Modern C++ Programming

A. Topics

Federico Busato

University of Verona, Dept. of Computer Science

2018, v1.0

1. Introduction

- **A Little History of C and C++ Programming Languages**
- **C++ Philosophy**
- **Why C++ is so popular?**
- **Why C++ is so difficult?**

2. Basic Concepts I

- **Before Start**
  - What compiler
  - What editor/IDE?
  - How to compile?
- **Hello World**
- **I/O Stream**
  - `cout/cin`
  - `Filestream (ifstream/ofstream)`
- **C++ Primitive Types**
  - Built-in types
  - `size_t`, `void`, `auto`, `nullptr`
  - Conversion rules
- **Floating Point**
  - Floating point representation
  - Floating point issues
  - Floating point comparison
  - Overflow/Underflow
- **Strongly Typed Enumerators**
- **Math Operators**
- **Statement and Control Flow**
  - Loop
  - Range Loop
  - Undefined behavior
  - goto

3. Basic Concepts II

- **Memory Management: Heap and Stack**
  - Heap allocation and memory leak
  - Stack memory
  - Stack 2D allocation
  - Initialization
  - Data/Bss memory segment
- **Storage Class Specifiers**
- **Pointers and References**
  - Pointers
  - Void Pointer
  - Address-of Operator
  - Pointer Arithmetic
  - Reference
- **`sizeof` Operator**
- **Other Keywords**
  - `const`, `constexpr`
  - `volatile`
  - `using`, `decltype`
- **Explicit Type Conversion**
- **Declaration and Definition**
- **Functions**
  - Call-by-value/pointer/reference
  - `inline`
  - Default parameters
  - Overloading
- **Unions and Bitfields**
- **Preprocessing**
  - Macro
  - Pragma

4. Utilities

- **Math Functions**
  - CMath library
  - Numerical limits
  - Integer division
- **Algorithm Library**
- **String**
  - Methods
  - Operators
  - Conversion
- **Random Numbers**
  - Period and quality
  - Engines
  - Distributions
- **Time Measuring**
  - Wall-clock time
  - User time
  - System time

5. C++ Object Oriented Programming

- **C++ Classes**
  - Class hierarchy
  - Inheritance attributes
  - Class constructor
  - Default constructor
  - Class initialization
  - Copy constructor
  - `default` keyword
  - Class destructor
- **Class keyword**
  - `this`
  - `static`
  - `const`
  - `mutable`
  - `using`
  - `friend`
  - `delete`
- **Polymorphism**
  - Function binding
  - `virtual` method
  - `override/final` keywords
  - `virtual` common errors
  - Pure virtual methods
  - Abstract class and interface
- **Operator Overloading**
  - Operator $\ll$
  - Operator `operator()`
  - Operator `operator=`
- **Special Objects**
  - Aggregate
  - Trivial class
  - Standard-layout class
  - Plain old data type

6. C++ Templates and Meta-programming I

- **Function Templates**
  - Template parameters
  - Default parameters
  - Template specialization
  - Template overloading
- **Type Deduction**
  - Pass-by-Reference
  - Pass-by-Pointer
  - Pass-by-Value
  - Array type deduction
- **Compile-Time Utilities**
  - `static_assert`
  - `decltype`
  - `declval`
  - `using`
- **Type Traits**
  - Type trait library
  - Type manipulation
  - Type Relation and Transformation
- **Template Parameters**

6. C++ Templates and Meta-programming II

- **Class Templates**
  - Full/Partial specialization
  - Declaration and definition
  - `virtual`, `members`, `friend`
  - `template` keyword
  - Template template arguments
  - Template variable
- **Template Meta-Programming**
  - Factorial
  - Log
  - Unroll
- **SFINAE**
  - Function implementation
  - Class implementation
- **Variadic Template**
  - Parameter recursion
  - `sizeof...`
  - Meta-Programming
  - Specialization
- **STD Template Utility**
  - `std::pair`
  - `std::tuple`

8. Containers, Iterators, and Algorithms

- **Containers and Iterators**
- **Sequence Containers**
  - `std::array`
  - `std::vector`
  - `std::deque`
  - `std::list`
  - `std::forward_list`
  - Operations and complexity
- **Associative Containers**
  - `std::set`, `std::map`, etc.
  - Operations and complexity
- **Container Adaptors**
  - Methods
- **Implement a Custom Iterator**
  - Iterator semantic
  - Implementation example
- **Iterator Utility Methods**
  - Iterator operations
  - Range access methods
  - Iterator traits
- **Algorithms Library**
  - Implementation example
- **Lambda Expressions**
  - Capture list
  - Capture list and classes
  - `mutable`

9. Code Organization and Conventions

- **Basic Concepts**
  - Translation Unit
  - Linkage
  - Global and local scope
- **Variables Storage**
  - Storage class specifiers
  - Storage duration
- **Dealing with Multiple Files**
  - One definition rule
  - Limit template instantiations
- **Namespace**
  - One definition rule
  - Namespace alias
  - Inline namespace
  - Anonymous namespace
- **C++ Project Organization**
  - Project Files
  - Include and library
- **Coding Style and Conventions**
  - File names and spacing
  - `#include`
  - Namespace
  - Variables
  - Functions
  - Structs and Classes
  - C++11/C++14 features
  - Control Flow
  - Entity names
  - Issues

10. Debugging and Tools

- **Debugging**
  - Assertion
  - Execution debuggging
  - Memory debuggging
  - Clang sanitazer
  - Demangling
- **CMake**
- **Code Checking and Analysis**
  - Compiler warning
  - Static analyzer
- **Code Quality (Linter)**
- **Code Testing**
  - Built-in types
  - `size_t`, `void`, `auto`, `nullptr`
  - Code coverage
- **Code Commenting (Doxygen)**
- **Code Statistics**
  - Count lines of code
  - Cyclomatic complexity
- **Other Tools**
  - Code formatting
  - Assembly explorer

11. Advanced Topics

- **Move Semantic**
  - lvalues and rvalues
  - Class move semantic
  - `std::move`
  - Universal reference
  - Reference collapsing rules
  - Type deduction
  - Copy elision and RVO
  - Perfect forwarding
  - Compiler implicitly declared
- **C++ Idioms**
  - Rules of Three (and Zero)
  - Rules of Five
  - Singleton
  - PIMLP
  - CRTP
  - Template virtual function
- **Smart Pointers**
  - `std::unique_ptr`
  - `std::shared_ptr`
  - `std::weak_ptr`
- **Concurrency**
  - Thread methods
  - Parameters passing
  - Mutex
  - Atomic
  - Task-based parallelism
- **C++ Guidelines**