


## 2022\_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA\_TN - METATURMA

[PAINEL](#) > [MINHAS TURMAS](#) > [2022\\_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA\\_TN - METATURMA](#) > [GERAL](#)  
> [L02E06 - FILA EXCEPCIONAL \(3,0 PTS\)](#)

[Descrição](#)[Enviar](#)[Editar](#)[Visualizar envios](#)

### L02E06 - Fila Excepcional (3,0 pts)

 **Data de entrega:** sexta, 1 Jul 2022, 23:59

 **Arquivos requeridos:** main.cpp, LimitedQueue.hpp, LimitedQueue.cpp, Exceptions.hpp ([Baixar](#))

**Tipo de trabalho:**  Trabalho individual

O objetivo desse exercício é praticar a criação e tratamento de **Exceções**.

Você deve fazer a implementação do TAD **LimitedQueue**, cuja especificação já está no arquivo **LimitedQueue.hpp**. O TAD representa uma Fila (itens inseridos no final e removidos do início) de tamanho limitado. A Fila irá armazenar apenas números inteiros. Você é livre para adicionar quaisquer outros atributos ou métodos auxiliares que julgar necessário.

Esse TAD utilizará 4 exceções específicas que deverão ser implementadas por você:

- **InvalidSizeException**
- **QueueFullException**
- **QueueEmptyException**
- **ItemNotFoundException**

As exceções devem ser todas implementadas no arquivo **Exceptions.hpp** (não há necessidade de criar um arquivo .cpp). As exceções deverão ser tipos individuais e **não devem herdar de std::exception**. Além disso, todas devem possuir um **atributo público** do tipo string chamado 'name', cujo valor é o próprio nome da exceção.

Por fim, você deve implementar o arquivo **main.cpp** e adicionar toda a parte de entrada/saída que será responsável por manipular os seguintes comandos:

- **'i max\_size'**: comando para inicializar a fila com um certo tamanho máximo.
- **'p k'**: comando para inserir o inteiro **k** no final da fila.
- **'o'**: comando para remover o primeiro elemento da fila.
- **'f'**: comando para imprimir o primeiro elemento da fila.
- **'l'**: comando para imprimir o último elemento da fila.
- **'m'**: comando para imprimir toda a fila.
- **'d k'**: comando que busca o inteiro **k** na fila e imprime a sua posição.
- **'b'**: deve chamar a função 'avaliacao\_basica()' implementada no arquivo "avaliacao\_basica\_excecoes.hpp" (já incluído no main.cpp). Essa função faz uma avaliação do código (não apenas dos resultados).

Para cada comando chamado deve-se realizar o correto tratamento de exceções. Ou seja, se ao executar um comando uma exceção for lançada, o programa não deve interromper a execução, mas a exceção deve ser capturada e a seguinte mensagem impressa: 'Excecao: *ExceptionName*'. Observe que o tratamento não deve ser feito em LimitedQueue, mas o TAD deve apenas lançar a exceção.

Abaixo são apresentados dois exemplos de entrada/saída:

```
input =
i 5
p 1
p 2
p 3
m
output =
```

```
1
2
3

input =
i 5
m
output =
Excecao: QueueEmptyException
```

**Dica 1:**

Você pode utilizar a estrutura **list** da **STL** para implementar a fila (salvar/manipular os elementos internamente).

**Dica 2:**

O código da avaliação básica pode ser copiado [aqui](#), caso você queira depurar algo localmente.

**Referências:**

<https://www.cplusplus.com/reference/exception/exception/>

## Arquivos requeridos

### main.cpp

```
1 // NÃO ALTERE ESSA LINHA
2 #include "avaliacao_basica_excecoes.hpp"
3
4 int main() {
5
6
7     //
8     // Adicione seu código aqui e faça as demais alterações necessárias
9     //
10
11
12     return 0;
13 }
```

### LimitedQueue.hpp

```
1 #ifndef LIMITEDQUEUE_H
2 #define LIMITEDQUEUE_H
3
4 // Fila de números inteiros de tamanho limitado
5 class LimitedQueue {
6
7     public:
8
9         // Construtor da fila.
10        // Lança a exceção InvalidSizeException se max_size <= 0.
11        // O tamanho máximo da fila deve ser acessível por um método 'get_max_size()'.
12        LimitedQueue(int max_size);
13
14        // Insere um elemento no final da fila.
15        // Lança a exceção QueueFullException se a fila estiver cheia.
16        void push_back(int k);
17
18        // Retira o elemento do início da fila.
19        // Lança a exceção QueueEmptyException se a fila estiver vazia.
20        void pop_front();
21
22        // Retorna (sem retirar) o primeiro elemento (início da fila).
23        // Lança uma exceção QueueEmptyException se a fila estiver vazia.
24        int front();
25
26        // Retorna (sem retirar) o último elemento (final da fila).
27        // Lança a exceção QueueEmptyException se a fila estiver vazia.
28        int back();
29
30        // Verifica se um elemento está na fila e retorna a posição dele (começando de 0).
31        // Lança a exceção QueueEmptyException se a fila estiver vazia.
32        // Lança a exceção ItemNotFoundException se o elemento não estiver na fila.
33        int find(int k);
34
35        // Imprime todos os elementos da fila em ordem, sendo um por linha.
36        // Lança a exceção QueueEmptyException se a fila estiver vazia.
37        void print();
38
39 };
40
41 #endif
42
```

### LimitedQueue.cpp

### Exceptions.hpp

◀ [L02E05 - Revisão de código e Rafatoração \(4,0 pts\)](#)

Seguir para...

[L02E07 - Diário classe \(2,0 pts\)](#) ▶