|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include <list> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| using namespace std; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bool par(const int& n) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return (n % 2 == 0); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bool comparar\_tamanho(const string& s1, const string& s2) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| // se o tamanho da primeira string deve vir antes, |
|  |

|  |
| --- |
| // então retorna true |
|  |

|  |
| --- |
| if(s1.length() < s2.length()) |
|  |

|  |
| --- |
| return true; |
|  |

|  |
| --- |
| return false; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int main() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| list<int> l1; // lista de inteiros vazia |
|  |

|  |
| --- |
| list<int> l2(3, 10); // três inteiros com o valor 10 |
|  |

|  |
| --- |
| list<int>::iterator it; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // inserindo elementos em l1 - push\_back e push\_front |
|  |

|  |
| --- |
| l1.push\_back(10); // l1: 10 |
|  |

|  |
| --- |
| l1.push\_front(20); // l1: 20, 10 |
|  |

|  |
| --- |
| l1.push\_back(30); // l1: 20, 10, 30 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "Mostrando os elementos de l1:\n"; |
|  |

|  |
| --- |
| for(it = l1.begin(); it != l1.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\nMostrando os elementos de l2:\n"; |
|  |

|  |
| --- |
| for(it = l2.begin(); it != l2.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // acessando o primeiro e último elemento de l1 - front e back |
|  |

|  |
| --- |
| cout << "\nprimeiro elemento de l1: " << l1.front(); |
|  |

|  |
| --- |
| cout << "\nultimo elemento de l1: " << l1.back() << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\nTamanho de l1: " << l1.size() << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // l1: 20, 10, 30 |
|  |

|  |
| --- |
| // removendo o primeiro elemento de l1 |
|  |

|  |
| --- |
| l1.pop\_front(); // remove o 20 |
|  |

|  |
| --- |
| // removendo o último elemento de l1 |
|  |

|  |
| --- |
| l1.pop\_back(); // remove o 30 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\nMostrando novamente os elementos de l1:\n"; |
|  |

|  |
| --- |
| for(it = l1.begin(); it != l1.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // removendo todos os elementos de l2 |
|  |

|  |
| --- |
| while(!l2.empty()) |
|  |

|  |
| --- |
| l2.pop\_front(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\nNovo tamanho de l2: " << l2.size() << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // atribuindo elementos a l2 - função assign |
|  |

|  |
| --- |
| int vet[] = {1, 2, 3, 4}; |
|  |

|  |
| --- |
| l2.assign(vet, vet + 4); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\nMostrando novos elementos de l2:\n"; |
|  |

|  |
| --- |
| for(it = l2.begin(); it != l2.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // inserindo um elemento em l2 - função insert |
|  |

|  |
| --- |
| l2.insert(l2.begin(), 100); |
|  |

|  |
| --- |
| cout << "\nMostrando novamente os elementos de l2:\n"; |
|  |

|  |
| --- |
| for(it = l2.begin(); it != l2.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| it = l2.begin(); |
|  |

|  |
| --- |
| it++; // aponta para o segundo |
|  |

|  |
| --- |
| l2.insert(it, 200); |
|  |

|  |
| --- |
| cout << "\nElementos de l2:\n"; |
|  |

|  |
| --- |
| for(it = l2.begin(); it != l2.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // inserindo duas vezes o valor 50 na primeira posição |
|  |

|  |
| --- |
| l2.insert(l2.begin(), 2, 50); |
|  |

|  |
| --- |
| cout << "\nElementos de l2:\n"; |
|  |

|  |
| --- |
| for(it = l2.begin(); it != l2.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // apagando os dois primeiros números |
|  |

|  |
| --- |
| list<int>::iterator it2 = l2.begin(); |
|  |

|  |
| --- |
| it2++; |
|  |

|  |
| --- |
| it2++; |
|  |

|  |
| --- |
| l2.erase(l2.begin(), it2); |
|  |

|  |
| --- |
| cout << "\nElementos de l2:\n"; |
|  |

|  |
| --- |
| for(it = l2.begin(); it != l2.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // remover todos os elementos - função clear |
|  |

|  |
| --- |
| l2.clear(); |
|  |

|  |
| --- |
| cout << "\nTamanho de l2: " << l2.size() << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // função splice - transfere elementos de uma lista para outra |
|  |

|  |
| --- |
| list<int> lista1(2, 10), lista2(2, 20); |
|  |

|  |
| --- |
| it = lista1.begin(); |
|  |

|  |
| --- |
| // transfere elementos de lista2 para lista1 |
|  |

|  |
| --- |
| lista1.splice(it, lista2); |
|  |

|  |
| --- |
| cout << "\nElementos de lista1:\n"; |
|  |

|  |
| --- |
| for(it = lista1.begin(); it != lista1.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
| cout << "\nTamanho de lista2: " << l2.size() << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // remover todos os elementos iguais a um valor |
|  |

|  |
| --- |
| // função remove |
|  |

|  |
| --- |
| lista1.remove(20); |
|  |

|  |
| --- |
| cout << "\nElementos de lista1 sem o valor 20:\n"; |
|  |

|  |
| --- |
| for(it = lista1.begin(); it != lista1.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // função remove\_if |
|  |

|  |
| --- |
| // remove se ocorrer determinada condição |
|  |

|  |
| --- |
| int vet2[] = {6, 8, 10, 5, 20, 21}; |
|  |

|  |
| --- |
| list<int> lista3(vet2, vet2 + 6); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\nElementos de lista3 antes da remocao:\n"; |
|  |

|  |
| --- |
| for(it = lista3.begin(); it != lista3.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << " "; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // removendo todos os elementos pares de lista3 |
|  |

|  |
| --- |
| lista3.remove\_if(par); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << "\n\nElementos de lista3 depois da remocao:\n"; |
|  |

|  |
| --- |
| for(it = lista3.begin(); it != lista3.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << " "; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int vet3[] = {17, 60, 5, 30, 50}; |
|  |

|  |
| --- |
| list<int> lista4(vet3, vet3 + 5); |
|  |

|  |
| --- |
| lista4.sort(); |
|  |

|  |
| --- |
| cout << "\n\nElementos de lista4 ordenados crescentemente:\n"; |
|  |

|  |
| --- |
| for(it = lista4.begin(); it != lista4.end(); it++) |
|  |

|  |
| --- |
| cout << \*it << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // ordenando strings |
|  |

|  |
| --- |
| list<string> lista5; |
|  |

|  |
| --- |
| list<string>::iterator it5; |
|  |

|  |
| --- |
| lista5.push\_back("python"); |
|  |

|  |
| --- |
| lista5.push\_back("c++"); |
|  |

|  |
| --- |
| lista5.push\_back("ruby"); |
|  |

|  |
| --- |
| lista5.push\_back("haskell"); |
|  |

|  |
| --- |
| lista5.sort(); |
|  |

|  |
| --- |
| cout << "\n\nElementos de lista5 ordenados:\n"; |
|  |

|  |
| --- |
| for(it5 = lista5.begin(); it5 != lista5.end(); it5++) |
|  |

|  |
| --- |
| cout << \*it5 << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // ordenando usando algum critério |
|  |

|  |
| --- |
| // exemplo: ordenando pelo tamanho da string |
|  |

|  |
| --- |
| lista5.sort(comparar\_tamanho); |
|  |

|  |
| --- |
| cout << "\n\nElementos de lista5 ordenados pelo tamanho:\n"; |
|  |

|  |
| --- |
| for(it5 = lista5.begin(); it5 != lista5.end(); it5++) |
|  |

|  |
| --- |
| cout << \*it5 << " - tamanho: " << (\*it5).length() << endl; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return 0; |
|  |

}