

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PIAUÍ
Campus Teresina - Central

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA-CENTRAL

DIRETORIA DE ENSINO

Estrutura de Dados

Aula 5 – A biblioteca STL e uso de Pilhas (Stack) e Filas (Queue)

Professora: Elanne Cristina O. dos Santos

elannecristina.santos@gmail.com

elannecristina.santos@ifpi.edu.br

STL

- A STL – Parte da biblioteca padrão do C++, a Standard Template Library é um conjunto de tipos abstratos de dados, e funções projetados para **manipularem diferentes tipos de dados de forma transparente**.
- A STL faz uso de templates, já definidos, para implementação de diversos algoritmos que manipulam dados de forma eficiente, como por exemplo **containers (vetores, listas, pilhas, etc)**.
- No STL também foi adicionado um tipo *string*, que facilita as operações de manipulação de caracteres quando comparado a biblioteca `string.h` da linguagem C.

STL e namespace

- Ao se usar STL, deve-se usar o conceito de ***namespace***, que permite agrupar classes, objetos e funções e associá-los a um nome, o que facilita a quebra de escopos globais em subescopos menores. Ex:

```
namespace teste
```

```
{ int a, b; }
```

```
int main()
```

```
{   teste::a = 8;  
    printf("%d", teste::a);  
    return 0;    }
```

Namespace std

- Em STL, todas as classes, objetos e funções são definidos com o **namespace std**. Pode-se **usar o operador de escopo a cada uso da STL (std::vector, por exemplo)** ou **indicar o namespace corrente**, da seguinte forma:

using namespace std;

Contêiner

- Um **contêiner** é um objeto de suporte que armazena uma coleção de outros objetos (seus elementos). Eles são implementados como modelos de classe, o que permite uma grande flexibilidade aos elementos.
- Os contêineres replicam estruturas muito usadas na programação: arrays dinâmicas (**vector**), filas (**queue**), pilhas (**stack**), listas (**list**), árvores (**sets**)...

Contêiners

- Muitos ***contêiners*** têm várias funções em comum e compartilham funcionalidades.
- A decisão de qual tipo de ***contêiner*** usar para uma necessidade específica geralmente **não depende apenas da funcionalidade oferecida pelo recipiente**, mas também da **eficiência de alguns de seus membros (complexidade)**.
- Isto é especialmente verdadeiro para ***contêiners de seqüência***, que oferecem **diferenças na complexidade** entre inserir / remover elementos e acessá-los. Alguns deles são eles: ***array, vector, deque, forward_list, list, queue*** e ***stack***.

STACK (Pilhas com STL)

- Incluir biblioteca `<stack>`:

#include <stack>

- Criar o objeto do tipo da classe ***stack***.

Exemplo:

stack<int> pilha;

Stack STL - Métodos

- Inserir um elemento no topo da pilha

push()

- Remover elemento do topo. Não retorna o elemento removido:

pop()

- Obter elemento no topo da pilha. Ele retorna o elemento **sem** removê-lo:

top()

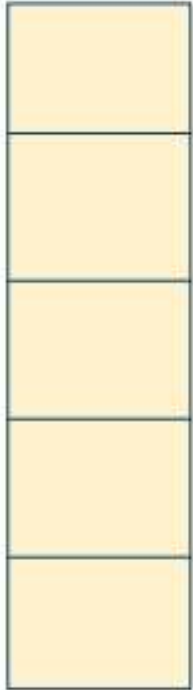
- Verifica se a pilha está vazia

empty()

- Tamanho atual da pilha

size()

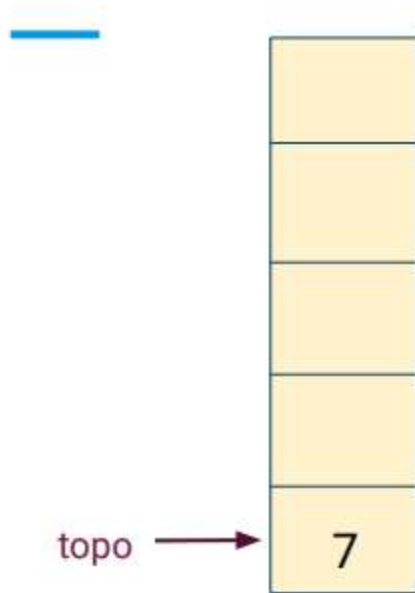
Stack STL - Métodos



```
#include <iostream>
#include <stack>
using namespace std;

int main() {
    stack<int> pilha;
```

