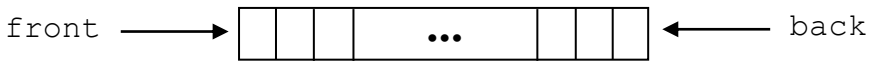
EEL670 - Linguagens de Programação
Departamento de Engenharia Eletrônica e de Computação - DEL-Poli/UFRJ
Prof.: Miguel Elias Mitre Campista
== Laboratório 10

---

1) Lista de exercícios: Resolva os exercícios abaixo como se pede.

a) Escreva uma classe template `ListaRestrita` que restringe as possíveis operações em um array a apenas as de inserção e remoção de elementos nas extremidades da estrutura. Isso quer dizer que a classe template `ListaRestrita` oferece apenas os métodos `push_front` e `push_back` para inserções no início e no final da estrutura, respectivamente, e os métodos `pop_front` e `pop_back` para remoção de elementos no início e no final da estrutura, respectivamente, como visto na figura abaixo.



A classe armazena os elementos em uma array privado alocado dinamicamente no construtor. Lembre-se de implementar um destrutor para liberação da memória. Uma cópia dos elementos no início e no fim da estrutura é retornada a partir dos métodos `front` e `back`.

O índice do elemento no fim da estrutura é armazenado no atributo privado `top`, assim como o tamanho máximo da estrutura, que é armazenado no atributo privado `maxsize`. Note que quando a lista estiver vazia, `top == -1` é verdadeiro; e quando ela estiver cheia, `top == (maxsize - 1)` é verdadeiro.

Faça uma função principal que contemple todos os métodos da classe template `ListaRestrita`.

b) Continuando a Questão 1.a, reescreva o programa anterior usando herança da classe `vector`. Isso irá dispensar a necessidade de implementação do array. Além do mais, note que os métodos `front` e `back` podem ter problemas, caso a estrutura esteja vazia. Dispare uma exceção do tipo `EstruturaVaziaException` derivada da classe `exception`, caso isso ocorra. A exceção deverá ser tratada na função principal.

Faça uma função principal que contemple todos os métodos da classe template `ListaRestrita`.

== Respostas da Lista de Exercícios

1)

a)

```
/*****************************************************************************/
/***************************** Programa Principal ****************************/
#include <iostream>
#include <cstdlib>
#include <ctime>

#include "lista-restrita.h"

using namespace std;

/* Programa do Laboratório 10:
   Programa de uma class template Lista
   Autor: Miguel Campista */


int main() {
    ListaRestrita <int> lista (5);
    srand (time (0));

    /* Inserções ao final da estrutura */
    int numero = rand () % 10;
    while (lista.push_back (numero)) {
           cout << "Inseri: " << numero << endl;
           numero = rand () % 10;
    }

    cout << endl;

    cout << "Elemento no início: " << lista.front()
         << "\nElemento no final: " << lista.back()
         << endl;

    cout << endl;

    while (lista.pop_back (numero)) {
           cout << "Removi: " << numero << endl;
    }

    cout << "\n*** Agora ao contrário\n" << endl;

    /* Inserções no início da estrutura */
    numero = rand () % 10;
    while (lista.push_front (numero)) {
           cout << "Inseri: " << numero << endl;
           numero = rand () % 10;
    }

    cout << endl;

    cout << "Elemento no início: " << lista.front()
         << "\nElemento no final: " << lista.back()
         << endl;

    cout << endl;

    while (lista.pop_front (numero)) {
           cout << "Removi: " << numero << endl;
    }

    return 0;
}

/*****************************************************************************/
/************************ Arquivo lista-restrita.h ***************************/
#include <iostream>

using namespace std;
```

```
#ifndef LISTA_RESTRITA_H
#define LISTA_RESTRITA_H

template <class T>
class ListaRestrita {
    public:
            ListaRestrita (int sz):
                    maxsize (sz), top (-1), ptr (new T [sz]) {}

            ~ListaRestrita () { delete [] ptr; }

            bool push_back (T v) {
                    if (!estaCheia ()) {
                            ptr [++top] = v;
                            return true;
                    }
                    return false;
            }

            bool push_front (T v) {
                    if (!estaCheia ()) {
                            int idx = ++top;
                            while (idx > 0)
                                    ptr [idx] = ptr [--idx];
                            ptr [0] = v;
                            return true;
                    }
                    return false;
            }

            bool pop_back (T& v) {
                    if (!estaVazia ()) {
                            v = ptr [top--];
                            return true;
                    }
                    return false;
            }

            bool pop_front (T& v) {
                    if (!estaVazia ()) {
                            int idx = 1;
                            v = ptr [0];
                            while (idx <= top) {
                                    ptr [idx - 1] = ptr [idx++];
                    }
                    top--;
                    return true;
                    }
                    return false;
            }

            T front () { return ptr [0]; }
            T back () { return ptr [top]; }

            bool estaCheia () { return top == maxsize - 1; }
            bool estaVazia () { return top == - 1; }

    private:
            T *ptr;
            int top, maxsize;
};

#endif

/*****************************************************************************/
```

