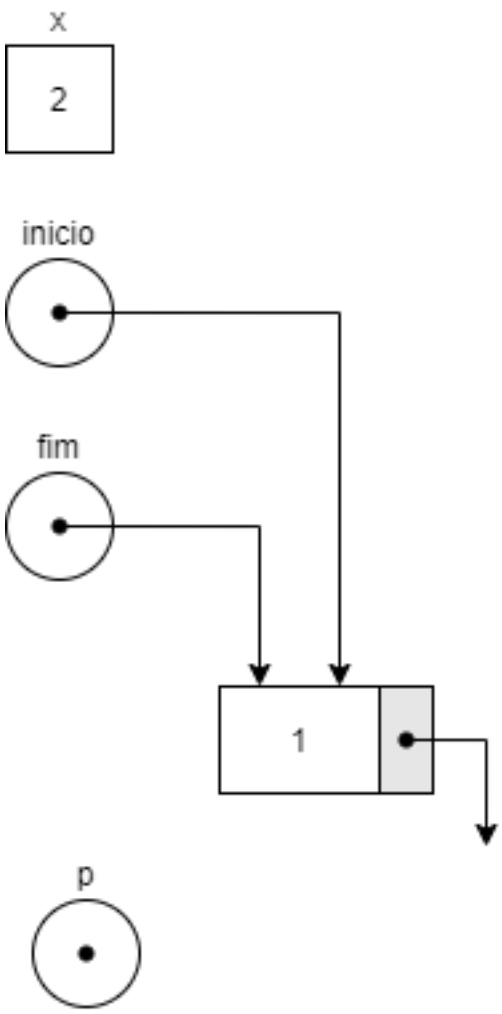
[11]:
```
minhaFila.inserir(2)
```

[11]: true

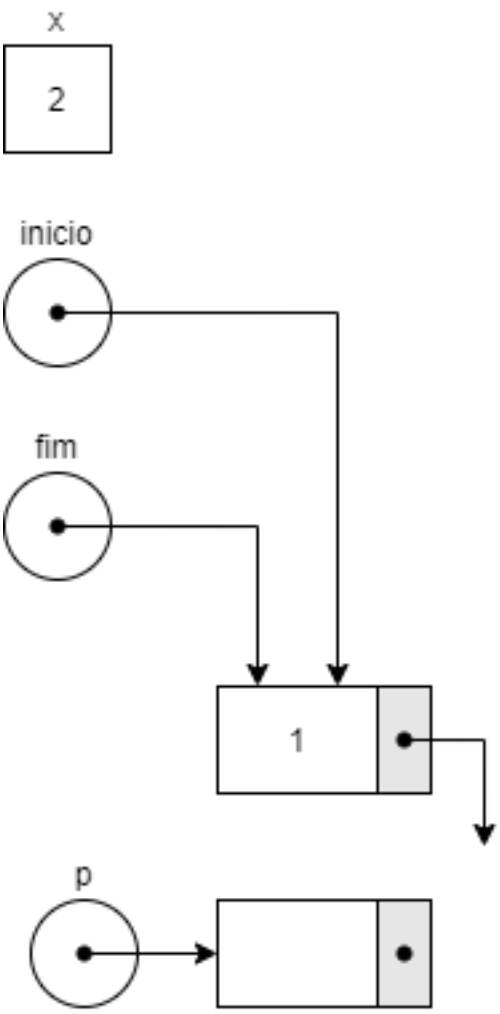
```
minhaFila.inserir(2)
```



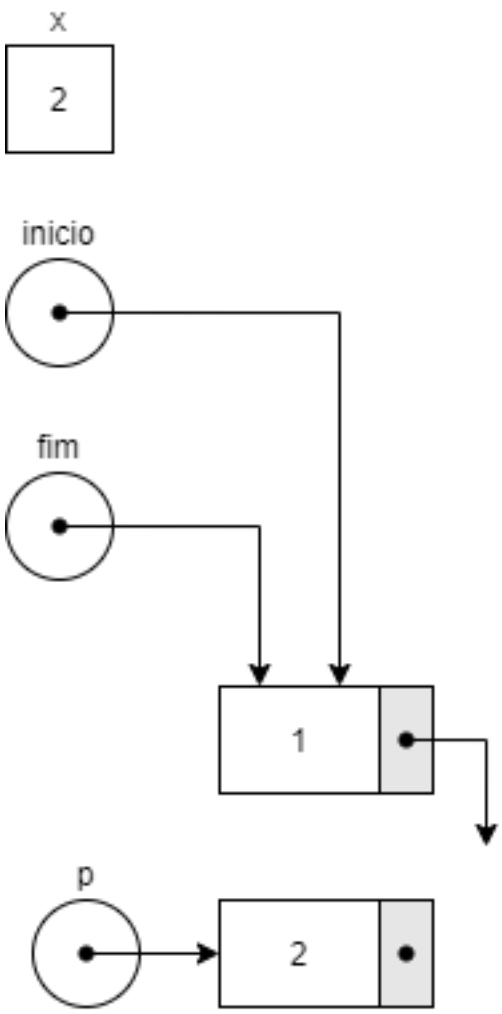

```
PonteiroElemento p;
```



