oopPilhaDinamica

September 17, 2021

1 Pilha Dinamica Orientada a Objetos

- Ultimo que entra é o primeiro que sai (LIFO)

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

1.1 Definição da classe

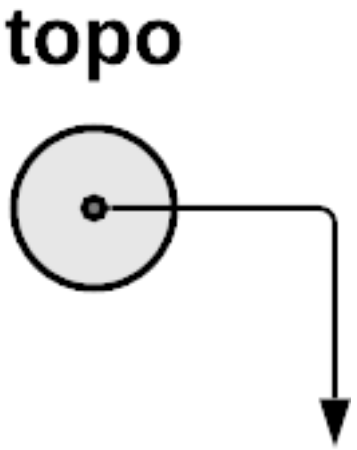
[21]:
```
class Pilha {
    private:
        struct noPilha {
            int valor;
            noPilha *proximoNo; // ligação próximo nó
        };
        typedef noPilha *PonteiroPilha;
        PonteiroPilha topo;
    public:
        Pilha();
        bool vazia();
        bool cheia();
        bool empilhar(int x);
        bool desempilhar(int &x);
        bool retornaTopo(int &x);
};
```

Temos somente um ponteiro do tipo PonteiroPilha chamado de topo:

1.2 Inicializando pilha (construtor)

[22]:
```
Pilha::Pilha() {
    topo = NULL;
}
```

A pilha é iniciada vazia.



[23]:
```
Pilha minhaPilha;
```

1.3 Verificando se pilha está vazia

[24]:
```
bool Pilha::vazia() {
    if (topo == NULL) {
        return true;
    } else {
        return false;
    }
}
```

[25]:
```
if (minhaPilha.vazia() == true)
    cout << "Ok!!!";
```

```
Ok!!!
```

1.4 Verificando se pilha está cheia

[26]:
```
bool Pilha::cheia() {
    return false;
}
```

[27]:
```
if (minhaPilha.cheia() == false)
    cout << "Ok!!!";
```

```
Ok!!!
```

1.5 Empilhando

[28]:
```
bool Pilha::empilhar(int x) {
    PonteiroPilha p;
    p = new noPilha;
    if (p == NULL) {
        return false;
    }
    p->valor = x;
    p->proximoNo = topo;
    topo = p;
    return true;
}
```

[29]:
```
minhaPilha.empilhar(8)
```

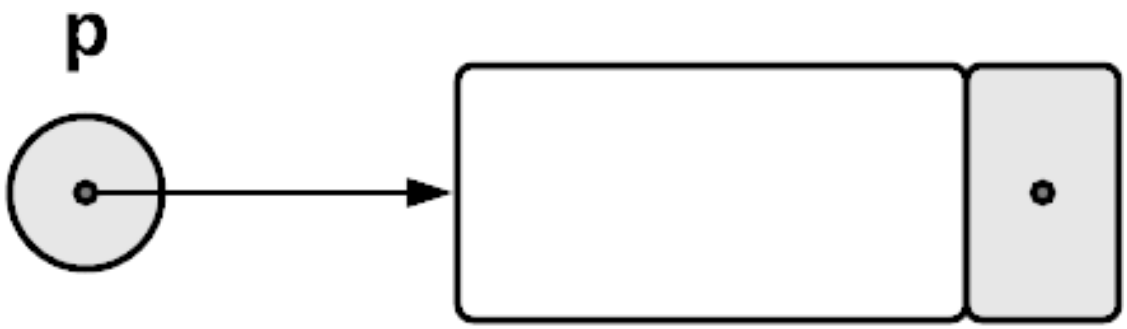
[29]:
```
true
```

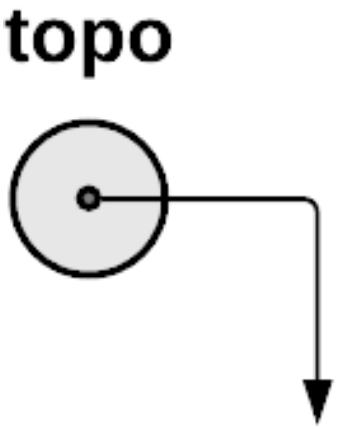
Primeiro o ponteiro p é criado.

```
PonteiroPilha p;
```

