Curso de C/C++ Avançado

Aula 8 – Standard Template Library

Allan Lima – http://allanlima.wordpress.com

- Você pode:
  - copiar, distribuir, exibir e executar a obra
  - criar obras derivadas
  - fazer uso comercial da obra
- Sob as seguintes condições:
  - **Atribuição**. Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.
  - **Compartilhamento pela mesma Licença**. Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.
  - Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.
  - Qualquer uma destas condições podem ser renunciadas, desde que Você obtenha permissão do autor.
- **Veja aqui a licença completa**

Standard Template Library

- É a biblioteca padrão de C++
- Padronizada pelo ANSI em 1997
- Possui basicamente classes de armazenamento (containers), algoritmos e iterators
- É uma biblioteca genérica, ou seja a maioria dos seus componentes são templates

Por que utilizar STL

- Podemos utilizar os componentes de STL para diminuir o nosso trabalho
- Por exemplo:
  - Não precisamos criar uma classe pilha, porque ela já existe na STL (stack)
- Não reinvente a roda!!!

Standard Template Library

- Convenções:
  - Quando incluímos um hearder não precisamos utilizar o ‘.h’ no seu final:
    - #include <iostream>
  - As bibliotecas antigas de c agora possuem um ‘c’ a mais no seu nome e também não precisamos mais do ‘.h’ no final do arquivo
    - #include <cstring>
  - Mas as formas antigas de incluir arquivos também funcionam

Organização da STL

Containers	
<vector>	Array unidimensional do tipo T
<list>	Lista duplamente ligada do tipo T
<deque>	Lista ligada ‘double-ended’ do tipo T
<queue>	Lista ligada do tipo T
<stack>	Pilha do tipo T
<map>	Array associativo do tipo T (red-blak tree)
<set>	Conjunto do tipo T
<bitset>	Array de booleanos

Organização da STL

General Utilities	
<utility>	Operadores
<funcional>	Funções
<memory>	Funções para manipulação de memória
<ctime>	Funções de C para manipulação de data e hora
Iterators	
<iterator>	Definição e suporte para Iterators
Algorithms	
<algorithm>	Algoritmos gerais
<cstdlib>	bsearch() e qsort()

Organização da STL

Diagnostics	
<exception>	Definição da classe exception
<stdexception>	Exceções da biblioteca padrão
<cassert>	Diagnóstico de Erros
<cerrno>	Tratamento de erro com as funções de C

Organização da STL

Strings	
<string>	string
<cctype>	Classificação de caracteres
<cwctype>	Classificação de wide-caracters
<cstring>	Funções de C para manipulação de strings
<cwchar>	Funções de C para classificação de wide-caracters
<cstdlib>	Mais funções de C para manipulação de strings

Organização da STL

Input/Output	
<iosfwd>	Declarações forward de facilidades de E/S.
<iostream>	Objetos e operações padrões de E/S.
<ios>	Classes bases de E/S.
<streambuf>	Stream buffers.
<istream>	Templates de stream de entrada.
<ostream>	Templates de stream de saída.
<iomanip>	Manipuladores.
<sstream>	Stream para/de strings.

Organização da STL

Input/Output	
<cstdlib>	Classificação de caracteres e funções
<fstream>	Streams para/de arquivos
<cstdio>	printf() e scanf()
<cwchar>	printf() e scanf() para wide characters
Language Suport	
<locale>	Representa diferenças culturais
<clocale>	Funções de C para representar diferenças culturais

Organização da STL

Language Suport	
<limits>	Limites numéricos
<climits>	Macros de C com os limites numéricos
<cfloat>	Macros de C com os limites de ponto flutuante
<new>	Gerenciamento de memória dinâmico
<typeinfo>	Suporte para identificação de tipos em tempo de execução
<exception>	Suporte para o tratamento de exceções
<cstddef>	Suporte para a biblioteca da linguagem C
<cstdargs>	Criação de funções com lista de argumentos variável
<csetjmp>	Funções de C para manipulação de Pilhas

Organização da STL

Language Suport	
<stdlib>	Finalização do programa
<ctime>	Relógio (Clock) do sistema
<csignal>	Manipulação de sinais em C
Numéricos	
<complex>	Números complexos e seus operadores
<valarray>	Vetores numéricos e operações
<numeric>	Operações numéricas generalizadas
<cmath>	Funções matemáticas de C
<stdlib>	Geração de números aleatórios em C

Algumas classes da STL

- ofstream
- ifstream
- fstream
- vector
- basic_string
- queue
- stack
- set
- const_iterator
- map

E/S Padrão

- C++ possui uma biblioteca de E/S chamada iostream
- Esta possui alguns dispositivos predefinidos:

Nome	Tipo	Buffered	Descrição
cin	istream	Sim	Entrada padrão (normalmente o teclado)
cout	ostream	Sim	Saída Padrão (normalmente o monitor)
clog	ostream	Sim	Saída de erro padrão (normalmente o monitor)
cerr	ostream	Não	Saída de erro padrão (normalmente o monitor)

A classe ostream

- É um tipo de saída de dados
- Podemos enviar dados para objetos deste tipo através do operador << (operador de inserção)
- O operador << pode ser utilizado mais de uma vez na mesma sentença e não adiciona um '\n' no final da linha
- Exemplos:

```
cout << "UFPE\n";
cerr << "CIn" << endl;//endl realiza a quebra de linha
clog << "C++" << "\n";
```

A classe istream

- É um tipo de entrada de dados
- Podemos ler dados de objetos deste tipo através do operador >>
- O operador >> também pode ser utilizado mais de uma vez na mesma sentença
- A leitura só é realizada após um ‘/r’, ‘/n’ ou ‘ ’
- Exemplo:

```
int a;
float b;
cin >> a >> b;
```

E/S com Arquivos

- Principais classes:
  - **ifstream** – Utilizada para leitura de dados de arquivos
  - **ofstream** – Utilizada para gravação de dados de arquivos
  - **fstream** – Utilizada para ambos leitura e gravação de dados de arquivos
- Para utilizar esta classe devemos incluir o arquivo **fstream.h**

E/S com Arquivos

- Abrindo um arquivo:
  - void open(const char *nomeArquivo, openMode modo);
  - Exemplo:

```
fstream myStream();
myStream.open("meuArquivo", ios::in);
```

- Fechando um arquivo:
  - void close();
  - Exemplo:

```
myStream.close();
```

  - Este método é chamado no destrutor das classes ifstream, ofsteam e fstream

E/S com Arquivos

- Modos de Abertura:

Modo	Descrição
ios::in	Abre um arquivo para leitura
ios::out	Abre um arquivo para escrita
ios::app	Abre um arquivo apenas para adicionar dados ao seu final
Ios::ate	Abre um arquivo existente e move o cursor para o final do arquivo
ios::trunc	Se o arquivo existir ele será sobrescrito
ios::binary	Abre o arquivo em modo binário

Os modos acima podem ser combinadas com o operador '|'

E/S com Arquivos

- Modos de abertura padrão das classes:

Classe	Modo
ofstream	ios::out | ios::trunc
ifstream	ios::in
fstream	ios::in | ios::out

E/S com Arquivos

- Podemos abrir um arquivo sem usar o método open:
  - ofstream arq(“meuAquivo”, ios::out | ios::app);
- Método is_open()
  - indica se o arquivo esta aberto
- Método eof()
  - indica se o arquivo chegou ao seu fim
- Função getline()
  - Lê uma linha do arquivo

E/S com Arquivos

- Podemos utilizar os operadores >> e << para ler e escrever em arquivos respectivamente
- Para manipulação de arquivos binários existem os métodos:
  - void write(char *buffer, steamsize tamanho);
  - void read(char *buffer, steamsize tamanho);

Exemplo

- exemploArquivo.cpp

A classe vector

- É um ‘array’ dinâmico, ou seja o seu tamanho não é fixo!
- Seus elementos podem ser acessados através do operador [ ]
- Tem uma ótima performance
- Exemplo:
  - exemploVector.cpp

strings

- O C++ **não** possui uma classe chamada string
- Mas sim a classe **basic_string** que é um template e pode representar uma string de qualquer tipo de dado.
- Mas como podemos declarar uma string?
- O namespace std possui a seguinte linha:
  - typedef basic_string<char> string;

strings

- Seus caracteres podem ser acessadas através do operador [ ]
- Não possuem limite de tamanho
- A classe base_string possui um grande número de métodos que são muito úteis e muito mais completos do que as funções definidas no header <string.h>
- Exemplo:
  - exemploString.cpp

A classe queue

- Implementa a disciplina de um fila
- Política FIFO: *first-in-first-out*
- Exemplo:
  - exemploQueue.cpp

A classe stack

- Implementa a disciplina de uma pilha
- Política LIFO: *last-in-firt-out*
- Pode ser implementada sobre diversas estruturas de dados:
  - vector
  - list
  - deque
- O valor default é deque
- Exemplo:
  - exemploStack.cpp

Classe set

- Representa a idéia de um conjunto
- Por isso, não possui elementos duplicados
- Os elementos são inseridos de forma ordenada
- Exemplo:
  - exemploSet.cpp

As classes const_iterator

- É um tipo utilizado para percorrer os containers da STL
- Objetos deste tipo são ponteiros
- Cada container possui uma classe com este nome;
- Para percorrer um container podemos utilizar os métodos:
  - begin() - Ponteiro para o início do container
  - end() - Ponteiro para o fim do container
- Exemplo:
  - exemploIterator.cpp

A classe map

- Associa um valor a uma chave
- Chaves duplicadas são ignoradas
- Elementos podem ser inseridos ou modificados através do operador [ ]
- Muito rápida para operações de busca
- Exemplo:
  - exemploMap.cpp

Exercícios

1)Crie um programa chamado mycopy que recebe como argumento, na sua função main, dois caminhos para arquivos de texto e copia todo o conteúdo o primeiro para segundo. O programa também deverá imprimir no monitor o número total de linhas copiadas.

Exercícios

2) Crie uma função isPalindromo() que recebe um vetor como parâmetro. Ela deve retornar true se ele é um palíndromo e false caso contrário.

- Exemplo: Um vetor contendo 1, 2, 3, 2, 1 é um palíndromo. E um vetor contendo 1, 4, 3, 2, 1 não é um palíndromo.

Referências

- Stroustrup, Bjarne. The C++ Programming Language, Special Edition
- Eckel, Bruce. Thinking in C++, 2nd ed. Volume 1
- TechZone
  - http://www.nexsun.com.br/techzone.html
- Slides de Gustavo (ghcp@cin.ufpe.br) do curso de C/C++
- Standard Template Library Programmer's Guide
  - http://www.sgi.com/tech/stl/