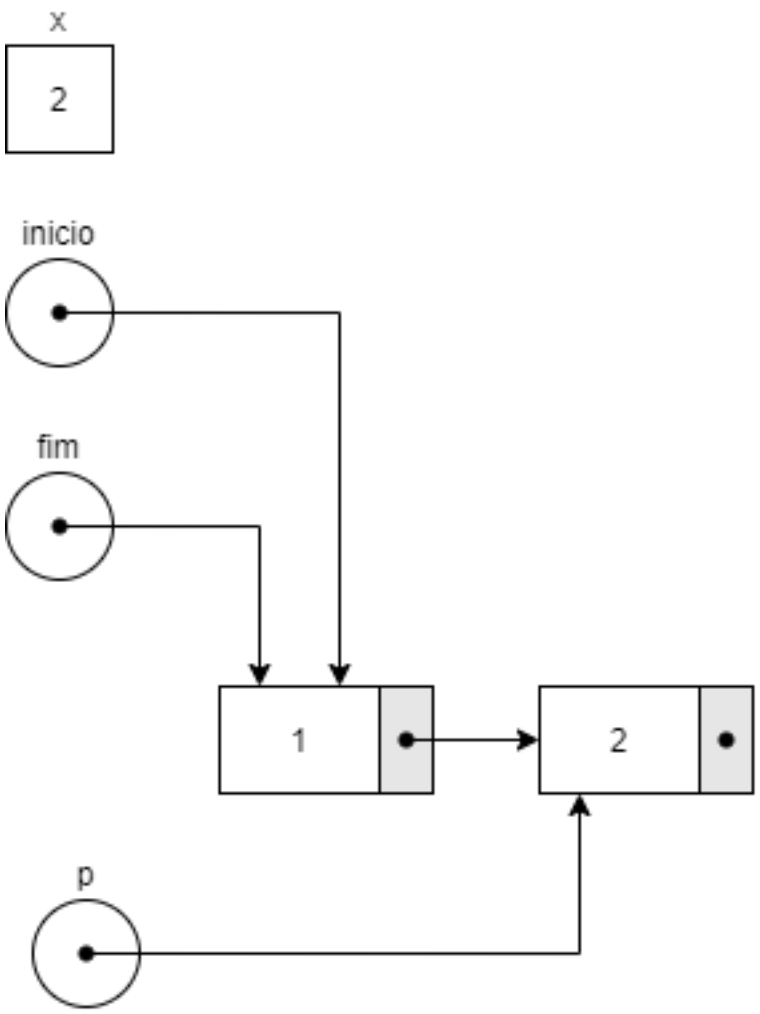

```
p = new elemento;
```



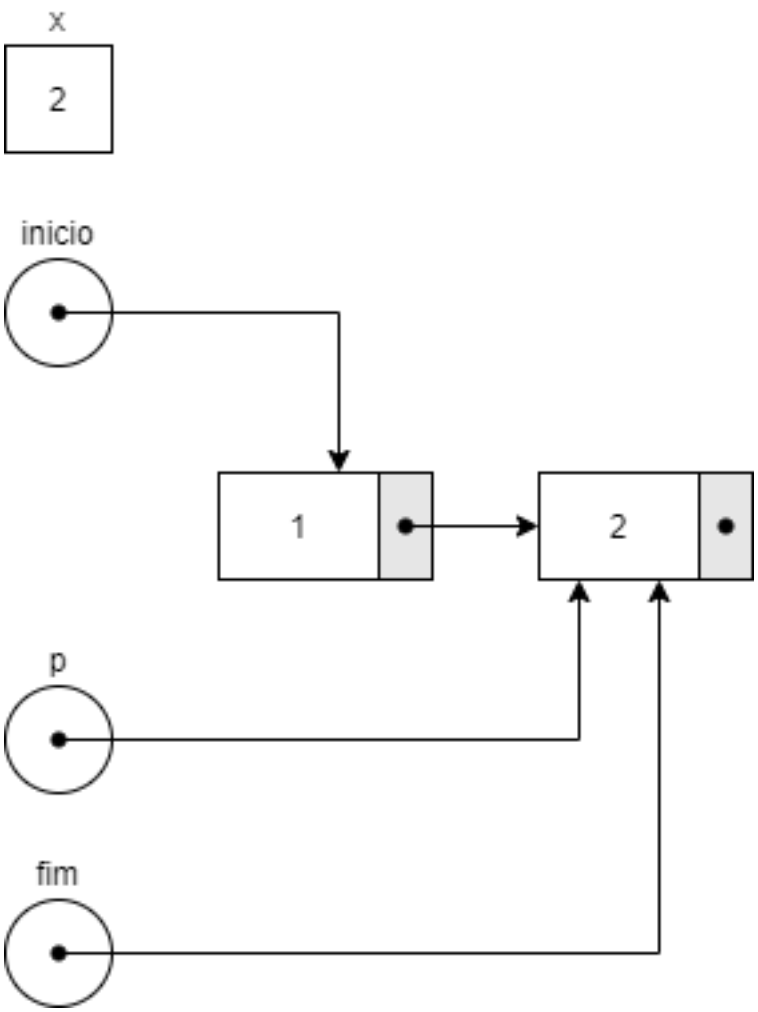

```
p->valor = x;
```



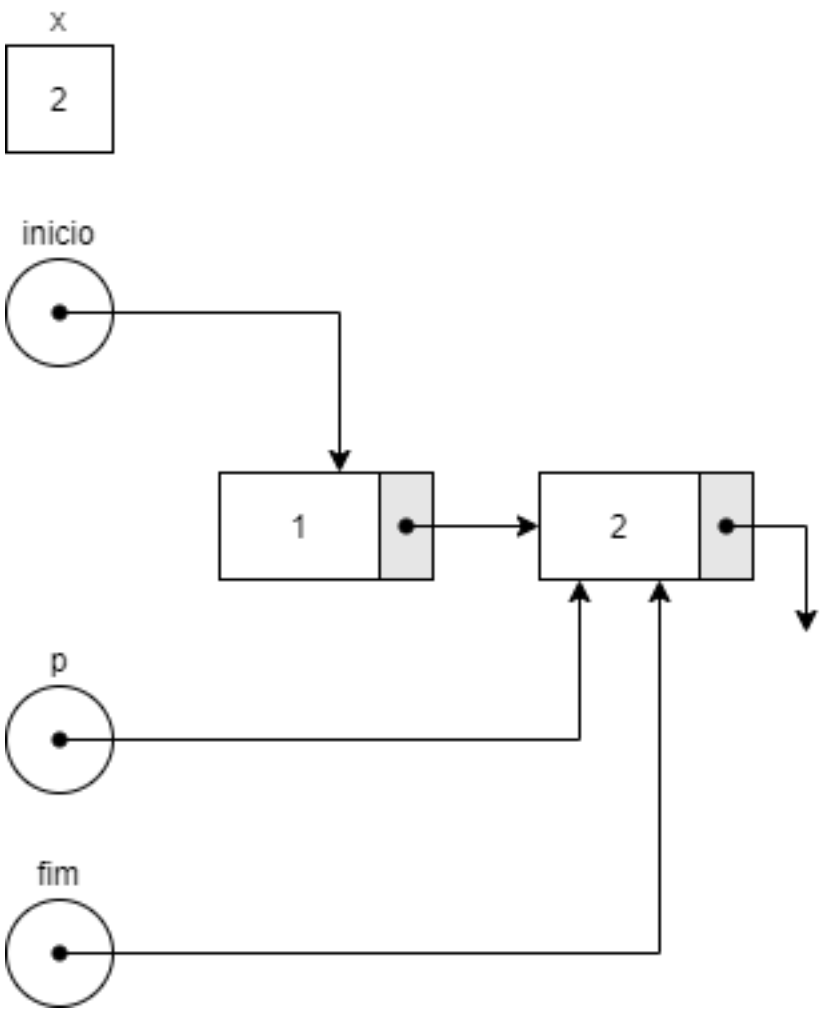

```
if (vazia()) {
    inicio = p;
    fim = p;
} else {
    fim->proximoElemento = p;
```