b)

```
/*****************************************************************************/
/***************************** Programa Principal ****************************/
#include <iostream>
#include <cstdlib>
#include <ctime>

#include "lista-restrita.h"
```

```
#include "estrutura-vazia-exception.h"

using namespace std;

/* Programa do Laboratório 10:
   Programa de uma class template Lista
   Autor: Miguel Campista */


int main() {
    ListaRestrita <int> lista (5);
    srand (time (0));

    /* Inserções ao final da estrutura */
    int numero = rand () % 10;
    while (lista.push_back (numero)) {
            cout << "Inseri: " << numero << endl;
            numero = rand () % 10;
    }

    cout << endl;

    try {
            cout << "Elemento no início: " << lista.front()
                    << "\nElemento no final: " << lista.back()
                    << endl;
    } catch (EstruturaVaziaException &except) {
            cout << except.what() << endl;
    }

    cout << endl;

    while (lista.pop_back (numero)) {
            cout << "Removi: " << numero << endl;
    }

    cout << "\n*** Agora ao contrário\n" << endl;

    /* Inserções no início da estrutura */
    numero = rand () % 10;
    while (lista.push_front (numero)) {
            cout << "Inseri: " << numero << endl;
            numero = rand () % 10;
    }

    cout << endl;

    try {
            cout << "Elemento no início: " << lista.front()
                    << "\nElemento no final: " << lista.back()
                    << endl;
    } catch (EstruturaVaziaException &except) {
            cout << except.what() << endl;
    }

    cout << endl;

    while (lista.pop_front (numero)) {
            cout << "Removi: " << numero << endl;
    }

    /* Só para provocar a exceção... */
    try {
            cout << "Elemento no início: " << lista.front()
                    << "\nElemento no final: " << lista.back()
                    << endl;
    } catch (EstruturaVaziaException &except) {
            cout << "\n***" << except.what() << endl;
    }

    return 0;
}

/*************************************************************************/
/************************ Arquivo lista-restrita.h ************************/
#include <iostream>
#include <vector>
```

```
#include "estrutura-vazia-exception.h"

using namespace std;

#ifndef LISTA_RESTRITA_H
#define LISTA_RESTRITA_H

template <class T>
class ListaRestrita : public vector <T> {
    public:
            ListaRestrita (int sz):
                    maxsize (sz), top (-1), vector <T> (sz) {}

            bool push_back (T v) {
                    if (!estaCheia ()) {
                            vector <T>::at (++top) = v;
                            return true;
                    }
                    return false;
            }

            bool push_front (T v) {
                    if (!estaCheia ()) {
                            int idx = ++top;
                            while (idx > 0)
                                    vector <T>::at (idx) = vector <T>::at (--idx);
                            vector <T>::at (0) = v;
                            return true;
                    }
                    return false;
            }

            bool pop_back (T& v) {
                    if (!estaVazia ()) {
                            v = vector <T>::at (top--);
                            return true;
                    }
                    return false;
            }

            bool pop_front (T& v) {
                    if (!estaVazia ()) {
                            int idx = 1;
                            v = vector <T>::at (0);
                            while (idx <= top) {
                                    vector <T>::at (idx - 1) = vector <T>::at (idx++);
                    }
                    top--;
                    return true;
                    }
                    return false;
            }

            T front () {
                    if (estaVazia())
                            throw EstruturaVaziaException ();
                    return vector <T>::at (0);
            }

            T back () {
                    if (estaVazia())
                            throw EstruturaVaziaException ();
                    return vector <T>::at (top);
            }

            bool estaCheia () { return top == vector <T>::size() - 1; }
            bool estaVazia () { return top == - 1; }

    private:
            int top, maxsize;
};

#endif

/*************************************************************************/
/****************** Arquivo estrutura-vazia-exception.h ******************/
```

```
#include <iostream>
#include <stdexcept>

using namespace std;

#ifndef ESTRUTURA_VAZIA_EXCEPTION_H
#define ESTRUTURA_VAZIA_EXCEPTION_H

class EstruturaVaziaException : public exception {
    public:
            virtual const char * what () const throw ();
};

#endif

/*************************************************************************/
/****************** Arquivo estrutura-vazia-exception.cpp ******************/
#include "estrutura-vazia-exception.h"

const char * EstruturaVaziaException::what () const throw () {
    return "Estrutura Vazia Exception!";
}
/*************************************************************************/
```