AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 240: Advanced Programming Techniques
Lecture 16: C++ Standard Library, PA 3 (Due)

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

2ND MARCH 2023

Carolinian Creed: "I will practice personal and academic integrity."

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lecture 16

- Introduction Section
  - Recap of Lecture 15
- Main Section
  - Concept: Standard Library
  - Discussion: Project
- Concluding Section
  - About next lecture – Lecture 17
  - Ask me anything

Introduction Section

Recap of Lecture 15

- Reviewed HW#4
- We looked at the concept of operators
  - Many types
  - Precedence order when evaluating
  - Defining one's own operator
- Programming Assignment #3 due today

Main Section

Concept: C++ Standard Library

C++ reference

C++98, C++03, C++11, C++14, C++17, C++20, C++23 | Compiler support C++11, C++14, C++17, C++20, C++23

- Freestanding implementations
- ASCII chart

Language
- Basic concepts
- Keywords
- Preprocessor
- Expressions
- Declaration
- Initialization
- Functions
- Statements
- Classes
- Overloading
- Templates
- Exceptions

Standard library (headers)

Named requirements

Feature test macros (C++20)

Language support library
- Source code information (C++20)
- Type support
- Program utilities
- Coroutine support (C++20)
- Three-way comparison (C++20)
- numeric_limits — type_info
- initializer_list (C++11)

Concepts library (C++20)

Diagnostics library
- exception — System error
- basic_stacktrace (C++23)

Memory management library
- unique_ptr (C++11)
- shared_ptr (C++11)

Metaprogramming library (C++11)
- Type traits — ratio
- integer_sequence (C++14)

General utilities library
- Function objects — hash (C++11)
- Utility functions — bitset
- pair — tuple (C++11)
- optional (C++17)
- expected (C++23)
- variant (C++17) — any (C++17)
- String conversions (C++17)
- Formatting (C++20)
- Bit manipulation (C++20)

Strings library
- basic_string — char_traits
- basic_string_view (C++17)
- Null-terminated strings: byte — multibyte — wide

Containers library
- array (C++11) — vector — deque
- list — forward_list (C++11)
- set — multiset
- map — multimap
- unordered_map (C++11)
- unordered_multimap (C++11)
- unordered_set (C++11)
- unordered_multiset (C++11)
- stack — queue — priority_queue
- flat_set (C++23)
- flat_multiset (C++23)
- flat_map (C++23)
- flat_multimap (C++23)
- span (C++20) — mdspan (C++23)

Iterators library

Ranges library (C++20)

Algorithms library
- Execution policies (C++17)
- Constrained algorithms (C++20)

Numerics library
- Common math functions
- Mathematical special functions (C++17)
- Mathematical constants (C++20)
- Numeric algorithms
- Pseudo-random number generation
- Floating-point environment (C++11)
- complex — valarray

Date and time library
- Calendar (C++20) — Time zone (C++20)

Localizations library
- locale — Characeter classification

Input/output library
- Print functions (C++23)
- Stream-based I/O
- Synchronized output (C++20)
- I/O manipulators

Filesystem library (C++17)
- path

Regular expressions library (C++11)
- basic_regex — algorithms

Concurrency support library (C++11)
- thread — jthread (C++20)
- atomic — atomic_flag
- atomic_ref (C++20)
- memory_order — condition_variable
- Mutual exclusion — Semaphores (C++20)
- future — promise — async
- latch (C++20) — barrier (C++20)

Technical specifications
- **Standard library extensions** (library fundamentals TS)
  - resource_adaptor — invocation_type
- **Standard library extensions v2** (library fundamentals TS v2)
  - propagate_const — ostream_joiner — randint
  - observer_ptr — detection idiom
- **Standard library extensions v3** (library fundamentals TS v3)
  - scope_exit — scope_fail — scope_success — unique_resource
- **Parallelism library extensions v2** (parallelism TS v2)
  - simd
- **Concurrency library extensions** (concurrency TS) — **Transactional Memory** (TM TS)
- **Reflection** (reflection TS)