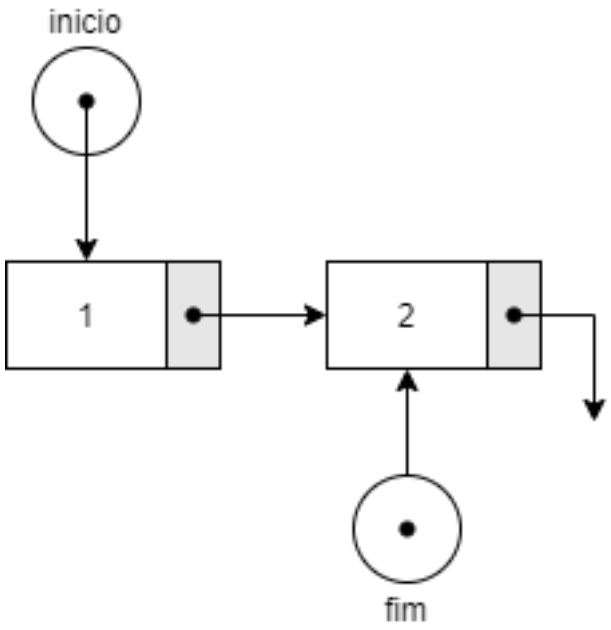

```
    fim = p;
}
```




```
p->proximoElemento = nullptr;
```

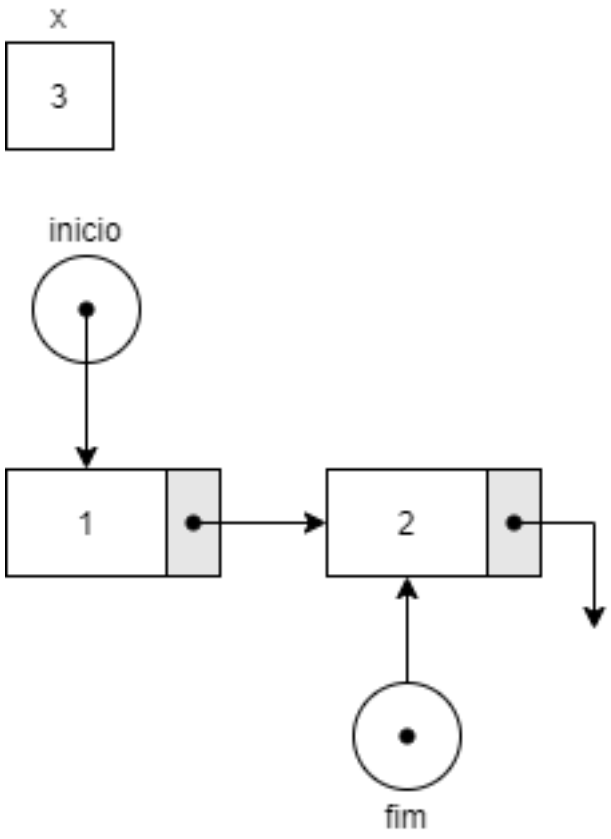


Finalizado o método:

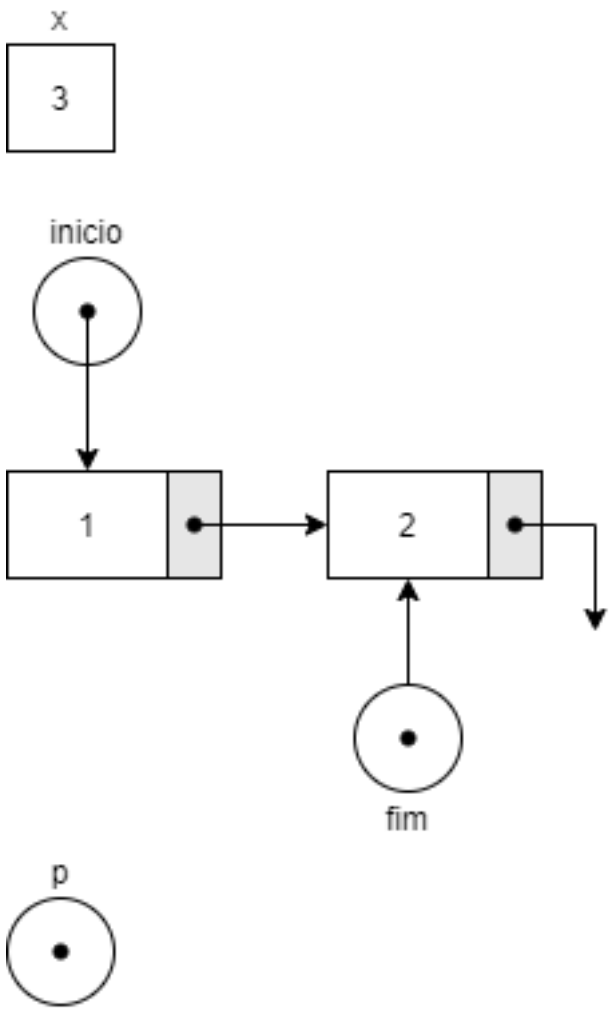


[ ]:
```
minhaFila.inserir(3)
```

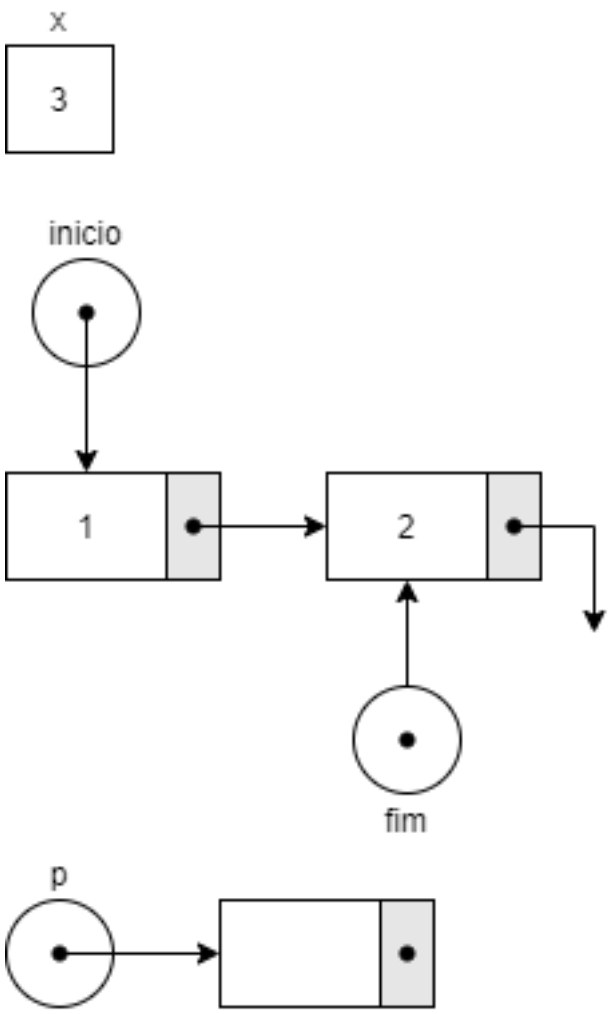
```
minhaFila.inserir(3)
```



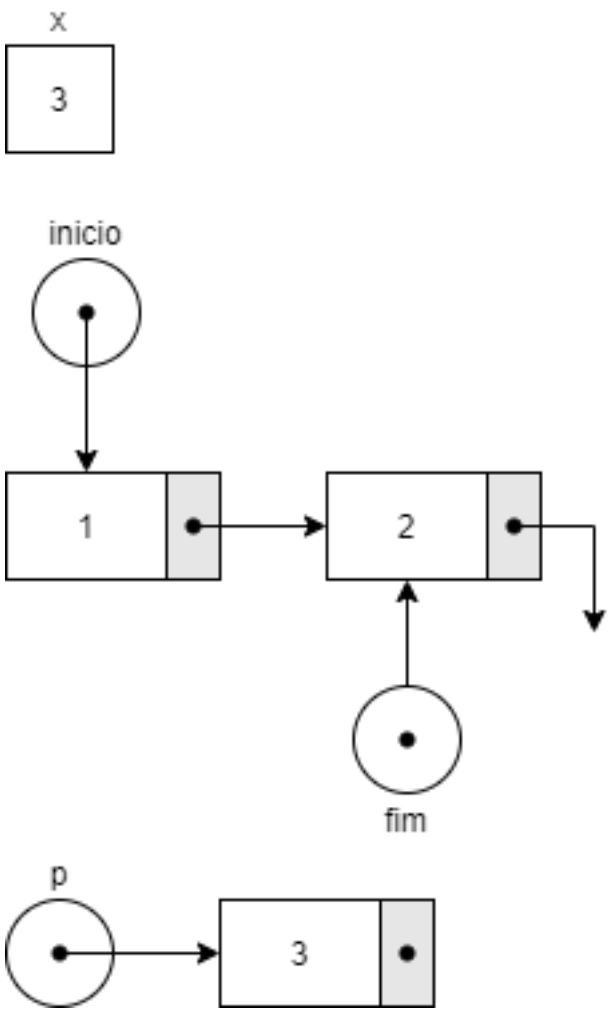
```
PonteiroElemento p;
```



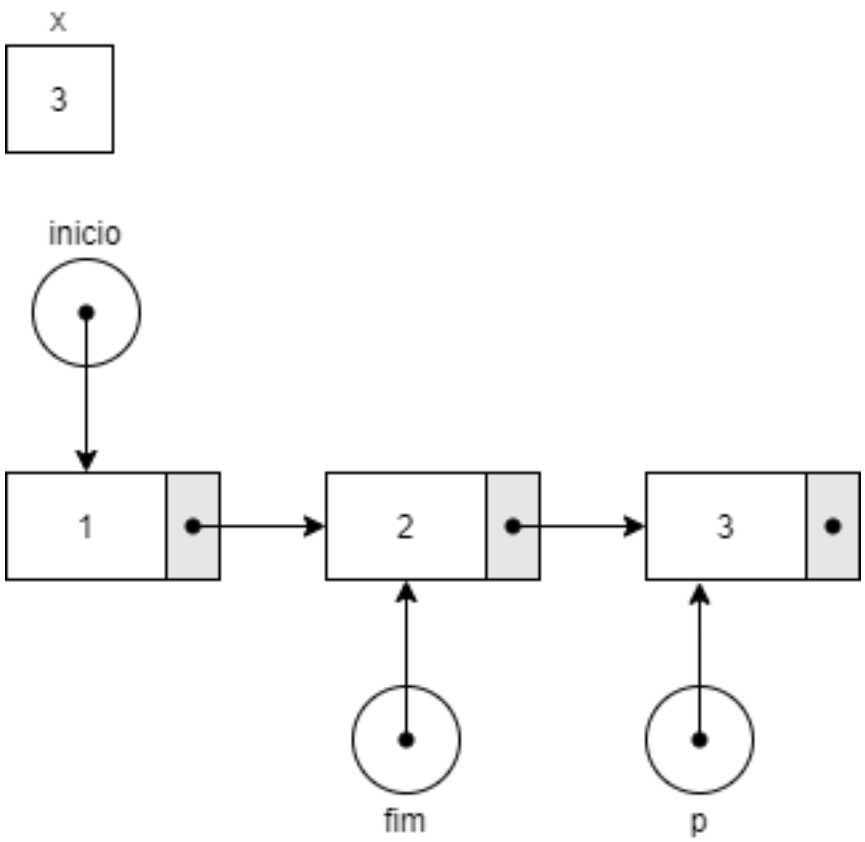

```
p = new elemento;
```



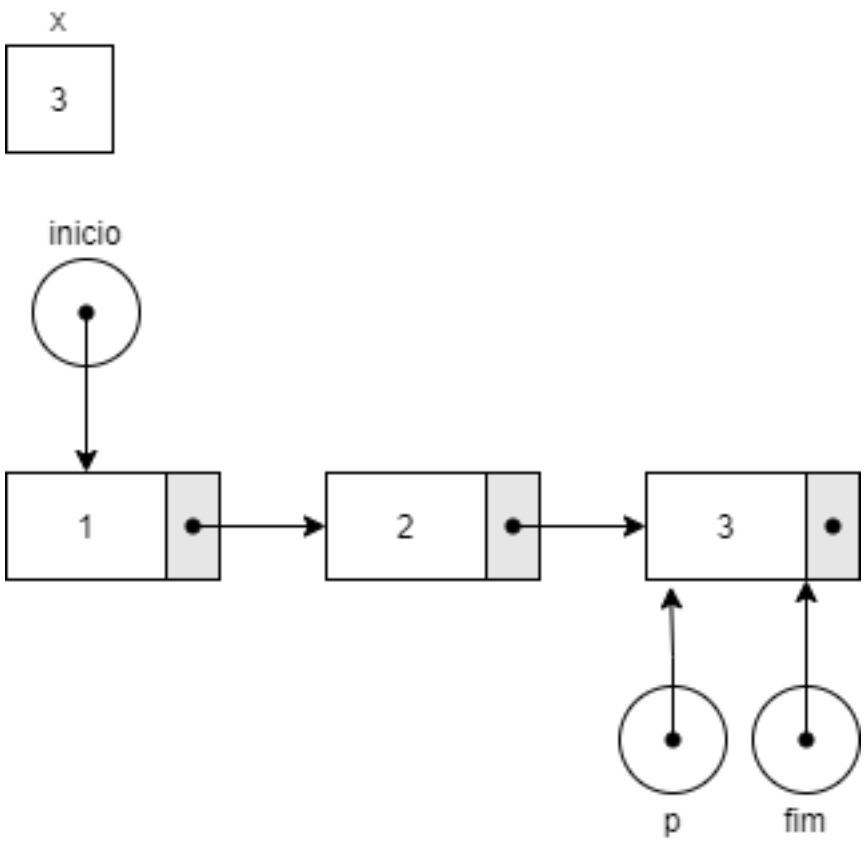