Stack STL - Métodos

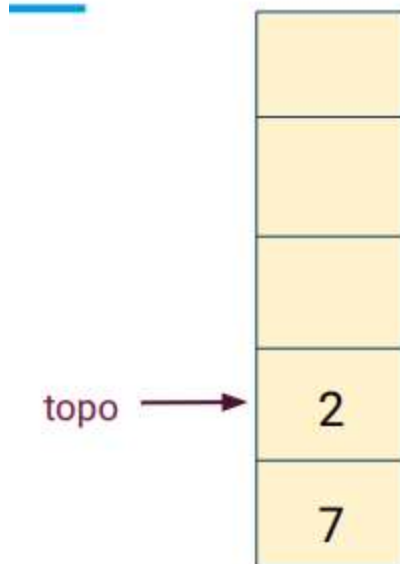


```
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> pilha;
    pilha.push(7);

}
```

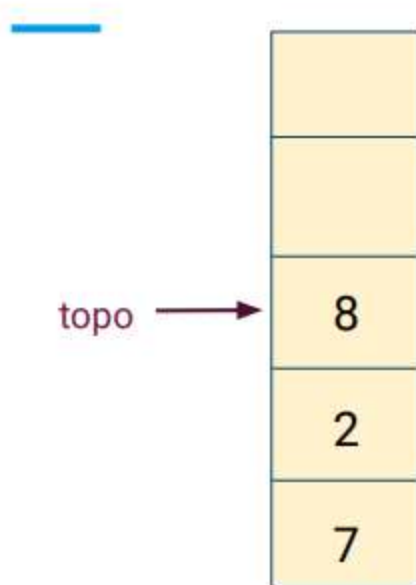
Stack STL - Métodos



```
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
}
```

Stack STL - Métodos

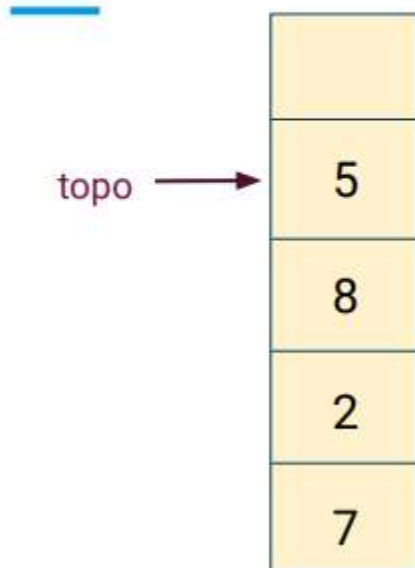


```
#include <iostream>
#include <stack>
using namespace std;

int main() {
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);

}
```

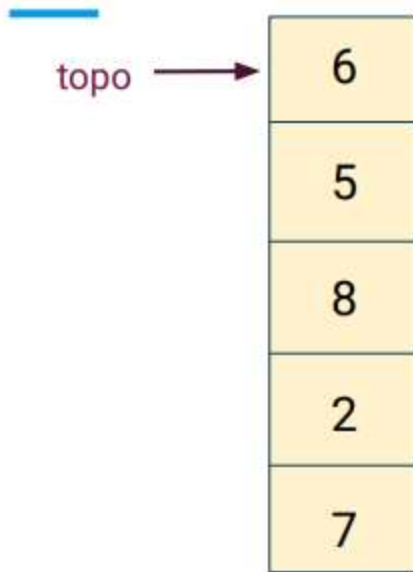
Stack STL - Métodos



```
#include <iostream>
#include <stack>
using namespace std;

int main() {
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);
    pilha.push(5);
}
```