Definindo p como um struct do tipo noPilha

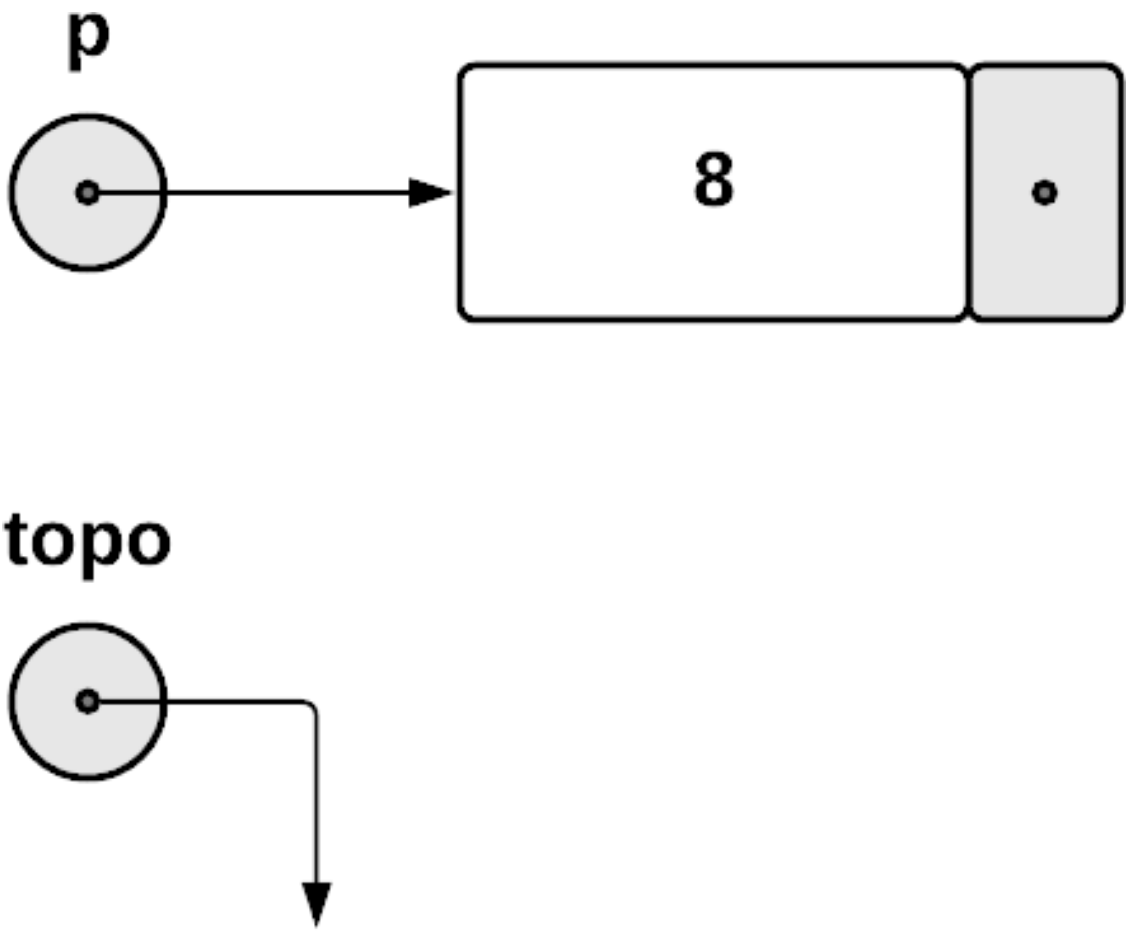
```
p = new noPilha;
```





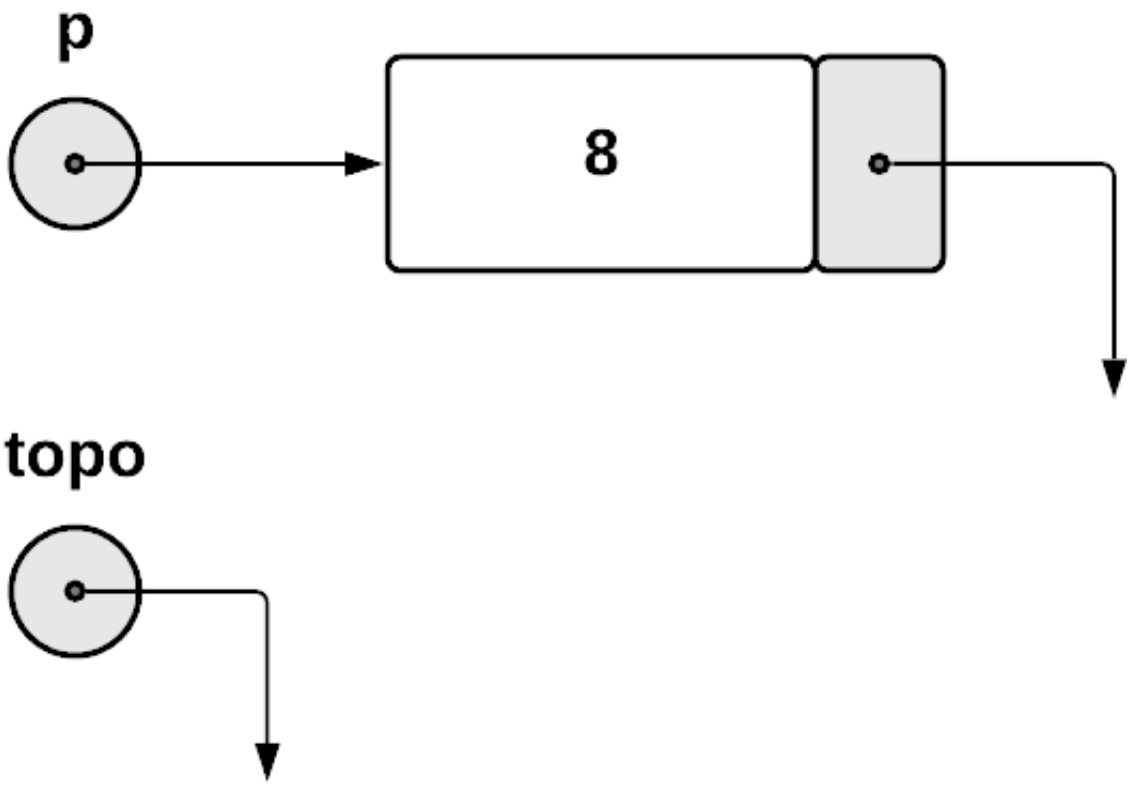
Armazenando o valor 8 em p.valor

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



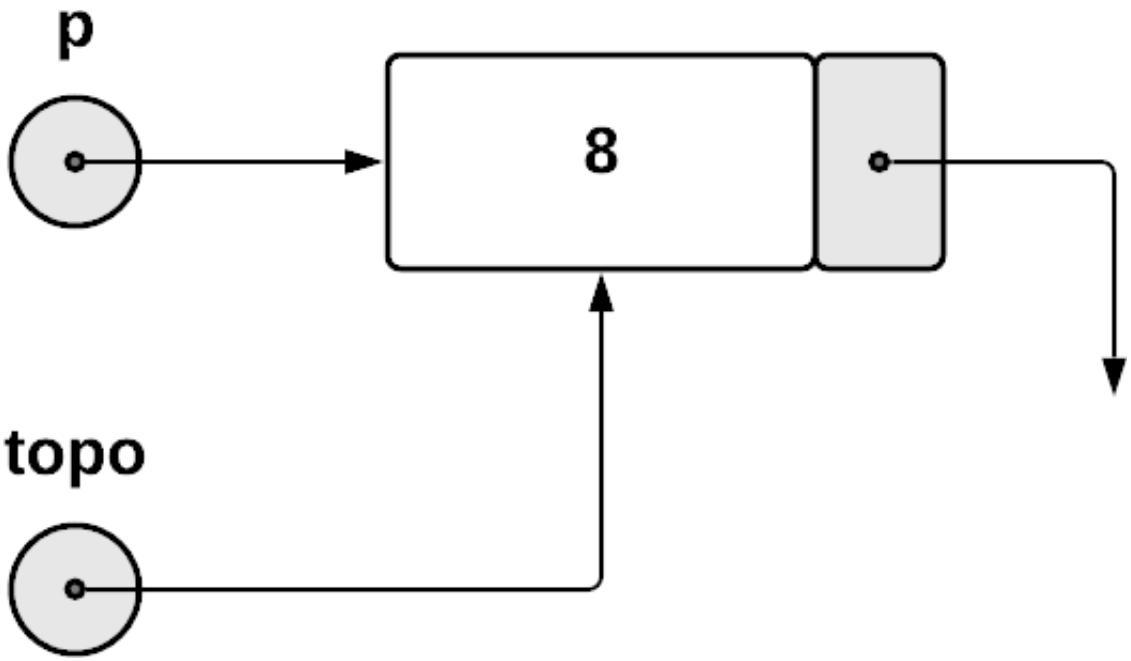