Credit: https://en.cppreference.com/w/cpp

Many Implementations

Name	Organization	Homepage	Acronym	Licence	Latest release
GNU C++ Standard Library	GNU Project and Free Software Foundation	[1]	libstdc++	GPLv3	Unknown
LLVM C++ Standard Library	LLVM Developer Group	[2]	libc++	Apache License 2.0 with LLVM Exceptions	Every 2 weeks
NVIDIA C++ Standard Library	Nvidia	[3]	libcu++	Apache License 2.0 with LLVM Exceptions	October 12, 2022; 4 months ago
Microsoft C++ Standard Library	Microsoft	[4]	MSVC STL	Apache License 2.0 with LLVM Exceptions	Daily
HPX C++ Standard Library for Parallelism and Concurrency	STELLAR Group	[5]	HPX	Boost Software License 1.0	August 6, 2022; 6 months ago
Electronic Arts Standard Template Library	Electronic Arts	[6]	EASTL	BSD 3-Clause License	October 20, 2021; 16 months ago
Dinkum C++ Library	Dinkumware	[7]	Unknown	Commercial	Unknown
Cray C++ Standard Library	Cray User Group	[8]	Unknown	Commercial	Unknown

Credit: https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library

Why Use Standard Library and Why Not ?

- Note: One can always implement a functionality themselves
- Reasons to reuse
  - Lesser development effort. Someone has created it.
  - Task needs specialized knowledge that the developer does not have
  - Usually, well tested.
  - Usually, efficient.
  - Well-documented. So, code using them easier to maintain
- Reasons not to reuse
  - Want to be in control of behavior and performance
  - Want to control code size/ memory footprint
  - Task needs specialized knowledge that the developer has

Credit: Adapted from 'Fundamentals of C++ Programming', Richard Halterman

Commonly Used: String

- Purpose: Make working with strings easy
- Examples
  - **Position**: front, back
  - **Size related**: size, capacity
  - **Character manipulation:** replace
  - **Search**: find
  - **Type conversion**: stoi, stof

Reference:
https://en.cppreference.com/w/cpp/string/basic_string

Credit: https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library

C++ Standard Library
- Input/output
- Strings
- algorithm
- functional

Containers
- Sequence containers
- Associative containers
- Unordered associative containers

C standard library
- Data types
- Character classification
- Strings
- Mathematics
- File input/output
- Date/time
- Localization
- Memory allocation
- Process control
- Signals
- Alternative tokens
- Miscellaneous headers:
  - <assert.h>
  - <errno.h>
  - <setjmp.h>
  - <stdarg.h>

Commonly Used: String

- Code illustration
  - Front
  - Back
  - Size
  - Capacity
  - substr

Description: https://en.cppreference.com/w/cpp/string/basic_string
Demo: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class15and16_OperatorSTL/src/Class15and16_OperatorSTL.cpp, demoStrings()

Commonly Used: Mathematical Functions

- Purpose: Make numerical computation easy

- Examples
  - **Basic**: abs, mod, nan (not a number), round, nearestint, infinity
  - **Exponential**: exp, log
  - **Power:** pow, sqrt, hypot (computes square root of the sum of the squares of two or three)
  - **Trigonometric**: sin, cos, tan, atan
  - **Floating point**: round, floor, ceil

Description: https://en.cppreference.com/w/cpp/numeric/math
Demo: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class15and16_OperatorSTL/src/Class15and16_OperatorSTL.cpp, demoMaths()

Commonly Used: Mathematical Functions

- Code illustration
  - Sqrt -- square root
  - Cbrt -- cubic root
  - Round
  - Nearbyint
  - Infinity, nan

- Support for complex numbers - example
  - **Description:** https://en.cppreference.com/w/cpp/numeric/complex

Sometimes Used: Algorithmic Functions

- Purpose: Make ready implementation of popular algos

- Examples
  - Sequence operations: count, find, search
  - Sorting: sort
  - Partitioning
  - Permutation
  - Set operations
  - Numeric

Notes

- auto: a *placeholder* datatype defined in C++11 whose actual type is inferred from initialization
  - https://learn.microsoft.com/en-us/cpp/cpp/auto-cpp?view=msvc-170
- use of templates, which will be explained in a later class

Sometimes Used: Algorithmic Functions

- Code illustration
  - Sort
  - permutation

Description: https://en.cppreference.com/w/cpp/algorithm
Demo: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class15and16_OperatorSTL/src/Class15and16_OperatorSTL.cpp, demoAlgos()

Sometimes Used: Container Functions

- Purpose: Make implementation of useful containers easily available

- Examples
  - Array
  - List - https://en.cppreference.com/w/cpp/container/list
  - Vector
  - Map (also called HashMap or dict in other languages)
  - Priority_queue