Stack STL - Métodos

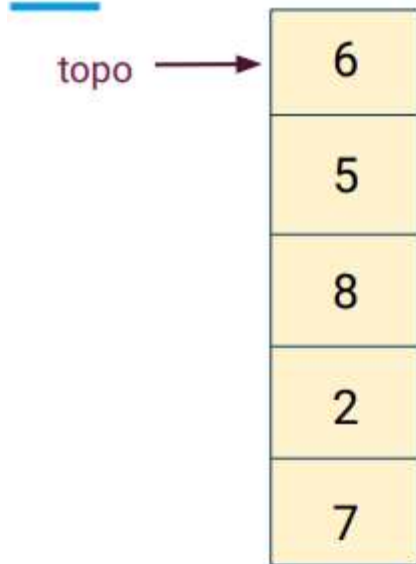


```
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);
    pilha.push(5);
    pilha.push(6);

}
```

Stack STL - Métodos

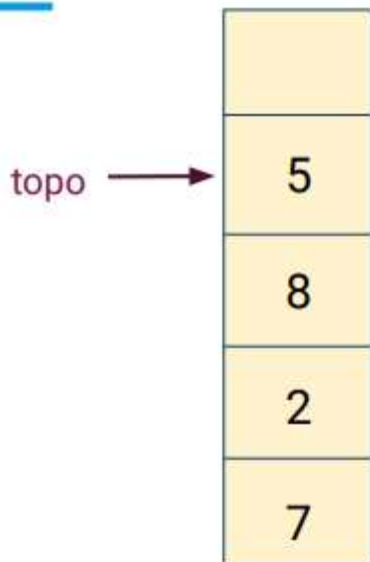


```
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);
    pilha.push(5);
    pilha.push(6);
    cout << "Topo: " << pilha.top() << endl;
```

Topo: 6

Stack STL - Métodos

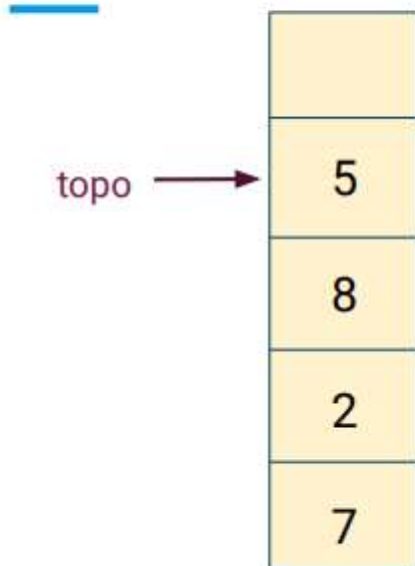


```
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);
    pilha.push(5);
    pilha.push(6);
    cout << "Topo: " << pilha.top() << endl;
    pilha.pop();
}
```

Topo: 6

Stack STL - Métodos



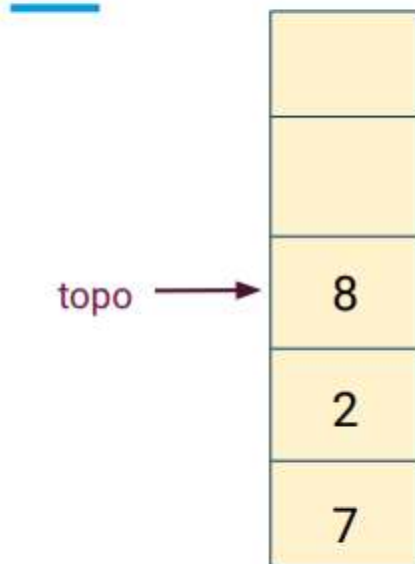
```
#include <iostream>
#include <stack>
using namespace std;
```

```
int main() {
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);
    pilha.push(5);
    pilha.push(6);
    cout << "Topo: " << pilha.top() << endl;
    pilha.pop();
    cout << "Topo: " << pilha.top() << endl;
}
```

