```
#ifndef LISTAESTATICA_H_INCLUDED
#define LISTAESTATICA_H_INCLUDED

#include <iostream>

using namespace std;

const
int tamanho =  100;

struct ListaEstatica
{
    int cardinalidade;
    int elementos [tamanho];
};

ListaEstatica cria ()
{
    ListaEstatica lista;

    lista.cardinalidade = 0;
    return lista;
}

bool ehVazia (ListaEstatica lista)
{
    return lista.cardinalidade == 0;
}

bool temEspaco (ListaEstatica lista)
{
    return lista.cardinalidade < tamanho;
}

int numeroDeElementos (ListaEstatica lista)
{
    return lista.cardinalidade;
}

bool existeElemento (ListaEstatica lista, int elemento)
{
    bool encontrou = false;
    int i = 0;
    while ((i < lista.cardinalidade) && (encontrou == false))
    {
        if (lista.elementos[i] == elemento)
            encontrou = true;
        i++;
    }
    return encontrou;
}

bool existePosicao (ListaEstatica lista, int posicao)
{
    return ((posicao >= 1) && (posicao <= lista.cardinalidade));
}

int umElemento (ListaEstatica lista, int posicao)
{
    return lista.elementos[posicao - 1];
}

int posicao (ListaEstatica lista, int elemento)
{
    int i = 0;
    while (lista.elementos[i] != elemento)
```

```
        i++;
    return i + 1;
}

void insere (ListaEstatica &lista, int elemento, int posicao)
{
    for (int i = lista.cardinalidade; i >= posicao; i--)
        lista.elementos[i] = lista.elementos[i - 1];
    lista.elementos[posicao - 1] = elemento;
    lista.cardinalidade++;
}

void retira (ListaEstatica &lista, int posicao)
{
    for (int i = posicao; i < lista.cardinalidade; i++)
        lista.elementos[i - 1] = lista.elementos[i];
    lista.cardinalidade--;
}

void mostra (ListaEstatica lista)
{
    for (int i = 0; i < lista.cardinalidade; i++)
        cout << lista.elementos[i] << "  ";
}

#endif // LISTAESTATICA_H_INCLUDED
```