Description: https://en.cppreference.com/w/cpp/container
Demo: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class15and16_OperatorSTL/src/Class15and16_OperatorSTL.cpp, demoContainer()

Discussion: Course Project

PA #3 Check: Due Thursday, March 2, 2023

Course Project – Building and Assembling of Prog. Assignments in Health

- **Project**: Develop collaborative assistants (chatbots) that offer useful information about diseases
- Specifically, use the CDC dataset on diseases at: https://wwwnc.cdc.gov/travel/diseases
  - For polio, it is: https://wwwnc.cdc.gov/travel/diseases/poliomyelitis
  - Each student will choose two diseases (from 47 available).
  - Each student will also use data about the disease from WebMD. Example for polio - https://www.webmd.com/children/what-is-polio
  - Programming assignment programs will: (1) extract data about a disease from two sites, (2) process it, (3) make content available in a command-line interface, (4) handle any user query and (5) report on interaction statistics.
- *Other sources for disease information are possible. Example – NIH https://www.ninds.nih.gov/health-information/disorders*

Core Programs Needed for Project

- Prog 1: extract data from the district **[prog1-extractor]**
- Prog 2: process it (extracted data) based on questions **[prog2processor]**
- **Prog 3: make content available in a command-line interface [prog3-ui]**
- Prog 4: handle any user query and
- Prog 5: report statistics on interaction of a session, across session

Programming Assignment # 3

- **Goal**: **make content available in a command-line interface** [Name: **prog3-ui**]

- Program should do the following:
  - Run in an infinite loop until the user wants to quit
  - Handle any user response
    - User can quit by typing “Quit” or “quit” or just “q”
    - User can enter any other text and the program has to handle it. The program should write back what the user entered and say – “I do not know this information”.
  - Handle known user query types
    - “Tell me about the disease”, “What is ***malaria***?” => (Type-I1)
    - “What can I do after travel?” => (Type-I4)
    - “what is the treatment? ” => (Type-I10)
    - “Tell me about ***malaria*** vaccine” => (Type-12)
    - …
    - “**Tell me everything**” => ***Give all information extracted (I1-I12)***

S1: https://www.cdc.gov/travel/diseases/malaria
- What is malaria? [I1]
- Who is at risk? [I2]
- What can travelers do to prevent malaria? [I3]
- After Travel [I4]
- More Information [I5]

S2: https://www.webmd.com/a-to-z-guides/malaria-symptoms
- What Is Malaria? [I1]
- Malaria Causes and Risk Factors [I2]
- Types of Malaria [I6]
- Symptoms [I7]
- When to Call a Doctor About Malaria [I8]
- Malaria Diagnosis [I9]
- Malaria Treatment [I10]
- Malaria Complications [I11]
- Malaria Vaccine [I12]

Notes on PA#3

- Handle all 12 information types
  - Multiple ways to ask for same information type
  - Variant assumes disease name from context or is specified
- Handle special query: ***Tell me everything***
- Handle others
  - Chit-chat
  - Give controlled response under all condition

Programming Assignment # 3

- Code organization
  - Create a folder in your GitHub called “prog3-ui”
  - Have sub-folders: src (or code), data, doc, test
  - Write a 1-page report in ./doc sub-folder
  - Send a confirmation that code is done by updating Google sheet; optionally, send email to instructor and TA

- Use concepts learned in class
  - Classes
  - Exceptions
  - UML Diagrams

Concluding Section

Lecture 16: Concluding Comments

- We looked at the c++ standard library
  - Many types of functionality
  - String, I/O, Mathematical libraries most commonly used
- Remember that many implementations of C++ standard library, usually based on different OS or hardware
  - Implements changing specs
- Be ready to implement one's own (rather than reuse), if necessary, for performance

About Next Lecture – Lecture 17

Lecture 17: C++ Standard Libraries

- No class next week
- Code testing strategies
- Start of PA #4
- Will give HW #5

	Mar 7 (Tu)		Spring break – No class
	Mar 9 (Th)		Spring break – No class
17	Mar 14 (Tu)	Testing strategies	Prog 4 - start
18	Mar 16 (Th)	Advanced: Pointers	HW 5 due
19	Mar 21 (Tu)	Advanced: Pointers, I/O	
20	Mar 23 (Th)	Advanced: Operator overloading	Prog 4 - end
21	Mar 28 (Tu)	Advanced: Memory Management	Prog 5 - start