O próximo nó de p aponta para o topo, no caso NULL.

```
p->proximoNo = topo;
```

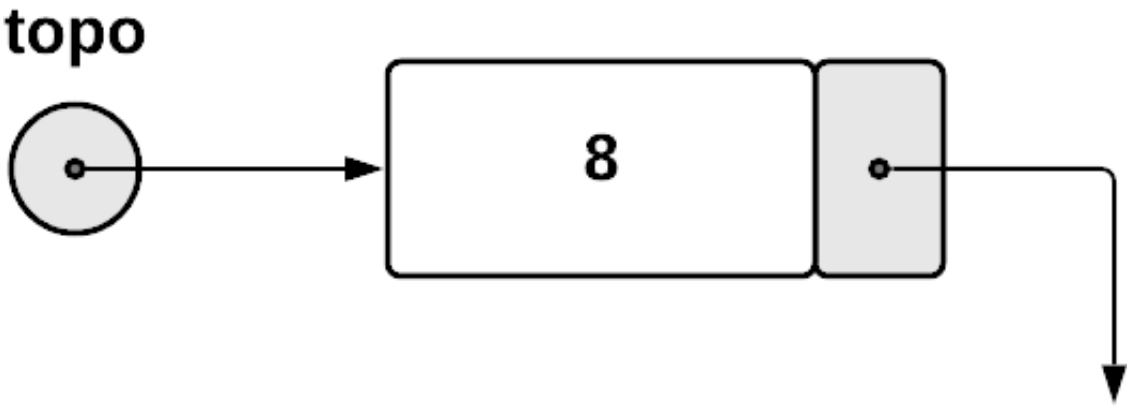


Topo agora aponta para p.

```
topo = p;
```



Como o ponteiro p era pertencente ao método empilhar, ao termino da execução do método permanece apenas o atributo topo.

