



Lista 13

Exercícios

1. Iterando sobre um `std::vector`

Crie um `std::vector<int>` com valores de 1 a 10. Utilize um iterador para percorrer o vetor e imprima os valores.

Modifique o código para duplicar cada valor no vetor usando um iterador.

2. Iterando e Modificando um `std::list`

Crie uma `std::list<int>` com valores de 10 a 1 (em ordem decrescente).

Use um iterador para percorrer a lista e remova todos os elementos pares.

3. Acessando Elementos em um `std::map`

Crie um `std::map<std::string, int>` que mapeia nomes de frutas para quantidades (por exemplo, "maçã": 5, "banana": 3, "laranja": 8).

Usando um iterador, exiba cada par chave-valor.

Modifique o exercício para atualizar o valor de cada fruta adicionando 2 ao valor original.

4. Usando `std::set` para Valores Únicos

Crie um `std::set<int>` com os valores {2, 4, 6, 8, 10, 12}.

Percorra o `std::set` com um iterador e exiba os valores.

Tente adicionar um valor duplicado ao `std::set` e observe o resultado ao imprimir os elementos novamente.

5. Iteração com `std::deque`

Crie um `std::deque<int>` e adicione os valores 1, 2, 3 no início e os valores 10, 20, 30 no final.

Use um iterador para exibir todos os elementos.

Remova o primeiro e o último elemento da `std::deque` usando métodos apropriados e imprima o conteúdo restante.

6. Exemplo com `std::stack` (Sem Iterador)

Crie uma `std::stack<int>` e adicione os valores 5, 10, 15, 20.

Escreva um loop para acessar e remover elementos da `std::stack` até que ela esteja vazia, imprimindo o elemento superior a cada remoção.

7. Exemplo com `std::queue` (Sem Iterador)

Crie uma `std::queue<std::string>` e adicione os valores "primeiro", "segundo", "terceiro".

Escreva um loop que percorra e remova elementos da `std::queue`, exibindo cada elemento removido até que a fila esteja vazia.

8. Manipulando Iteradores com `std::forward_list`

Crie uma `std::forward_list<int>` e adicione os valores 1, 2, 3, 4, 5.

Use um iterador para remover o segundo elemento da lista e, em seguida, imprima os elementos restantes.

Escreva uma função que recebe uma `std::forward_list` e um valor `int` e retorna um iterador para o primeiro elemento igual ao valor, ou o iterador `end` se não houver correspondência.

9. Iteradores com `std::unordered_map`

Crie um `std::unordered_map<char, int>` para contar as ocorrências de cada caractere em uma string (por exemplo, "abracadabra").

Use um iterador para exibir cada caractere e seu número de ocorrências.

10. Operações de Aritmética de Iteradores com `std::vector`

Crie um `std::vector<int>` com valores de 1 a 10.

Use aritmética de iteradores para acessar o terceiro elemento e exibi-lo.

Modifique o valor do terceiro elemento para 99 e exiba o vetor completo para verificar a alteração.

11. Reversão de um `std::vector` com `std::reverse_iterator`

Crie um `std::vector<int>` com valores de 1 a 5.

Use um `std::reverse_iterator` para percorrer e exibir o vetor em ordem inversa.

12. Comparação de Iteradores

Crie um `std::vector<int>` com valores de 10 a 50 em incrementos de 10.

Escreva uma função que recebe dois iteradores e verifica se ambos apontam para o mesmo elemento no vetor.

Use a função para comparar diferentes iteradores e exiba o resultado.