```
p->valor = x;
```



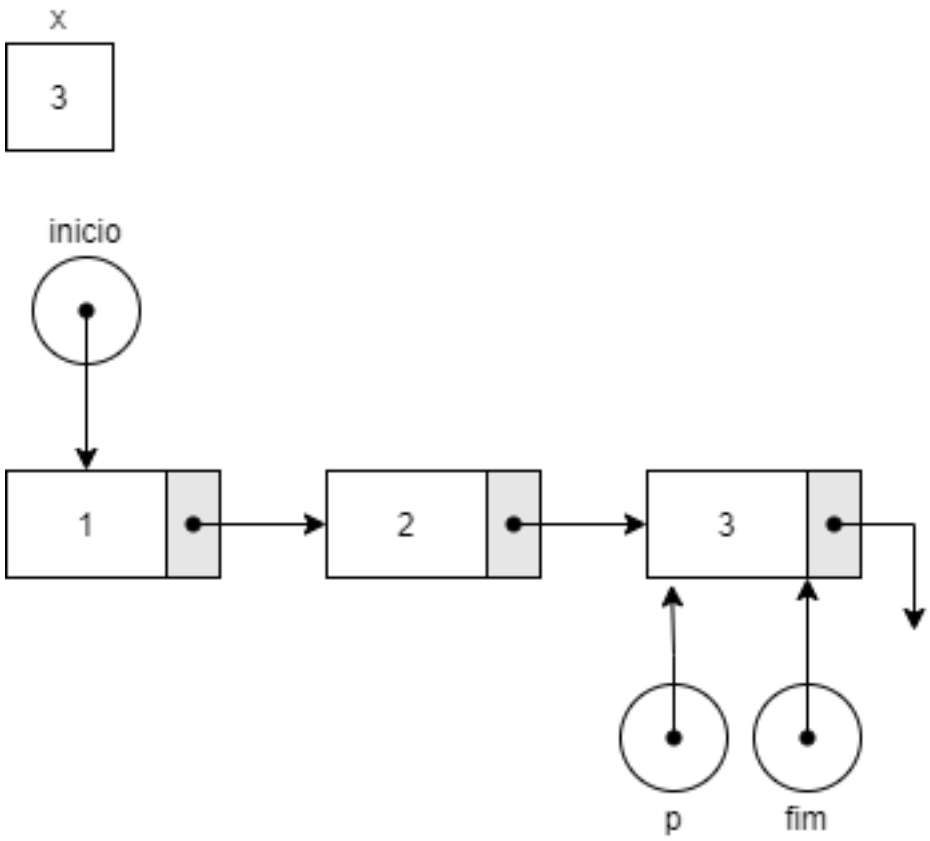

```
if (vazia()) {
    inicio = p;
    fim = p;
} else {
    fim->proximoElemento = p;
```



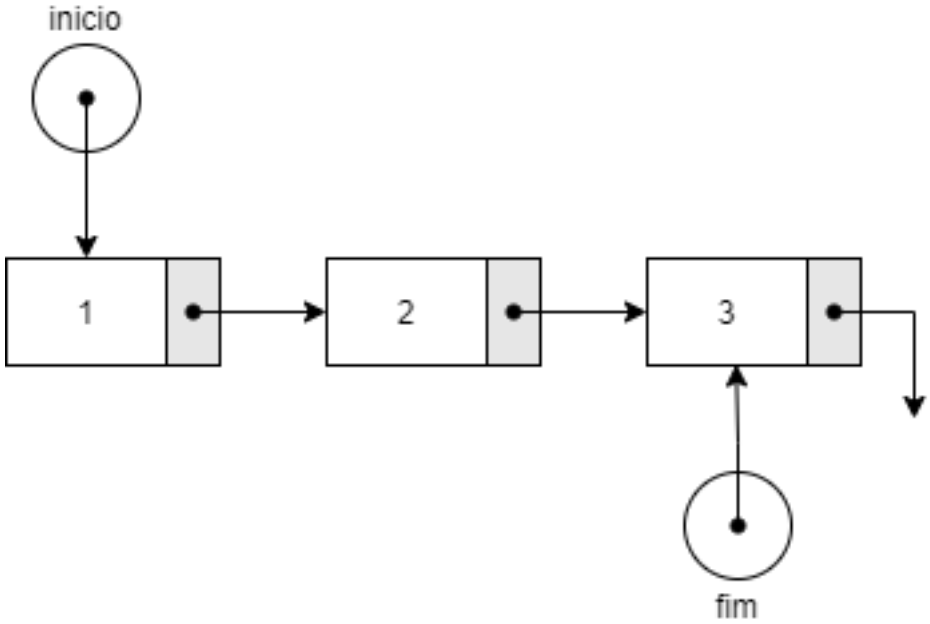

```
    fim = p;
}
```




```
p->proximoElemento = nullptr;
```



Finalizado o método:



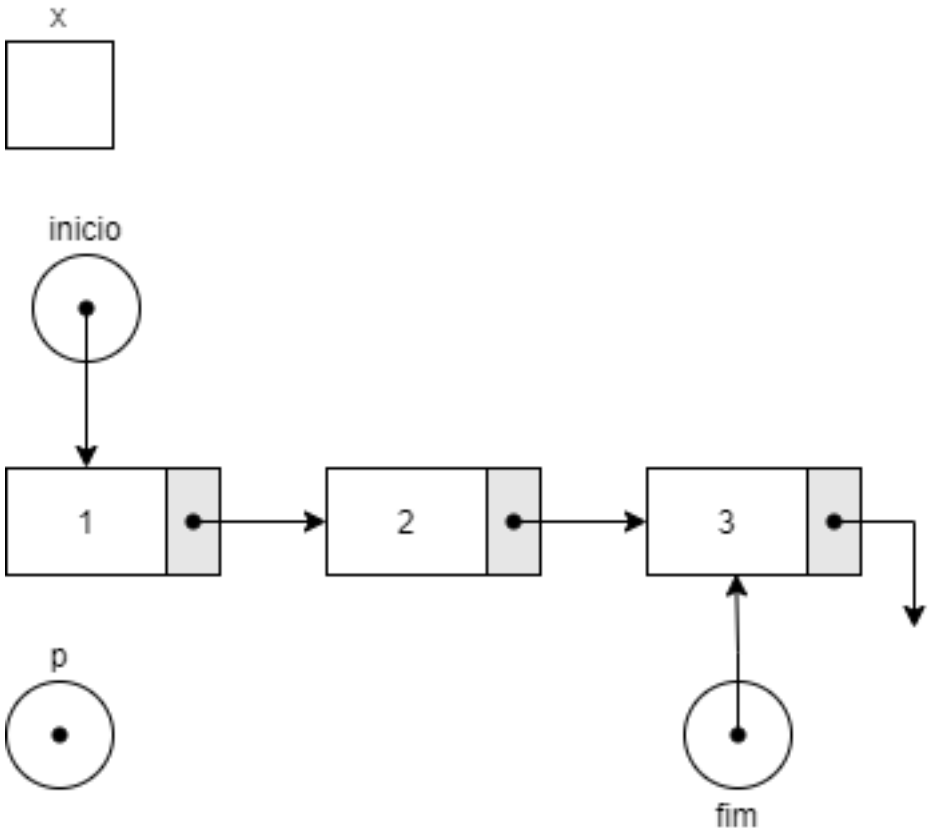
1.1.6 Remoção de Elemento da Fila

[13]:
```
bool Fila::remover(int &x) {
    PonteiroElemento p;
    if (vazia())
        return false;
    x = inicio->valor;
    p = inicio;
    inicio = inicio->proximoElemento;
    delete p;
    if (inicio == nullptr)
        fim = nullptr;
    return true;
}
```

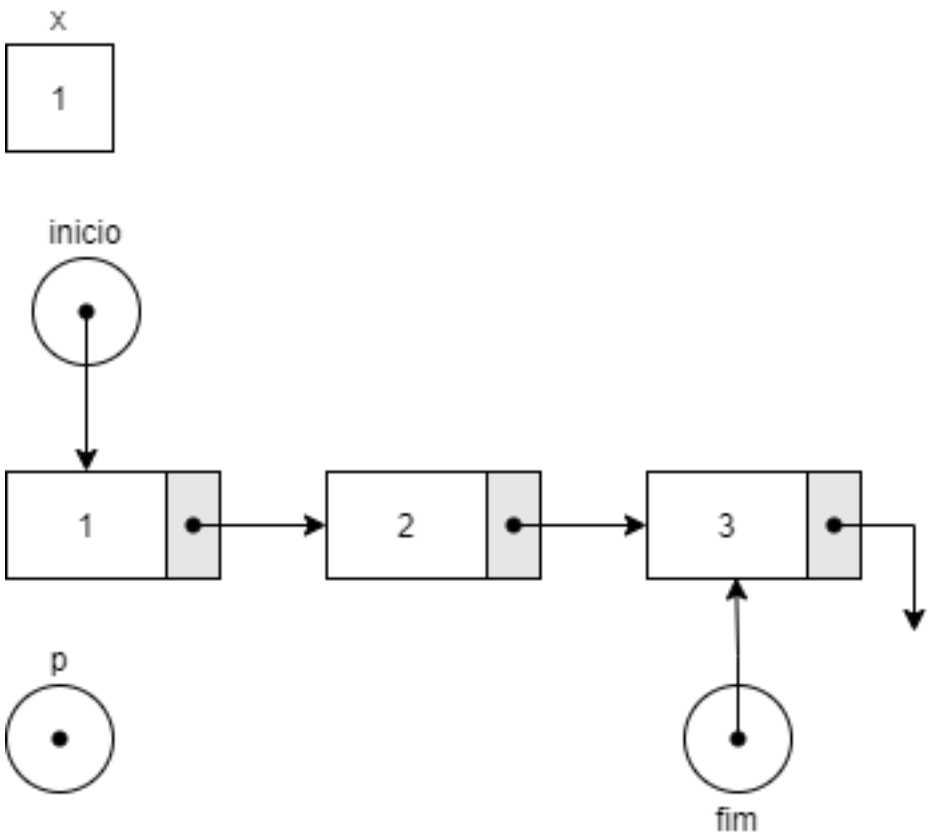
[14]:
```
int y;
if (minhaFila.remover(y))
    cout << y << " removido com sucesso!\n";
else
    cout << "não houve remoção!\n";
```

```
1 removido com sucesso!
```