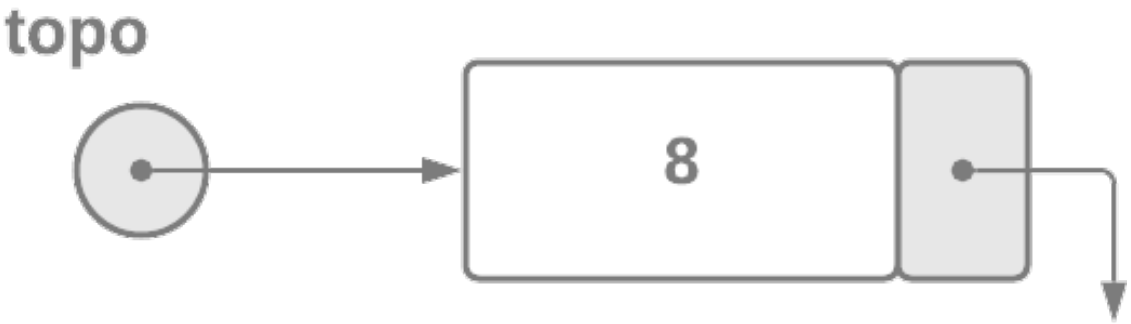


[30]:
```
minhaPilha.empilhar(10)
```

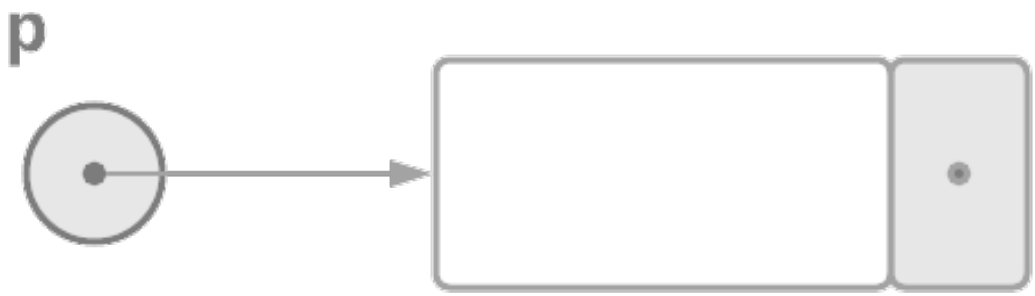
[30]:
```
true
```

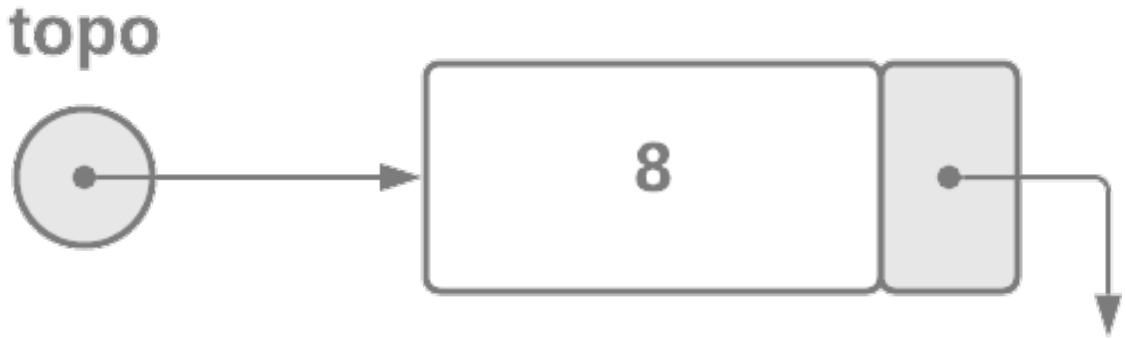
Outro nó p é criado.

```
PonteiroPilha p;
```

