Fundamentos de Orientação à Objetos
Prof. Leonardo Medeiros
17/02/2025
Lista de Exercícios #3

Gabriel dos Santos Schmitz
(RA: 2487438)

Este documento foi criado usando LaTeX
https://github.com/gabrielzschmitz/uni/tree/main/poo/lista3.

1. Escreva um programa C++ que use std::map

Resposta:

```cpp
/**
 * @file 1.cpp
 * @brief Exemplo de uso da estrutura std::map em C++
 *
 * Este programa demonstra como usar um std::map para armazenar pares
 * chave-valor, onde a chave é uma string (nome) e o valor é um inteiro (idade).
 * Ele permite inserir, exibir e buscar elementos no mapa.
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <map>
#include <string>

int main() {
  /* Declaração do mapa para armazenar nomes e idades */
  std::map<std::string, int> dados;

  /* Inserindo elementos no mapa */
  dados["Alice"] = 25;
  dados["Bob"] = 30;
  dados["Charlie"] = 22;

  /* Exibindo todos os elementos do mapa */
  std::cout << "Conteúdo do mapa:\n";
  for (std::map<std::string, int>::iterator it = dados.begin();
       it != dados.end(); ++it)
    std::cout << it->first << " -> " << it->second << '\n';

  /* Busca por uma chave no map */
  std::string chave = "Alice";
  std::map<std::string, int>::iterator it = dados.find(chave);
  if (it != dados.end())
    std::cout << chave << " tem " << it->second << " anos.\n";
  else std::cout << "Nome não encontrado no mapa.\n";

  return 0;
}
```

2. Escreva um programa C++ que use std::multimap

Resposta:

```cpp
/**
 * @file 2.cpp
 * @brief Exemplo de uso da estrutura std::multimap em C++
 *
 * Este programa demonstra como usar um std::multimap para armazenar pares
 * chave-valor, onde a chave é uma string (nome) e o valor é um inteiro (idade).
 * Diferente de std::map, std::multimap permite múltiplos valores para a mesma
 * chave. Ele permite inserir, exibir e buscar elementos no multimap.
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <map>
#include <string>

int main() {
  /* Declaração do multimap para armazenar nomes e idades */
  std::multimap<std::string, int> dados;

  /* Inserindo elementos no multimap */
  dados.insert(std::make_pair("Alice", 25));
  dados.insert(std::make_pair("Bob", 30));
  dados.insert(std::make_pair("Charlie", 22));
  dados.insert(std::make_pair("Alice", 28)); /* Alice aparece duas vezes */

  /* Exibindo todos os elementos do multimap */
  std::cout << "Conteúdo do multimap:\n";
  for (std::multimap<std::string, int>::iterator it = dados.begin();
       it != dados.end(); ++it)
    std::cout << it->first << " -> " << it->second << '\n';

  /* Busca por uma chave no multimap */
  std::string chave = "Alice";
  std::pair<std::multimap<std::string, int>::iterator,
            std::multimap<std::string, int>::iterator>
    faixa = dados.equal_range(chave);

  if (faixa.first != dados.end() && faixa.first->first == chave) {
    std::cout << chave << " tem as seguintes idades: ";
    for (std::multimap<std::string, int>::iterator it = faixa.first;
         it != faixa.second; ++it)
      std::cout << it->second << " ";
    std::cout << "\n";
  } else std::cout << "Nome não encontrado no multimap.\n";

  return 0;
}
```

3. Escreva um programa C++ que use std::set

Resposta:

```cpp
/**
 * @file 3.cpp
 * @brief Exemplo de uso da estrutura std::set em C++
 *
 * Este programa demonstra como usar um std::set para armazenar valores únicos.
 * O conjunto armazena nomes sem duplicatas e permite inserção, exibição
 * e busca de elementos.
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <set>
#include <string>

int main() {
  /* Declaração do conjunto para armazenar nomes */
  std::set<std::string> nomes;

  /* Inserindo elementos no conjunto */
  nomes.insert("Alice");
  nomes.insert("Bob");
  nomes.insert("Charlie");
  nomes.insert("Alice"); /* Inserção duplicada (ignorada pelo set) */

  /* Exibindo todos os elementos do conjunto */
  std::cout << "Conteúdo do conjunto:\n";
  for (std::set<std::string>::iterator it = nomes.begin(); it != nomes.end();
       ++it)
    std::cout << *it << '\n';

  /* Busca por uma chave no set */
  std::string chave = "Alice";
  if (nomes.find(chave) != nomes.end())
    std::cout << chave << " está no conjunto.\n";
  else std::cout << chave << " não está no conjunto.\n";

  return 0;
}
```