Topo: 6

Topo: 5

Stack STL - Métodos



```
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> pilha;
    pilha.push(7);
    pilha.push(2);
    pilha.push(8);
    pilha.push(5);
    pilha.push(6);
    cout << "Topo: " << pilha.top() << endl;
    pilha.pop();
    cout << "Topo: " << pilha.top() << endl;
    pilha.pop();
    cout << "Topo: " << pilha.top() << endl;
}
```

Topo: 6
Topo: 5
Topo: 8

QUEUE (FILAS com STL)

- Incluir biblioteca `<queue>`:

`#include <queue>`

- Criar um objeto do tipo da classe `<queue>`:

Exemplo:

`queue<int> fila;`

QUEUE (FILAS com STL)

- Inserir um elemento no final da fila

push()

- Remover elemento da fila. Não retorna o elemento removido:

pop()

- Obter elemento na frente da fila. Retorna o elemento sem removê-lo:

front()

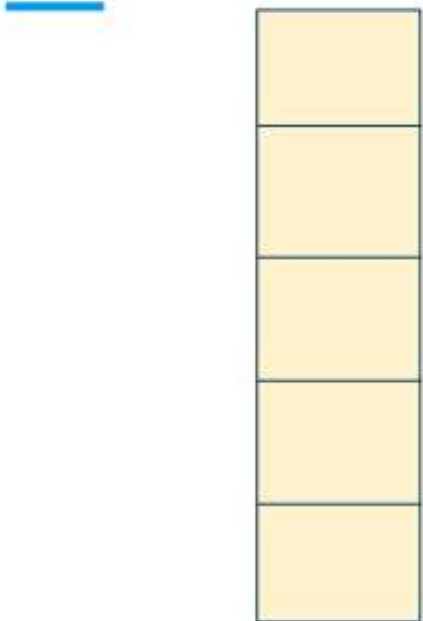
- Verifica se a fila está vazia

empty()

- Tamanho atual da fila

size()

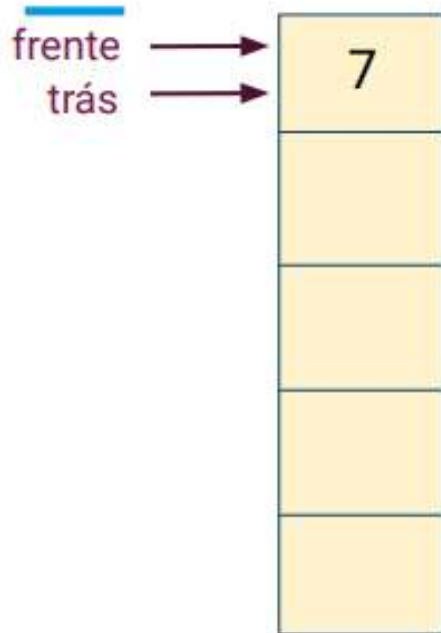
QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;

int main(){
    queue<int> fila;
```

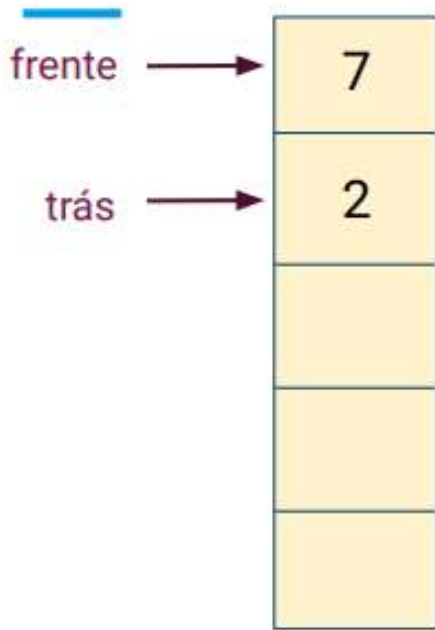
QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;
    fila.push(7);
}
```

QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;
    fila.push(7);
    fila.push(2);
}
```

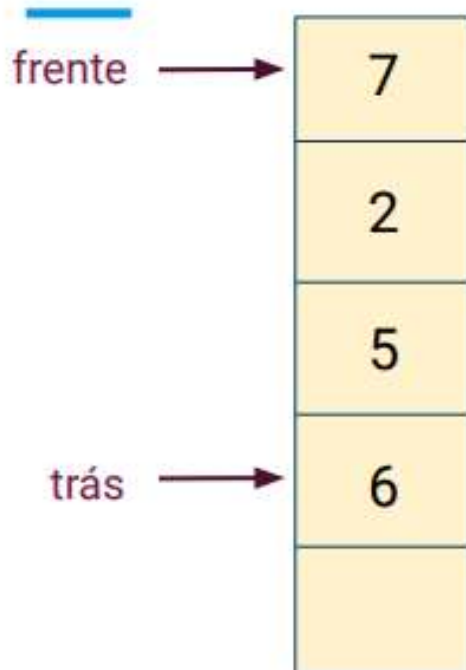
QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
}
```


QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;

int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
```

QUEUE STL - Métodos

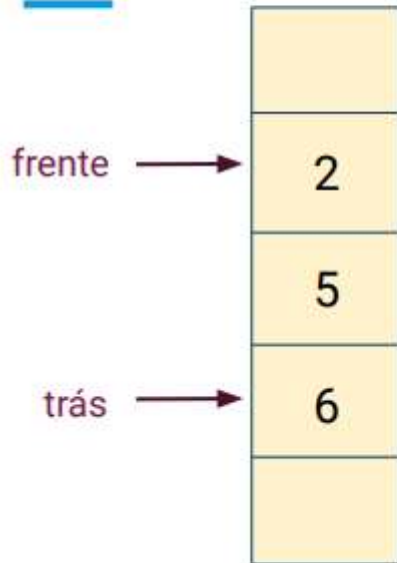


```
#include <iostream>
#include <queue>
using namespace std;
```

```
int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
    cout << "Frente: " << fila.front() << endl;
    cout << "Trás: " << fila.back() << "\n" << endl;
```

```
Frente: 7
Trás: 6
```

QUEUE STL - Métodos

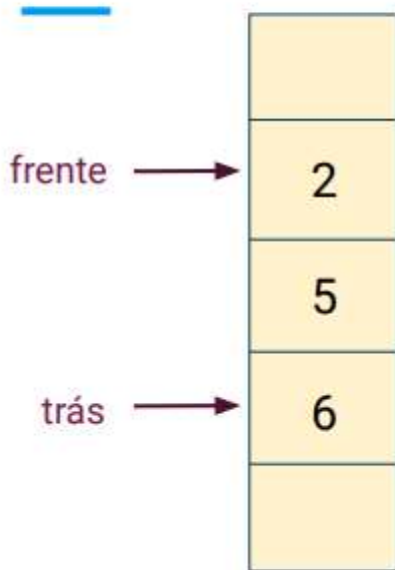


```
#include <iostream>
#include <queue>
using namespace std;
```

```
int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
    cout << "Frente: " << fila.front() << endl;
    cout << "Trás: " << fila.back() << "\n" << endl;
    fila.pop();
```

```
Frente: 7
Trás: 6
```

QUEUE STL - Métodos



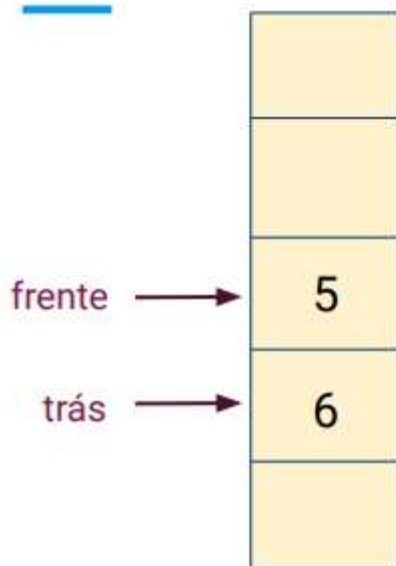
```
#include <iostream>
#include <queue>
using namespace std;
```