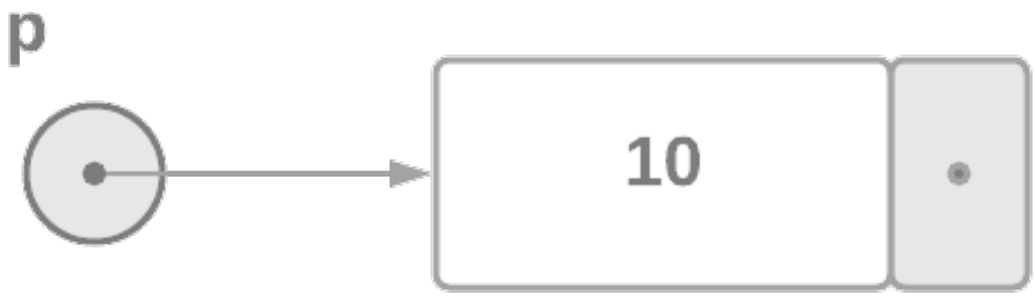


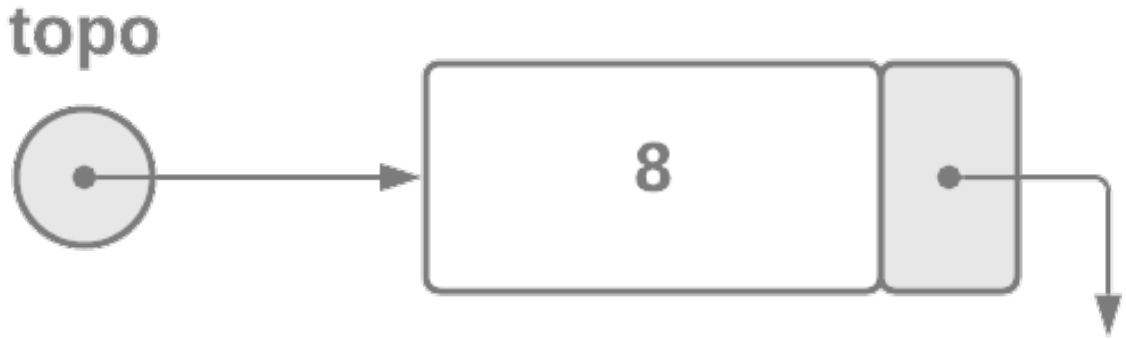

```
p = new noPilha;
```



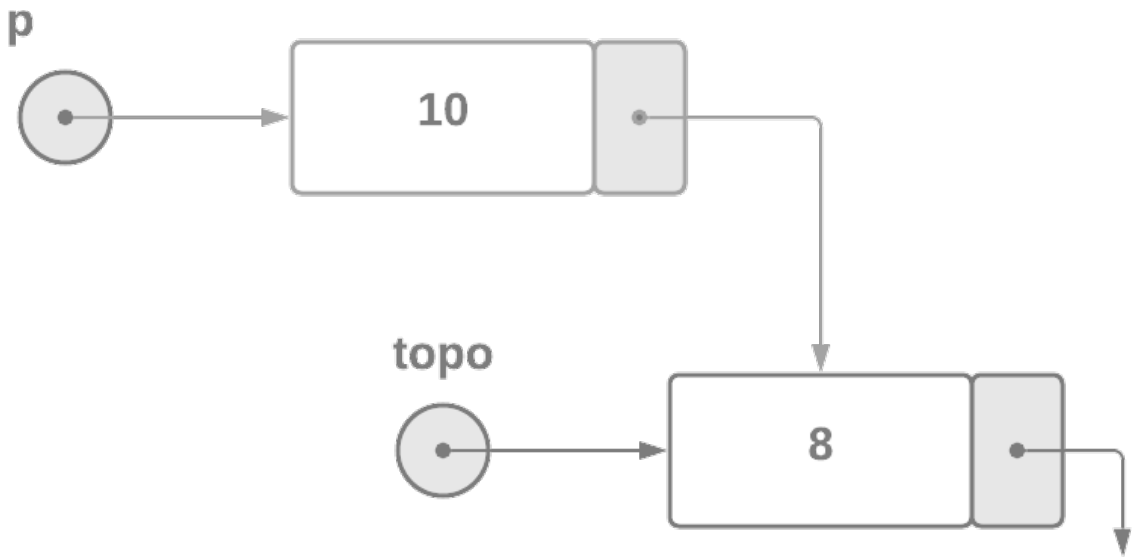



```
p->valor = x;
```

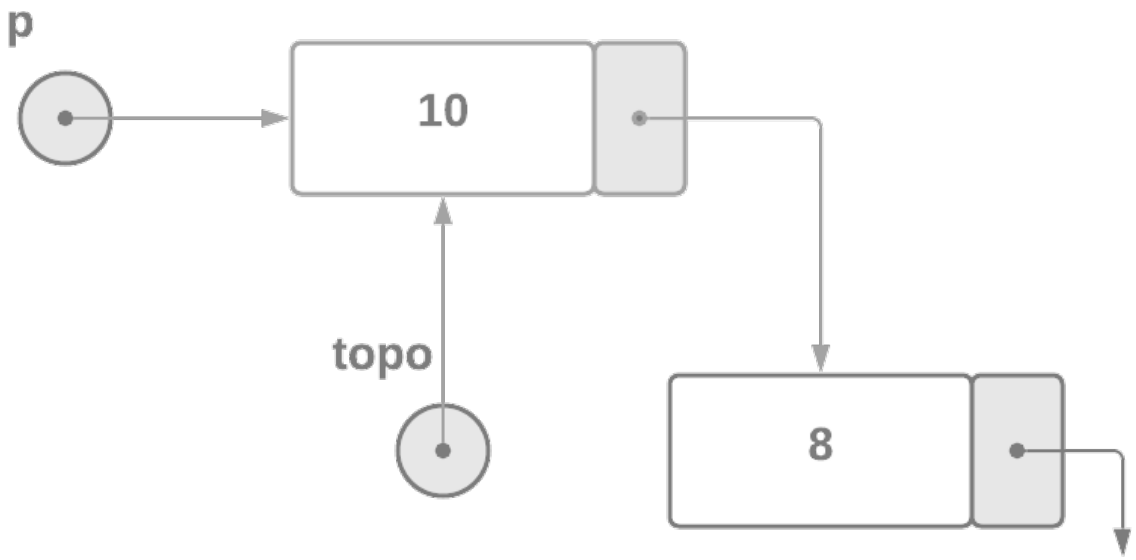




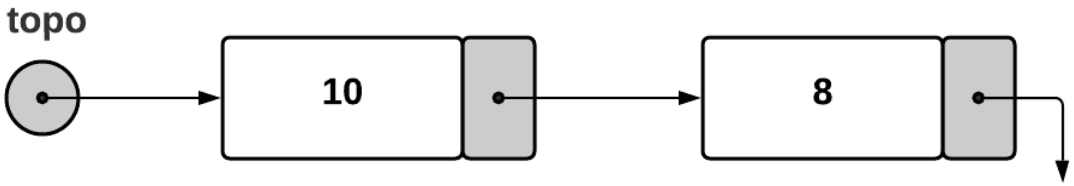

```
p->proximoNo = topo;
```




```
topo = p;
```



Como o ponteiro p era pertencente ao método empilhar, ao termino da execução do método permanece apenas o atributo topo.

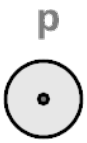


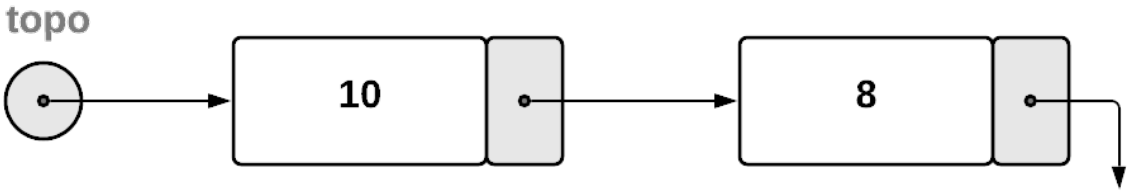
[31]:
```
minhaPilha.empilhar(4)
```

[31]:
```
true
```

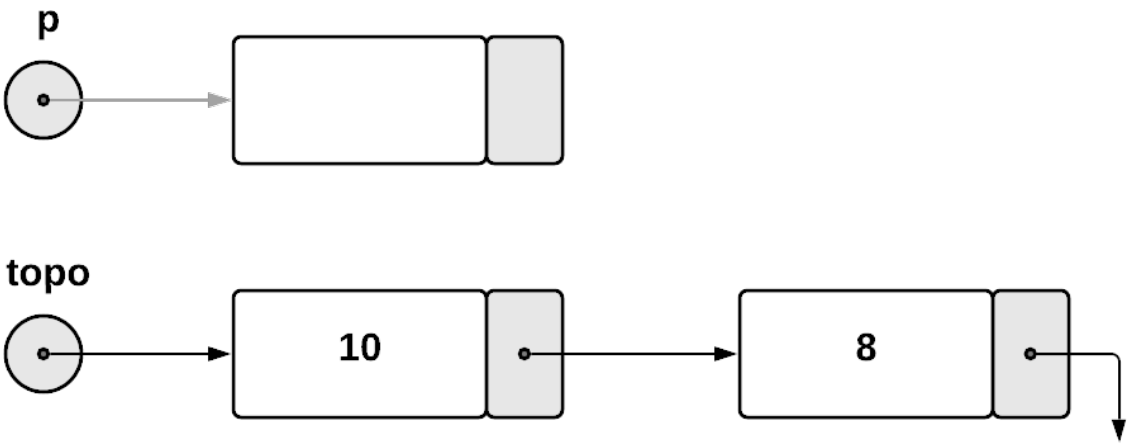
Outro nó p é criado.

```
PonteiroPilha p;
```