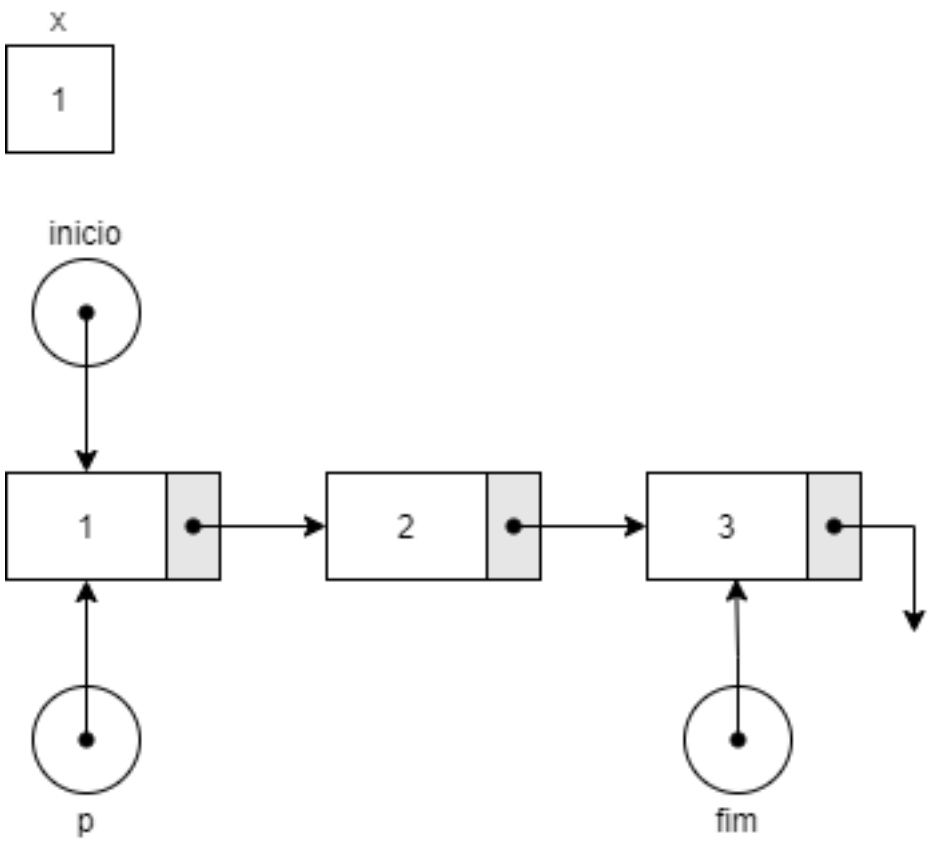
```
PonteiroElemento p;
```



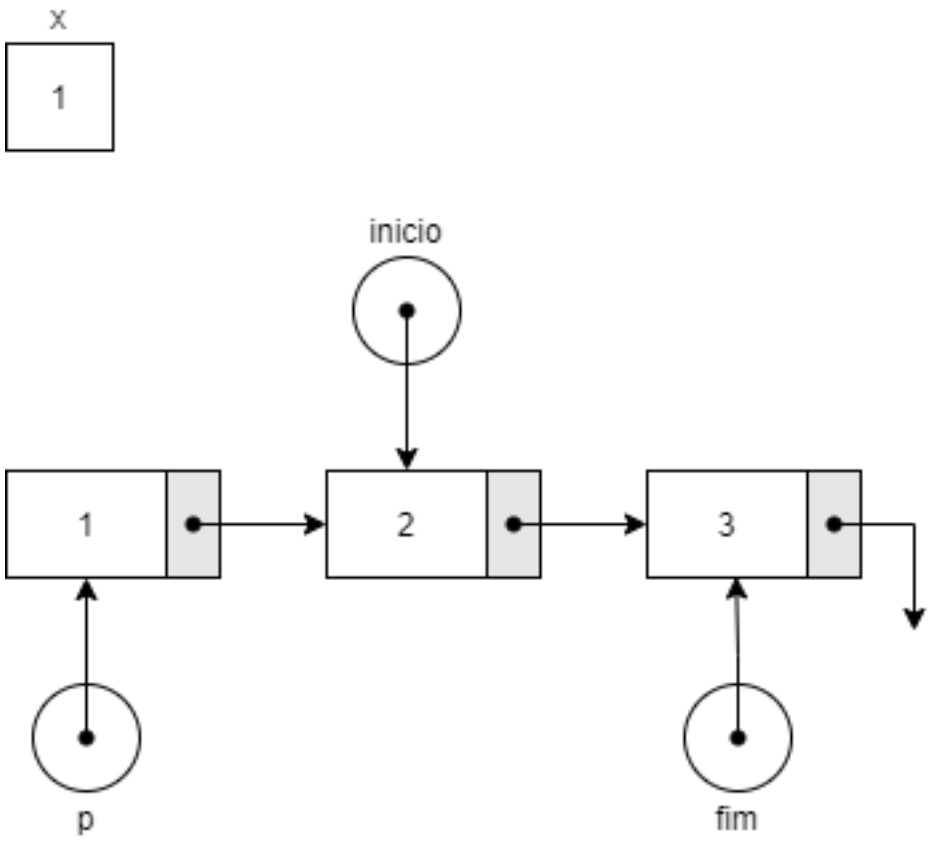
```
if (vazia())
    return false;
x = inicio->valor;
```



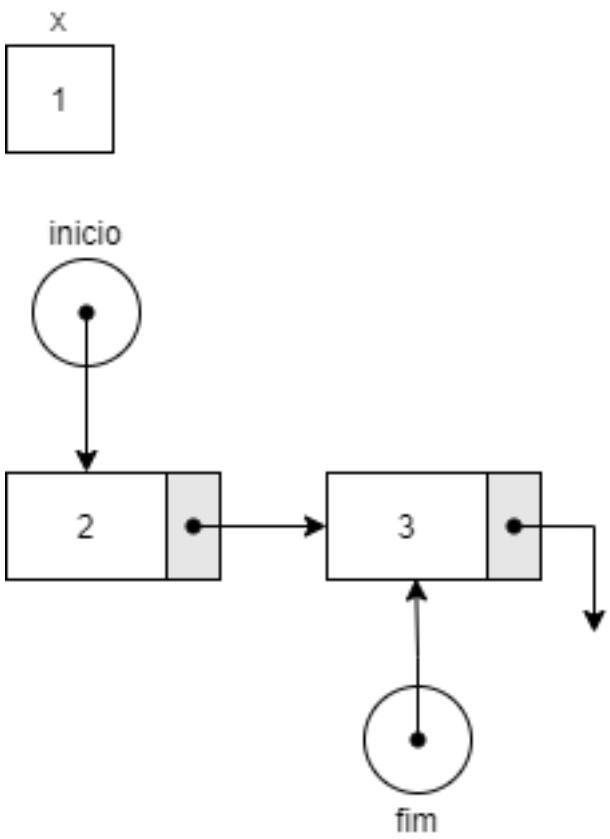

```
p = inicio;
```




```
inicio = inicio->proximoElemento;
```




```
delete p;
```




```
if (inicio == NULL)
    fim = NULL;
return true;
```

[27]:
```
int y;
if (minhaFila.remover(y))
    cout << y << " removido com sucesso!\n";
else
    cout << "não houve remoção!\n";
```

```
2 removido com sucesso!
```

[28]:
```
int y;
if (minhaFila.remover(y))
    cout << y << " removido com sucesso!\n";
else
    cout << "não houve remoção!\n";
```

```
3 removido com sucesso!
```

[29]:
```
int y;
if (minhaFila.remover(y))
    cout << y << " removido com sucesso!\n";
```

```
else
    cout << "não houve remoção!\n";
```

```
não houve remoção!
```

1.1.7 Outros testes

Teste 01

[30]:
```
Fila minhaFila;
```

[31]:
```
int y;
for (int i = 0; i < 4 ; i ++) {
    cout << "Digite um valor [" << i << "]: ";
    cin >> y;
    minhaFila.inserir(y);
}
```

```
Digite um valor [0]:
 3
Digite um valor [1]:
 4
Digite um valor [2]:
 5
Digite um valor [3]:
 6
```

[32]:
```
for (int i = 0; i < 6 ; i ++) {
    if (minhaFila.remover(y))
        cout << "Removido [" << i << "]: " << y << " - Vazia: " << minhaFila.
 ↪vazia() << "\n";
    else
        cout << "Sem elementos para remover!" << " - Vazia: " << minhaFila.
 ↪vazia() << "\n";
}
```

```
Removido [0]: 3 - Vazia: 0
Removido [1]: 4 - Vazia: 0
Removido [2]: 5 - Vazia: 0
Removido [3]: 6 - Vazia: 1
Sem elementos para remover! - Vazia: 1
Sem elementos para remover! - Vazia: 1
```