```
int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
    cout << "Frente: " << fila.front() << endl;
    cout << "Trás: " << fila.back() << "\n" << endl;
    fila.pop();
    cout << "Frente: " << fila.front() << endl;
    cout << "Final: " << fila.back() << "\n" << endl;
```

```
Frente: 7
Trás: 6
```

```
Frente: 2
Trás: 6
```

QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;
```

```
int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
    cout << "Frente: " << fila.front() << endl;
    cout << "Trás: " << fila.back() << "\n" << endl;
    fila.pop();
    cout << "Frente: " << fila.front() << endl;
    cout << "Final: " << fila.back() << "\n" << endl;
    fila.pop();
}
```

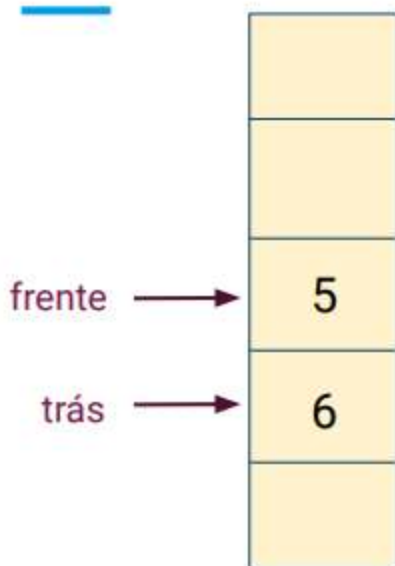
Frente: 7

Trás: 6

Frente: 2

Trás: 6

QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;
```

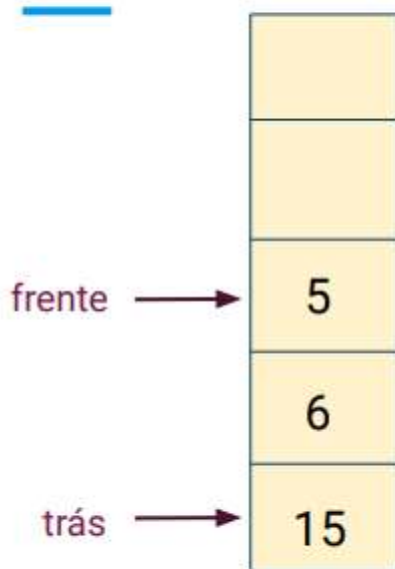
```
int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
    cout << "Frente: " << fila.front() << endl;
    cout << "Trás: " << fila.back() << "\n" << endl;
    fila.pop();
    cout << "Frente: " << fila.front() << endl;
    cout << "Final: " << fila.back() << "\n" << endl;
    fila.pop();
    cout << "Frente: " << fila.front() << endl;
    cout << "Final: " << fila.back() << "\n" << endl;
```

Frente: 7
Trás: 6

Frente: 2
Trás: 6

Frente: 5
Trás: 6

QUEUE STL - Métodos



```
#include <iostream>
#include <queue>
using namespace std;
```

```
int main(){
    queue<int> fila;
    fila.push(7);
    fila.push(2);
    fila.push(5);
    fila.push(6);
    cout << "Frente: " << fila.front() << endl;
    cout << "Trás: " << fila.back() << "\n" << endl;
    fila.pop();
    cout << "Frente: " << fila.front() << endl;
    cout << "Final: " << fila.back() << "\n" << endl;
    fila.pop();
    cout << "Frente: " << fila.front() << endl;
    cout << "Final: " << fila.back() << "\n" << endl;
    fila.push(15);
```

Frente: 7

Trás: 6

Frente: 2

Trás: 6

Frente: 5

Trás: 6

Atividade

Versão usando *stl* das questões:

Questão 11. Desenvolva os procedimentos de **entrar e sair de uma fila** a partir dos procedimentos **empilhar, desempilhar, topo, pilhaCheia** e **pilhaVazia** de uma pilha. **Utilize 2 (duas) pilhas.**

Questão 12. Desenvolva os procedimentos de **empilhar e desempilhar de uma pilha** a partir dos procedimentos **entrar, sair, primeiro, ultimo, filaCheia** e **filaVazia** de uma fila. **Utilize 2 (duas) filas.**