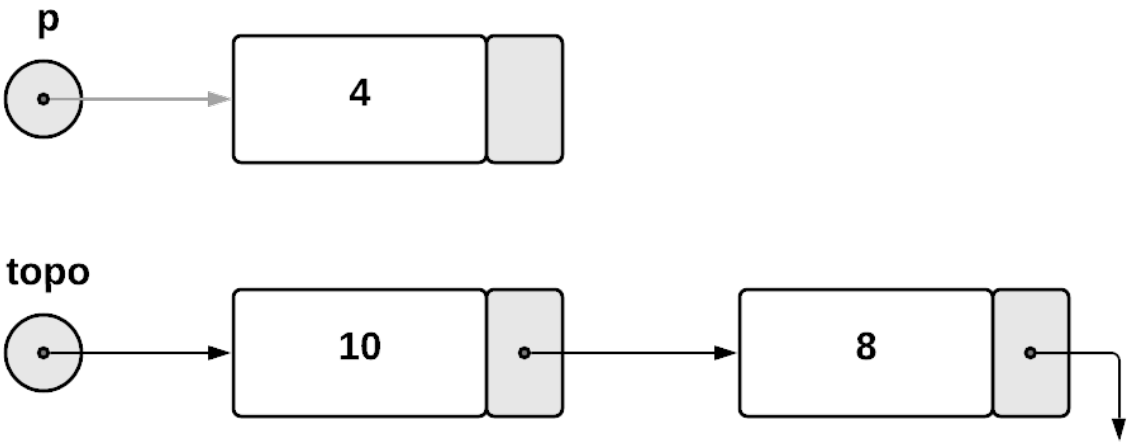




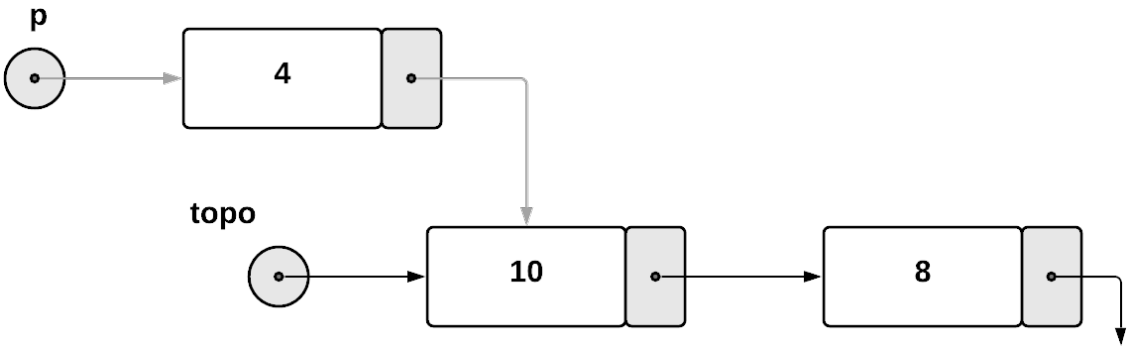

```
p = new noPilha;
```



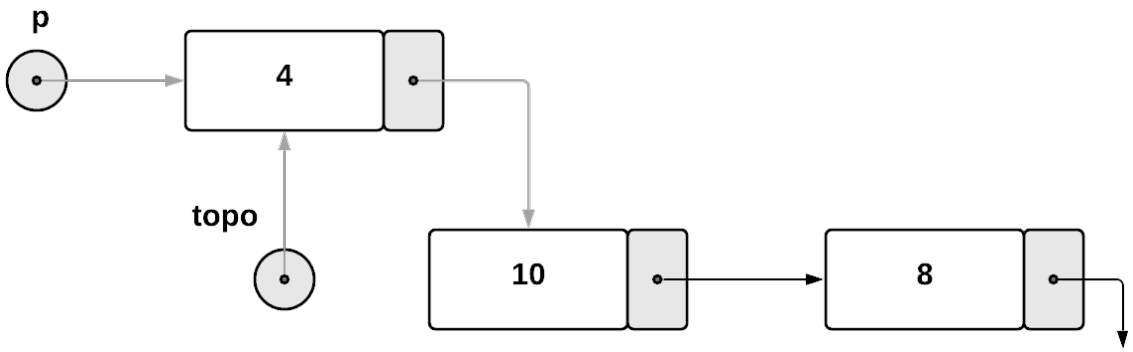

```
p->valor = x;
```



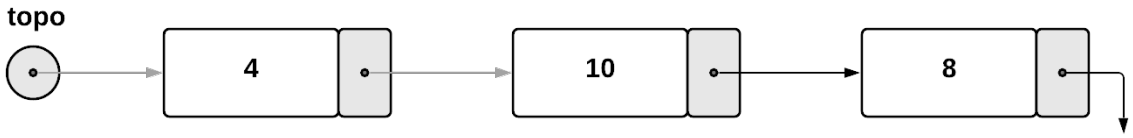

```
p->proximoNo = topo;
```




```
topo = p;
```



Como o ponteiro p era pertencente ao método empilhar, ao termino da execução do método permanece apenas o atributo topo.



1.6 Desempilhar

[32]:
```
int y;
bool resultado;
```

[33]:
```
bool Pilha::desempilhar(int &x) {
    PonteiroPilha p;
    if (vazia()) {
        return false;
    }
    x = topo->valor;
    p = topo;
    topo = topo->proximoNo;
    free(p);
    return true;
}
```

[34]:
```
resultado = minhaPilha.desempilhar(y);
cout << y;
```

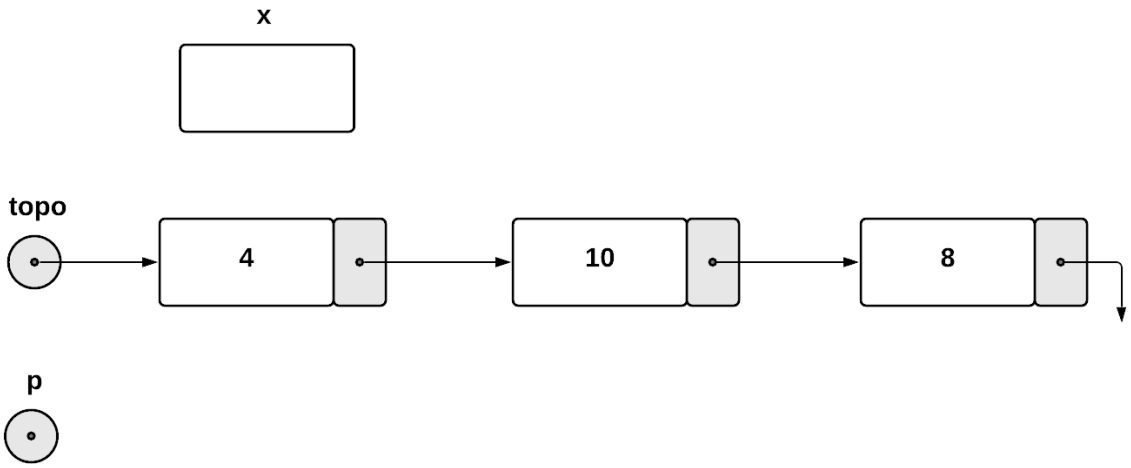
```
4
```

[35]:
```
cout << resultado;
```

```
1
```

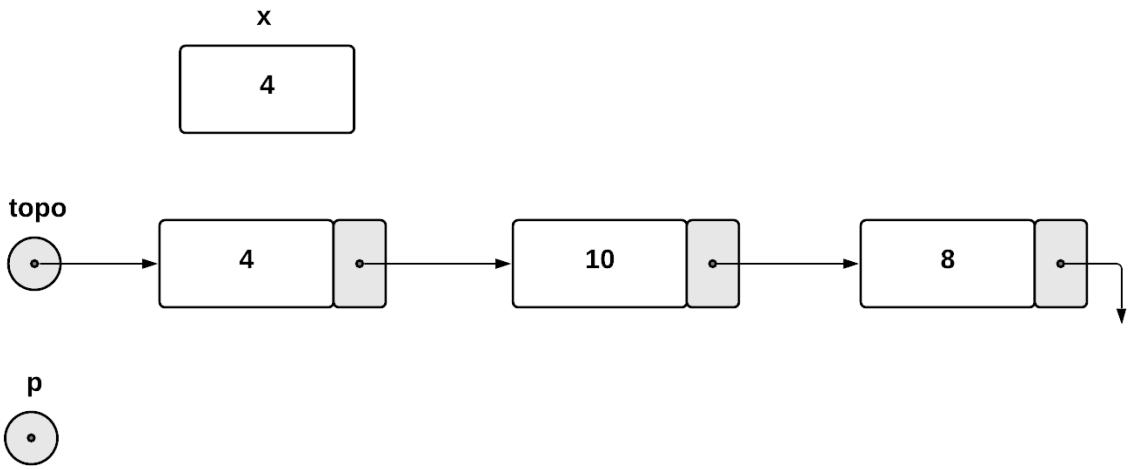
Um ponteiro p do tipo PonteiroP é criado:

```
PonteiroPilha p;
```



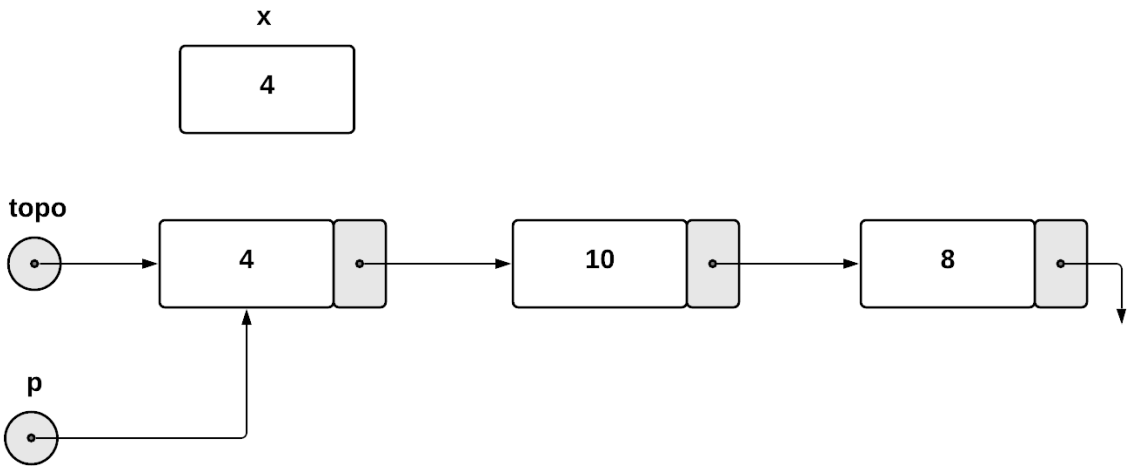
Em x é armazenado o valor atual de topo:

```
x = topo->valor;
```



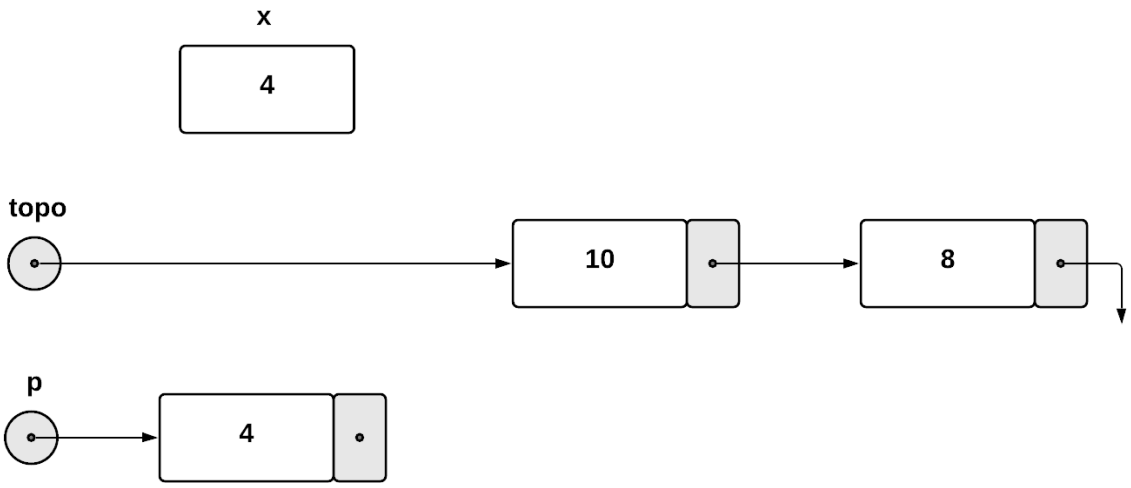
p passa a apontar para topo:

```
p = topo;
```