Teste 02

[33]:
```
Fila minhaFila;
```

[39]:
```
for (int i = 4; i < 20; i++) {
    if (minhaFila.inserir(i))
        cout << i << " inserido com sucesso!\n";
    else
        cout << i << " não foi inserido!\n";
}
```

```
4 inserido com sucesso!
5 inserido com sucesso!
6 inserido com sucesso!
7 inserido com sucesso!
8 inserido com sucesso!
9 inserido com sucesso!
10 inserido com sucesso!
11 inserido com sucesso!
12 inserido com sucesso!
13 inserido com sucesso!
14 inserido com sucesso!
15 inserido com sucesso!
16 inserido com sucesso!
17 inserido com sucesso!
18 inserido com sucesso!
19 inserido com sucesso!
```

[40]:
```
minhaFila.inserir(22);
```

[41]:
```
for (int i = 4; i < 24; i++) {
    if (minhaFila.remover(y)) {
        cout << i << ": " << y << " removido com sucesso! - ";
        if (y == i)
            cout << " [ok]\n";
        else
            cout << " [not ok]\n";
    }
    else
        cout << " fila vazia!\n";
}
```

```
4: 4 removido com sucesso! -  [ok]
5: 5 removido com sucesso! -  [ok]
6: 6 removido com sucesso! -  [ok]
7: 7 removido com sucesso! -  [ok]
8: 8 removido com sucesso! -  [ok]
9: 9 removido com sucesso! -  [ok]
10: 10 removido com sucesso! -  [ok]
11: 11 removido com sucesso! -  [ok]
12: 12 removido com sucesso! -  [ok]
13: 13 removido com sucesso! -  [ok]
```

```
14: 14 removido com sucesso! -  [ok]
15: 15 removido com sucesso! -  [ok]
16: 16 removido com sucesso! -  [ok]
17: 17 removido com sucesso! -  [ok]
18: 18 removido com sucesso! -  [ok]
19: 19 removido com sucesso! -  [ok]
20: 22 removido com sucesso! -  [not ok]
 fila vazia!
 fila vazia!
 fila vazia!
```

[ ]:

[ ]:

[ ]:

[42]:
```
%%file Fila.h
class Fila {
    private:
        struct elemento {
            int valor;
            elemento *proximoElemento; // ligação próximo nó
        };
        typedef elemento *PonteiroElemento;
        PonteiroElemento inicio;
        PonteiroElemento fim;
    public:
        Fila();
        bool vazia();
        bool cheia();
        bool inserir(int x);
        bool remover(int &x);
};
```

```
Writing Fila.h
```

[43]:
```
%%file Fila.cpp
#include <cstdlib>
#include "Fila.h"

Fila::Fila() {
    inicio = NULL;
    fim = NULL;
}

bool Fila::vazia() {
```

```
    return (inicio == NULL);
}

bool Fila::cheia() {
    return false;
}

bool Fila::inserir(int x) {
    PonteiroElemento p;
    p = new elemento;
    p->valor = x;
    if (vazia()) {
        inicio = p;
        fim = p;
    } else {
        fim->proximoElemento = p;
        fim = p;
    }
    p->proximoElemento = NULL;
    return true;
}

bool Fila::remover(int &x) {
    PonteiroElemento p;
    if (vazia())
        return false;
    x = inicio->valor;
    p = inicio;
    inicio = inicio->proximoElemento;
    delete p;
    if (inicio == NULL)
        fim = NULL;
    return true;
}
```

```
Writing Fila.cpp
```

[44]:
```
%%file teste01.cpp
#include <iostream>
#include "Fila.cpp"

using namespace std;

int main() {
    Fila minhaFila;
    int x, y;
```

```
    if (minhaFila.vazia()) {
        cout << "Esta vazia!\n";
    }
    minhaFila.inserir(1);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
    minhaFila.inserir(2);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
    minhaFila.remover(x);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
    minhaFila.remover(x);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
}
```

```
Writing teste01.cpp
```

[45]:
```
!g++ -o teste01 teste01.cpp
```

[46]:
```
!./teste01
```

```
Esta vazia!
Esta vazia: 0
Esta vazia: 0
Esta vazia: 0
Esta vazia: 1
```

[ ]:
```
```

[ ]:
```
%%file Teste01.h
#include <cxxtest/TestSuite.h>
#include "Fila.cpp"

class MyTestSuite1 : public CxxTest::TestSuite
{
public:
    void testaVazia(void)
    {
        Fila fila01;
        int y;

        TS_ASSERT(fila01.vazia() == true);
        TS_ASSERT(fila01.cheia() == false);

        TS_ASSERT(fila01.inserir(1) == true);
        TS_ASSERT(fila01.vazia() == false);
        TS_ASSERT(fila01.cheia() == false);

        TS_ASSERT(fila01.inserir(2) == true);
```

```
        TS_ASSERT(fila01.vazia() == false);
        TS_ASSERT(fila01.cheia() == false);

        TS_ASSERT(fila01.remover(y) == true);
        TS_ASSERT(y == 1);
        TS_ASSERT(fila01.vazia() == false);
        TS_ASSERT(fila01.cheia() == false);

        TS_ASSERT(fila01.remover(y) == true);
        TS_ASSERT(y == 2);
        TS_ASSERT(fila01.vazia() == true);
        TS_ASSERT(fila01.cheia() == false);

        fila01.remover(y);
        TS_ASSERT(fila01.vazia() == true);
        TS_ASSERT(fila01.cheia() == false);
    }

    void testaInserir(void)
    {
        Fila fila01;
        int y;


        TS_ASSERT(fila01.inserir(1) == true);

        TS_ASSERT(fila01.inserir(2) == true);

        TS_ASSERT(fila01.remover(y) == true);
        TS_ASSERT(y == 1);

        TS_ASSERT(fila01.remover(y) == true);
        TS_ASSERT(y == 2);

        TS_ASSERT(fila01.remover(y) == false);

    }


};
```

[ ]:
```
!cxxtestgen --error-printer -o runner01.cpp Teste01.h
```

[ ]:
```
!g++ -o runner01 runner01.cpp
```

[ ]:
```
!./runner01
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```
%%file teste01.cpp
#include <iostream>
#include "Fila.cpp"

using namespace std;

int main() {
    Fila minhaFila;
    int x, y;

    if (minhaFila.vazia()) {
        cout << "Esta vazia!\n";
    }
    minhaFila.inserir(1);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
    minhaFila.inserir(2);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
    minhaFila.remover(x);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
    minhaFila.remover(x);
    cout << "Esta vazia: " << minhaFila.vazia() << "\n";
}
```

[ ]:
```
!g++ -o teste01 teste01.cpp
```

[ ]:
```
!./teste01
```

[ ]: