

Pilha, Fila e Lista

- ▶ Para formarmos uma sequência e termos em C++ com alocação dinâmica, podemos utilizar 3 tipos de métodos.

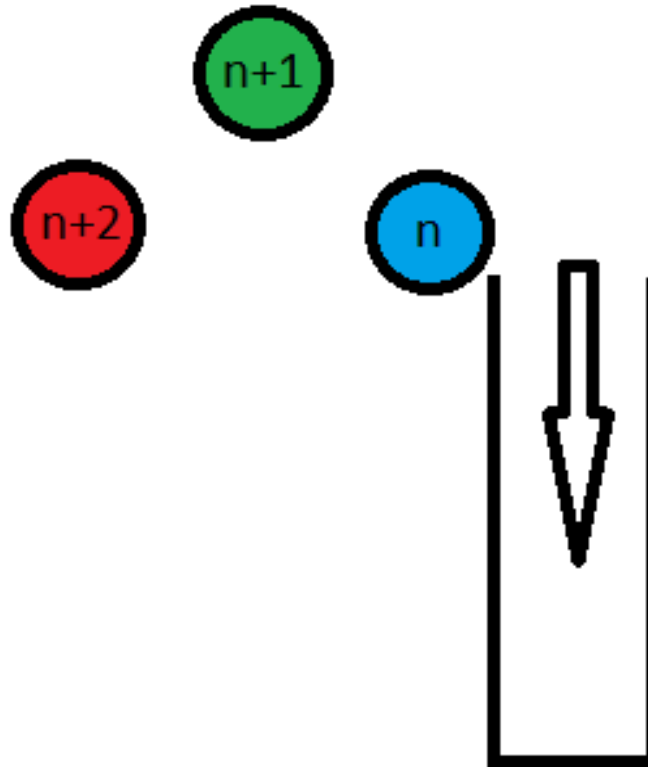
Pilha(Stack)

- ▶ A pilha é um método que guarda os termos de maneira em que o último termo a ser colocado será o primeiro termo a sair e o primeiro termo a ser colocado será o último a sair.
- ▶ Veja o exemplo:

Ordem de uma pilha:



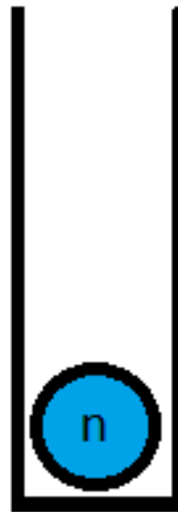
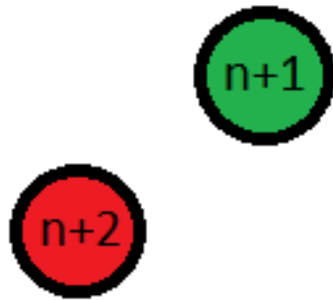
← Pilha



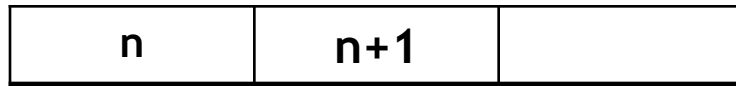
Ordem de uma pilha:



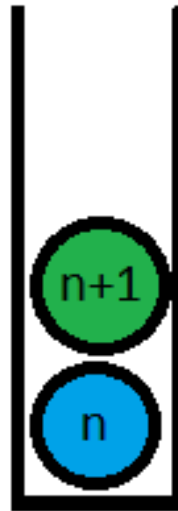
← Pilha



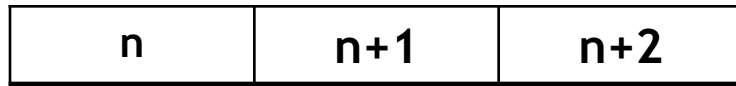
Ordem de uma pilha:



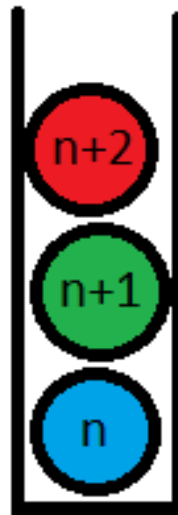
← Pilha



Ordem de uma pilha:



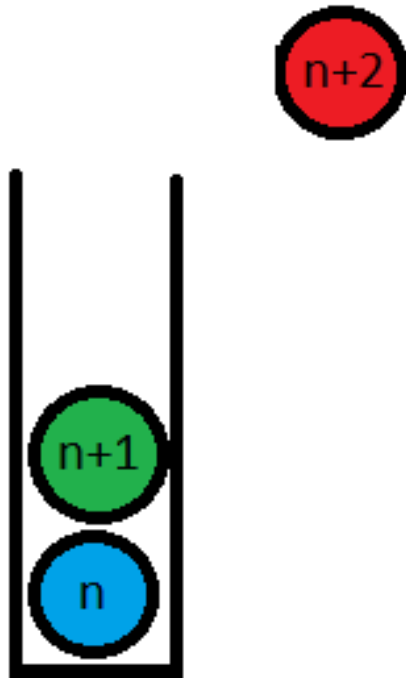
← Pilha



Ordem de uma pilha:

n	n+1	n+2
---	-----	-----

← Pilha



Comandos da pilha:

- ▶ Para acessar a lista de comandos da pilha é necessário, primeiro, incluir a biblioteca “stack”.
- ▶ Os comandos que podem ser usados em uma pilha são:
- ▶ `.push()` Adiciona elementos à pilha;
- ▶ `.top()` Mostra o último elemento da pilha;
- ▶ `.pop()` Retira o último elemento da pilha;
- ▶ `.empty()` Usado para se referir à pilha quando a mesma está vazia;
- ▶ Mostrar `pilha_exemplo.cpp`;

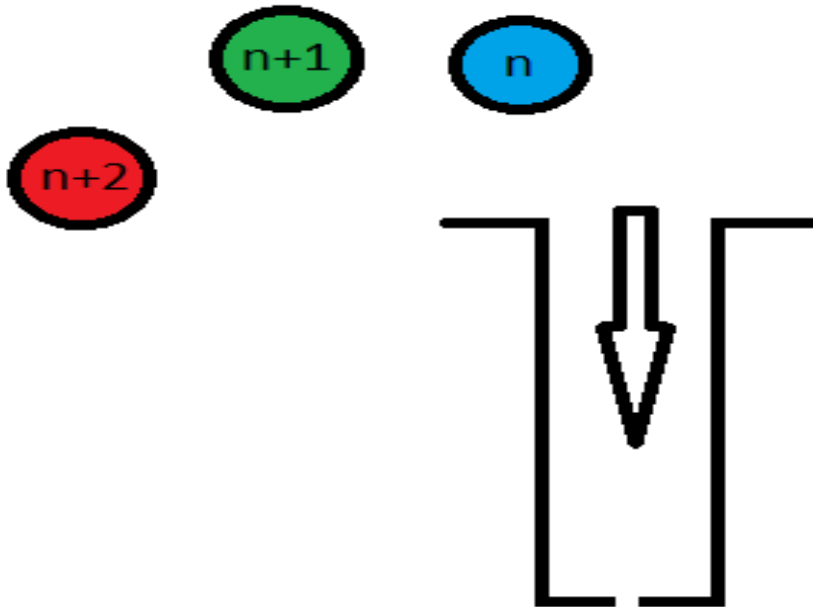
Fila(Queue)

- ▶ A fila é um método que se assemelha muito à pilha, porém, com a fila, o primeiro número a entrar é o primeiro a sair, veja no exemplo:

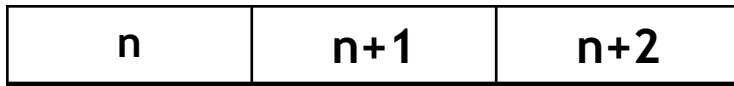
Ordem de uma Fila:



←Fila



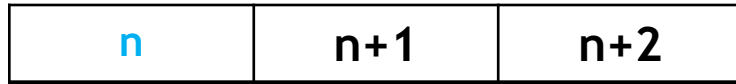
Ordem de uma Fila:



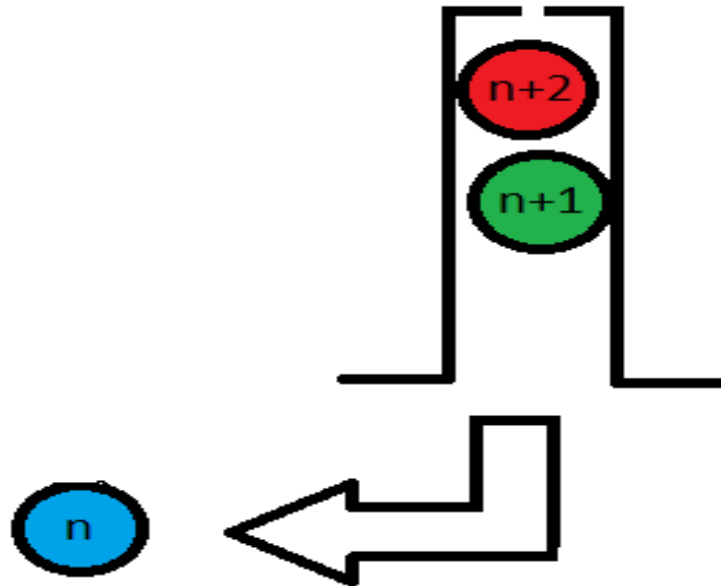
←Fila



Ordem de uma Fila:



← Fila



Fila

- ▶ Ao contrário da pilha, não usamos o comando `.top()`, mas sim os comandos `.front()` e `.back()`. Tirando isso, todos os comandos que valem para a pilha, valem para a fila.
- ▶ Mostrar Fila_exemplo.cpp

Desafio Pilha/Fila:

- Faça um programa em que o usuário possa escolher alguns números de uma pilha chamada “numeros” e adicioná-los a uma fila chamada “Meus_numeros”. No final, mostre os números escolhidos pelo usuário;

Lista(list)

- ▶ A lista é mais versátil do que a pilha e do que a fila, pois, com ela, você pode inserir e retirar termos tanto por cima quanto por baixo. Usando os comandos:
- ▶ `.push_front()` Insere termos por cima da lista
- ▶ `.push_back()` Insere termos por baixo da lista
- ▶ `.pop_front()` Retira termos por cima da lista
- ▶ `.pop_back()` Retira termos por baixo da lista
- ▶ `.clear()` Apaga todos os termos da lista
- ▶ Mostrar arquivo Lista_ex1.cpp;

Métodos da lista

- ▶ Podemos usar 3 Métodos uteis na lista, que são:
- ▶ 1- Iterator
- ▶ 2- .sort()
- ▶ 3- .reverse()

Iterator

- O iterator é um método que pode ser utilizado para inserir termos em uma determinada posição da lista. Apesar das especificidades, pode se tornar bem útil.

```
list<int>::iterator it;  
it = ex.begin();  
advance(it,5);  
ex.insert(it,0);
```

Iterator

- ▶ Para utilizar um Iterator, são necessários 4 passos:
- ▶ I) Declarar o iterator:

```
#include<iostream>
#include<list>
```

```
using namespace std;
```

```
int main(){
    int i;
    list<int> ex;
```

```
    list<int>::iterator it; //declarar o iterator
```

Iterator

- II) Depois que a lista for definida, indicar por onde ele vai começar a contar. Pode ser por cima ou por baixo:

```
for(i=1 ; i<11 ; i++){  
    ex.push_front(i); // definir os termos da lista  
}
```

```
it = ex.begin();  
//ou it= ex.end(); se for começar a contar por baixo
```

Iterator

- ▶ III) Assim que intarator for declarado, é necessário avançar ele um número x de posições para informar onde ele irá inserir o número:

```
advance(it,5);
```

```
//o iterator "it" avançará 5 casas
```

Iterator

- ▶ IV) Com a distância definida, agora falta apenas informar com qual finalidade o iterator será utilizado
- ▶ `.insert()` : inserir um valor na fila
- ▶ `.erase()` : apagar um valor na fila
- ▶ `.merge()` : adiciona o valor de uma lista em outra.

Mostrar `Lista_iterator_ex.cpp`

Sort e Reverse

- ▶ Sort e Reverse são dois métodos simples da lista, sendo `.sort()` responsável por ordenar os termos de uma lista e `.reverse()` responsável por inverter a ordem da lista.
- ▶ Mostrar Lista_sort_reverse.cpp

Desafio Fila

- ▶ Faça um programa em que o usuário põe matrículas em uma lista “matricula” e, com isso, possa interagir de diferentes maneiras. A interface deve ter as opções:
- ▶ 1) Inserir matrícula (sempre que uma nova matrícula for inserida, ela deve, primeiro, ser colocada em uma fila diferente para depois ser adicionada na lista utilizando o método `.merge()`);
- ▶ 2) Apagar matrícula (a matrícula deve ser apagada usando o método `.erase()`);
- ▶ 3) Limpar fila (a lista deve ser apagada com o método `.clear()`);
- ▶ 4) Exibir lista (a lista deve ser inserida de maneira crescente).