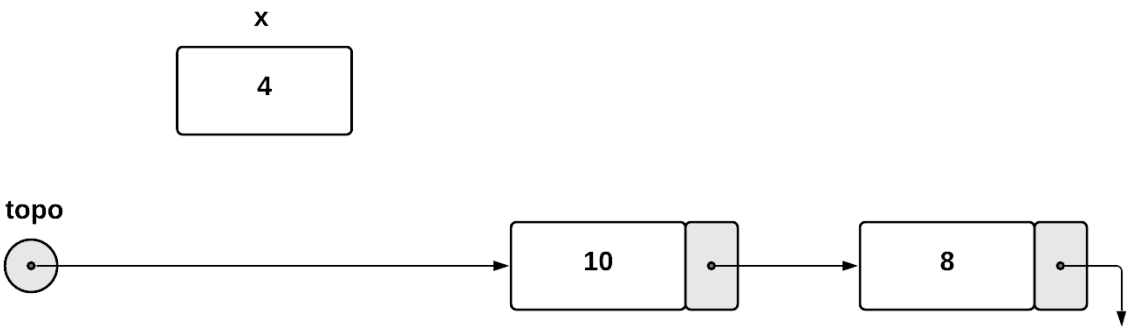
topo passa a apontar para seu próprio proximoNo:

```
topo = topo->proximoNo;
```



p é liberado da memória:

```
free(p);
```



[36]:
```
if (minhaPilha.desempilhar(y))
    cout << y;
else
    cout << "Não desempilhou!";
```

```
10
```

[37]:
```
if (minhaPilha.desempilhar(y))
    cout << y;
else
    cout << "Não desempilhou!";
```

```
8
```

[38]:
```
if (minhaPilha.desempilhar(y))
    cout << y;
```

```
else
    cout << "Não desempilhou!";
```

```
Não desempilhou!
```

[ ]:

[39]:
```
minhaPilha.empilhar(0);
```

[40]:
```
if (minhaPilha.desempilhar(y))
    cout << y;
else
    cout << "Não desempilhou!";
```

```
0
```

[41]:
```
if (minhaPilha.desempilhar(y))
    cout << y;
else
    cout << "Não desempilhou!";
```

```
Não desempilhou!
```

[42]:
```
minhaPilha.empilhar(1);
minhaPilha.empilhar(0);
```

[43]:
```
if (minhaPilha.desempilhar(y))
    cout << "desempilhou: " << y;
else
    cout << "não desempilhou!";
```

```
desempilhou: 0
```

[44]:
```
if (minhaPilha.desempilhar(y))
    cout << "desempilhou: " << y;
else
    cout << "não desempilhou!";
```

```
desempilhou: 1
```

[45]:
```
if (minhaPilha.desempilhar(y))
    cout << "desempilhou: " << y;
else
    cout << "não desempilhou!";
```

```
não desempilhou!
```

1.7 Retorna topo

[46]:
```
minhaPilha.empilhar(4);
```

[47]:
```
bool Pilha::retornaTopo(int &x) {
    if (vazia()) {
        return false;
    }
    x = topo->valor;
    return 1;
}
```

[48]:
```
if (minhaPilha.retornaTopo(y))
    cout << y;
else
    cout << "Pilha vazia!";
```

```
4
```

[49]:
```
if (minhaPilha.retornaTopo(y))
    cout << y;
else
    cout << "Pilha vazia!";
```

```
4
```

[50]:
```
minhaPilha.desempilhar(y);
```

[51]:
```
if (minhaPilha.retornaTopo(y))
    cout << y;
else
    cout << "Pilha vazia!";
```

```
Pilha vazia!
```

[ ]:

[ ]:

1.8 Testando pilha

[52]:
```
Pilha minhaPilha;
```

[53]:
```
for (int i = 10 ; i < 15 ; i++) {
    minhaPilha.empilhar(i);
    cout << i << endl;
}
```

```
10
11
12
13
14
```

[54]:
```
while(minhaPilha.desempilhar(y)) {
    cout << y << endl;
}
```

```
14
13
12
11
10
```

[55]:
```
if  (minhaPilha.vazia() == true)
    cout << "ok";
```

```
ok
```

[ ]:

[ ]:

[ ]:
```
if (minhaPilha.estaVazia())
    cout << "Esta vazia!\n";
else
    cout << "Não esta vazia!\n";
```

[ ]:
```
cout << "Empilhando: 4\n";
if (minhaPilha.empilhar(4))
    cout << "Empilhado\n";
else
    cout << "Não empilhou\n";
```

[ ]:
```
cout << "Empilhando: 3\n";
if (minhaPilha.empilhar(3))
    cout << "Empilhado\n";
else
    cout << "Não empilhou\n";
```

[ ]:
```
cout << "Empilhando: 2\n";
if (minhaPilha.empilhar(2))
    cout << "Empilhado\n";
else
    cout << "Não empilhou\n";
```

[ ]:
```
cout << "Empilhando: 1\n";
if (minhaPilha.empilhar(1))
    cout << "Empilhado\n";
else
    cout << "Não empilhou\n";
```

[ ]:
```
cout << "Topo da pilha: " << minhaPilha.retornaTopo() << "\n";
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```
minhaPilha.empilhar(0)
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```
if (minhaPilha.desempilhar(x))
    cout << "Desempilhou: " << x << "\n";
else
    cout << "Pilha vazia!";
```

[ ]:
```

```