4. Escreva um programa C++ que use std::set

Resposta:

```cpp
/**
 * @file 4.cpp
 * @brief Exemplo de uso da estrutura std::multiset em C++
 *
 * Este programa demonstra como usar um std::multiset para armazenar valores que
 * podem se repetir. O multiset armazena nomes permitindo duplicatas e permite
 * inserção, exibição e busca de elementos.
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <set>
#include <string>

int main() {
  /* Declaração do multiset para armazenar nomes */
  std::multiset<std::string> nomes;

  /* Inserindo elementos no multiset */
  nomes.insert("Alice");
  nomes.insert("Bob");
  nomes.insert("Charlie");
  nomes.insert("Alice"); /* Inserção duplicada (permitida pelo multiset) */

  /* Exibindo todos os elementos do multiset */
  std::cout << "Conteúdo do multiset:\n";
  for (std::multiset<std::string>::iterator it = nomes.begin();
       it != nomes.end(); ++it)
    std::cout << *it << '\n';

  /* Busca por uma chave no multiset */
  std::string chave = "Alice";
  int count = nomes.count(chave);
  if (count > 0)
    std::cout << chave << " aparece " << count << " vezes no multiset.\n";
  else std::cout << chave << " não está no multiset.\n";

  return 0;
}
```

5. Escreva um programa C++ que use std::stack

Resposta:

```cpp
/**
 * @file 5.cpp
 * @brief Exemplo de uso da estrutura std::stack em C++
 *
 * Este programa demonstra como usar um std::stack para armazenar e manipular
 * uma pilha de nomes. A pilha segue a ordem LIFO (Last In, First Out),
 * permitindo inserção (push), remoção (pop) e exibição do elemento no topo
 * (top).
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <stack>
#include <string>

int main() {
  /* Declaração da pilha para armazenar nomes */
  std::stack<std::string> pilha;

  /* Inserindo elementos na pilha */
  pilha.push("Alice");
  pilha.push("Bob");
  pilha.push("Charlie");

  /* Exibindo o elemento no topo da pilha */
  if (!pilha.empty())
    std::cout << "Elemento no topo da pilha: " << pilha.top() << '\n';

  /* Removendo elementos da pilha */
  std::cout << "Removendo elementos da pilha:\n";
  while (!pilha.empty()) {
    std::cout << pilha.top() << '\n';
    pilha.pop();
  }

  /* Verificando se a pilha está vazia */
  if (pilha.empty()) std::cout << "A pilha está vazia.\n";

  return 0;
}
```

6. Escreva um programa C++ que use std::queue

Resposta:

```cpp
/**
 * @file 6.cpp
 * @brief Exemplo de uso da estrutura std::queue em C++
 *
 * Este programa demonstra como usar um std::queue para armazenar e manipular
 * uma fila de nomes. A fila segue a ordem FIFO (First In, First Out),
 * permitindo inserção (push), remoção (pop) e acesso ao primeiro e último
 * elemento (front e back).
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <queue>
#include <string>

int main() {
  /* Declaração da fila para armazenar nomes */
  std::queue<std::string> fila;

  /* Inserindo elementos na fila */
  fila.push("Alice");
  fila.push("Bob");
  fila.push("Charlie");

  /* Exibindo o elemento na frente e atrás da fila */
  if (!fila.empty()) {
    std::cout << "Primeiro da fila: " << fila.front() << '\n';
    std::cout << "Último da fila: " << fila.back() << '\n';
  }

  /* Removendo elementos da fila */
  std::cout << "Atendendo os elementos da fila:\n";
  while (!fila.empty()) {
    std::cout << fila.front() << '\n';
    fila.pop();
  }

  /* Verificando se a fila está vazia */
  if (fila.empty()) std::cout << "A fila está vazia.\n";

  return 0;
}
```

7. Escreva um programa C++ que use std::priority_queue

Resposta:

```cpp
/**
 * @file 7.cpp
 * @brief Exemplo de uso da estrutura std::priority_queue em C++
 *
 * Este programa demonstra como usar um std::priority_queue para armazenar e
 * processar uma fila de prioridades de idades. Os elementos com maior valor têm
 * maior prioridade e são removidos primeiro.
 *
 * @author Gabriel dos Santos Schmitz
 * @date 17-02-2025
 */

#include <iostream>
#include <queue>

int main() {
  /* Declaração da fila de prioridade para armazenar idades */
  std::priority_queue<int> fila_prioridade;

  /* Inserindo elementos na fila de prioridade */
  fila_prioridade.push(25);
  fila_prioridade.push(30);
  fila_prioridade.push(22);

  /* Exibindo e removendo os elementos da fila de prioridade */
  std::cout << "Processando a fila de prioridade:\n";
  while (!fila_prioridade.empty()) {
    std::cout << fila_prioridade.top() << '\n';
    fila_prioridade.pop();
  }

  /* Verificando se a fila está vazia */
  if (fila_prioridade.empty())
    std::cout << "A fila de prioridade está vazia.\n";

  return 0;
}
```

Compilação e Execução dos Exercícios

Para compilar e executar os exercícios $1 - 7$, siga os passos abaixo:

1. **Torne o script `run_tests.sh` executável**:
   No terminal, execute o seguinte comando para tornar o script executável:

```
$ chmod +x run_tests.sh
```

2. **Execute o script para compilar e rodar os códigos**:
   Em seguida, execute o script para compilar e rodar os códigos dos exercícios:

```
$ ./run_tests.sh
```

Isso deve compilar os arquivos com `g++` e rodar os testes automaticamente. Se houver algum erro na compilação, o script irá mostrar as mensagens de erro.