Standard Template Library (STL)

A biblioteca padrão de gabaritos (*tradução livre*) ou STL, em bom jargão de c++, é um conjunto de classes e funções que resolvem muitos dos problemas que a linguagem C tem em relação ao uso de vetores, listas, ordenação, entre outras estruturas de dados e algoritmos que tem uso recorrente na resolução de problemas, mas não tem suporte nas bibliotecas padrão da linguagem.

Em C++ a STL possui uma implementação padrão para a maioria dessas estruturas, especialmente vetores, listas, pilhas, filas além de ordenação, reversão, processamento e outros algoritmos que são usualmente necessários em nossos problemas.

A STL implementa todas essas estruturas de forma dinâmica (assim não precisamos nos preocupar com alocação de memória das estruturas em si) bem como com o uso de Templates, dessa forma as estruturas são genéricas o suficiente e fazem uso de sobrecarga de operadores e funções, quando necessário, para realizar as operações relacionadas à estrutura em si. Por exemplo, se você quer criar um vetor de inteiros ordenados, o algoritmo *sort* vai usar os operadores relacionais (maior, menor e igual) para a ordenação, mas se seu objetivo é ordenar um vetor de objetos de uma classe criada por você, a classe precisa sobrecarregar esses operadores para que o vetor seja ordenado.

A STL é tão importante para C++, que muitos exemplos considerados simples na linguagem já assumem o uso dessa biblioteca. É comum, por exemplo, quando vamos pesquisar vetores em c++, que seja encontrado material usando a classe vector ao invés de materiais usando new[]/delete[] ou mesmo vetores estáticos.

Este material de estudo contém algumas apresentações sobre a STL, mas a melhor forma de aprender é usando através de problemas, uma vez que a quantidade de *containers* e seus métodos/algoritmos é muito grande para uma aula só. Por isso sugiro que não foque em decorar a API de um container específico, mas sim em saber que ele existe e aprender a usar a documentação contida no https://www.cplusplus.com/.

Além disso, é importante saber quando um container se encaixa melhor em um determinado problema, especialmente pensando nas principais operações que você irá realizar nele. Muitos containers podem parecer resolver o mesmo problema, mas a principal diferença está nas complexidades relacionadas ao uso de memória, acesso a um elemento específico, inserção de novos elementos, etc.

Slides

- **Conjunto de slides do grupo de professores LP1 (material 2020.6)**
  - Standard Template Library (2020.6)

- **Versão resumida dos slides (cobre apenas vector, iterator, list, map e algorithm)**
  - Standard Template Library
  - Exemplos de código (links também nos slides) no replit.

Textos

- **Documentação no https://www.cplusplus.com/**
  - **STL**
  - **Algorithm**

- **Geek for Geeks (em ingles)**
  - STL Tutorial: Link

- **Stl Cheat-Sheets (em ingles, MUITO bom)**
  - C++ Data Structures and Algorithms Cheat Sheet: Link (em ingles, **MUITO bom**)
  - Hackerearth: Link (em inglês)

- **Livro C++ como programar**
  - Capítulo 22: Standard Template Library
  - DEITEL, Harvey M.; DEITEL, Paul J. C++ como programar. 5. ed. São Paulo: Pearson Prentice Hall, 2006. 1163 p. ISBN: 9788576050568

Vídeos

- **Aula sobre o tema referente aos slides do grupo de professores LP1 (material 2020.6) : link**

- **The cherno (Ingles, mas podem usar as legendas do Youtube)**
  - Iterators
  - Vector