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

== Laboratório 8 ==

1. **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. A classe armazena os elementos em uma array privado alocado dinamicamente no construtor. Lembre-se de implementar um destrutor para liberação da memória. O índice do elemento do início e do fim da estrutura é armazenado nos atributos privados `front` e `back`, assim como o tamanho máximo da estrutura, que é armazenado no atributo privado `maxsize`.**

   Note que quando a lista estiver vazia `front == back` e quando ela está cheia, vazia, `front == (maxsize - 1)`.

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

2. **Reescreva o programa anterior usando herança da classe `vector`. Isso irá dispensar a necessidade de implementação do array.**

3. **Reescreva o programa da Questão 1 substituindo o array por uma lista encadeada. Note que, dessa forma, não será possível armazenar exatamente o elemento passado para as operações de inserção. Ao invés disso, será necessário utilizar uma estrutura de dados transparente para o usuário, que oferece ponteiro para o encadeamento da estrutura.**

4. **Escreva uma classe template `Arvore` para armazenamento de elementos em uma árvore binária. A classe oferece apenas operações de inserção e exibição de todos os elementos. Assim como na Questão 3, será necessário a criação de uma estrutura de dados para armazenamento do elemento e também dois ponteiros para a sub-árvore da direita e da esquerda.**

== Respostas ==

1.

```
/*********************************************************************/
/*************************** Programa Principal **************************/

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

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

```
            ~ListaRestrita () { delete [] ptr; }

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

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

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

            bool estaCheia () { return front == maxsize - 1; }
            bool estaVazia () { return front == - 1; }
      private:
              T *ptr;
              int front, maxsize;
};

int main() {
    ListaRestrita <int> lista (10);
    int numero = rand () % 10;
    srand (time (0));

    while (lista.push_back (numero)) {
          cout << "Inseri: " << numero << endl;
          numero = rand () % 10;
    }

    cout << endl;

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

    return 0;
}

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

4.

```
/*****************************************************************************/
/************************** Programa Principal ***************************/

#include <iostream>
#include <ctime>
```

```
#include <cstdlib>

using namespace std;

template <class T> class Arvore

template <class T>
struct Elemento {
       friend class Arvore <T>;
       public:
              Elemento (int v): valor (v), esquerda (NULL), direita (NULL) {}
       private:
               T valor;
               Elemento * esquerda;
               Elemento * direita;
};

template <class T>
class Arvore {
      public:
      Arvore (): raiz (NULL) {}

      Elemento <T> * insere (T v, Elemento <T> * atual) {
           Elemento <T> * e = new Elemento <T> (v);

           if (atual == NULL) {
                  raiz = e;
           } else {
                  if (e->valor >= atual->valor) {
                           if (atual->esquerda == NULL)
                                  atual->esquerda = e;
                           else insere (v, atual->esquerda);
                  } else {
                           if (atual->direita == NULL)
                                  atual->direita = e;
                           else insere (v, atual->direita);
                  }
           }
           return e;
      }

      void exibe (Elemento <T> * atual) {
           if (atual == NULL)
                 return;

           cout << atual->valor << endl;

           exibe (atual->esquerda);

           exibe (atual->direita);
      }

      Elemento <T> * getRaiz () { return raiz; }

      private:
              Elemento <T> *raiz;
};

int main()
{
    Arvore <int> arvore;

    srand (time (0));

    for (int i = 0; i < 10; i++) {
        int v = rand () % 10;
        cout << "Insere " << v << endl;
        arvore.insere (v, arvore.getRaiz ());
    }

    arvore.exibe (arvore.getRaiz ());

    return 0;
}

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