listaEstatica

October 22, 2021

1 Listas Lineares

- dias da semana: [segunda, terça, quarta, quinta, sexta, sábado, domingo]
- estações do ano: [primavera, verão, outono, inverno]
- polaridade: [positivo, negativo, neutro]

Em uma lista elementos podem ser inseridos, removidos ou substituídos em qualquer posição.

1.1 Implementações:

- estática (contígua)
- dinâmica (encadeada)

1.2 Definição

Ao inserir um elemento em uma dada posição, os elementos seguintes são deslocados para posições posteriores.

Exemplo:

Seja uma lista L = [4, 9, 3] contendo n=3 elementos: - a1 = 4 - a2 = 9 - a3 = 3

Se inserirmos um elemento 5 na 2ª posição da lista teremos L = [4, 5, 9, 3] - a1 = 4 - a2 = 5 - a3 = 9 - a4 = 3

Ao remover um elemento em uma dada posição, os elementos seguintes são deslocados para posições anteriores.

Se removermos o terceiro elemento da nossa lista teremos L = [4, 5, 3] - a1 = 4 - a2 = 5 - a3 = 3

1.3 Operações Básicas

1.3.1 Criação

Lista criada e iniciada vazia

1.3.2 Status: vazia

Verifica se a lista está vazia

1.3.3 Status: cheia

Verifica se a lista esta cheia

1.3.4 Inserção

Inserção de elemento em determinada posição.

Esta posição deve ser um valor entre 1 (primeiro elemento da lista) e contador + 1 (após o último elemento da lista).

As entradas posteriores ao elemento inserido devem ter suas posições incrementadas em uma unidade.

1.3.5 Remoção

Remoção de elemento em determinada posição.

Esta posição deve ser um valor entre 1 (primeiro elemento da lista) e contador (último elemento da lista).

As entradas posteriores ao elemento removido devem ter suas posições decrementadas em uma unidade.

1.3.6 Retorna

Retorna elemento armazenado em determinada posição.

Esta posição deve ser um valor entre 1 (primeiro elemento da lista) e contador (último elemento da lista).

1.3.7 Substitui

Substitui elemento armazenado em determinada posição.

Esta posição deve ser um valor entre 1 (primeiro elemento da lista) e contador (último elemento da lista).

1.4 Outras operações

1.4.1 Tamanho

Retorna o tamanho atual da lista.

1.4.2 Esvaziar

Remove todos os elementos da lista.

1.5 Implementação

[1]:
```
#include <iostream>
#include <string>
using namespace std;

#define MAX 5
```

1.6 Definição da classe

[2]:
```
// Lista.h
class Lista {
    private:
        int contador;
        int capacidade = MAX;
        int elementos[MAX];
    public:
        Lista();
        bool vazia();
        bool cheia();
        bool inserir(int posicao, int x);
        bool remover(int posicao, int &x);
        bool retornar(int posicao, int &x);
        bool substituir(int posicao, int x);
        bool esvaziar();
        int tamanho();
        string listar();
};
```

[ ]:
```

```

1.6.1 Método Construtor

[4]:
```
// Lista.cpp
Lista::Lista() {
    contador = 0;
}
```

[5]:
```
Lista minhaLista;
```

1.6.2 Verifica se lista está vazia

[85]:
```
bool Lista::vazia() {
    return contador == 0;
}
```

[86]:
```
if (minhaLista.vazia()) {
    cout << "Está vazia!";
}
```

```
Está vazia!
```

1.6.3 Verifica se lista está cheia

[87]:
```
bool Lista::cheia() {
    return contador == capacidade;
}
```

[88]:
```
if (! minhaLista.cheia()) {
    cout << "Não está cheia!";
}
```

```
Não está cheia!
```

1.6.4 Inserção elemento na fila

[70]:
```
bool Lista::inserir(int posicao, int x) {


}
```

```
input_line_81:2:1: error: control reaches end of non-

void function [-Werror,-Wreturn-type]
}
^
```

[71]:
```
bool Lista::remover(int posicao, int &x) {
    if (vazia())
        return false;

}
```

```
input_line_82:4:1: warning: control may reach end of

non-void function [-Wreturn-type]
}
^
```

2 Tarefa D

Para a classe que implementa uma lista estática, desenvolva os métodos: inserir, remover, retornar e substituir.

As instruções estão no arquivo Lista.cpp

Baixe o arquivo, implemente os métodos onde indicado, compacte e envie novamente.

[72]:
```
// exclusão física sempre do primeiro elemento
bool Lista::esvaziar() {
    int x;
    if (vazia())
        return false;

    while(remover(1, x)) {
        cout << "removido: " << x;
    }
    return true;
}
```

[73]:
```
// exclusão física sempre do último elemento (menos onerosa)
bool Lista::esvaziar() {
    int x;
    if (vazia())
        return false;

    while(remover(contador, x));
    return true;
}
```

```
input_line_84:2:13: error: redefinition of
'esvaziar'
bool Lista::esvaziar() {
            ^

input_line_83:2:13: note: previous definition is
here
bool Lista::esvaziar() {
            ^
```

```
Interpreter Error:
```

[74]:
```
// exclusão lógica (muito menos onerosa)
bool Lista::esvaziar() {
    int x;
    if (vazia())
        return false;
    contador = 0;
    return true;
}
```

```
input_line_85:2:13: error: redefinition of
'esvaziar'
```

```
bool Lista::esvaziar() {
            ^

input_line_83:2:13: note: previous definition is
here
bool Lista::esvaziar() {
            ^
```

```
Interpreter Error:
```

[ ]:

[ ]:

[75]:
```
if (! minhaLista.esvaziar()) {
    cout << "Lista já está vazia!";
}
```

```
Lista já está vazia!
```

[89]:
```
int Lista::tamanho() {
    return contador;
}
```

[90]:
```
if (minhaLista.tamanho() == 0) {
    cout << "Lista está vazia!";
}
```

```
Lista está vazia!
```

[ ]:

[ ]:
```
string Lista::listar() {
    string aux = "";
    if (vazia())
        return aux;

    for (int i = 0 ; i < contador ; i++) {
        aux = aux + to_string(elementos[i]) + "\n";
    }
    return aux;
}
```

[ ]:
```
// versão otimizada
string Lista::listar() {
    string aux = "";
```

```
    for (int i = 0 ; i < contador ; i++) {
        aux = aux + to_string(elementos[i]) + "\n";
    }
    return aux;
}
```

[ ]:

[ ]:

[78]:
```
string Lista::listar() {
    int valor = 0;
    string aux = "";

    elementos[0] = 3;
    elementos[1] = 2;
    elementos[2] = 1;
    contador = 3;

    for (int i = 0 ; i < contador ; i++) {
        aux = aux + to_string(elementos[i]) + "\n";
    }
    return aux;
}
```

[ ]:

[79]:
```
cout << minhaLista.listar();
```

```
3
2
1
```

[94]:
```
string nome, cidade;
nome = "fatec";
cidade = "ribeirão";
nome = nome + " " + cidade;

cout << nome;
```

```
fatec ribeirão
```

[95]:
```
cout << nome[2];
```

```
t
```