COMP 322 Lecture 1 - Introduction to C++

Junji Duan

2024/1/5

Today's Outline

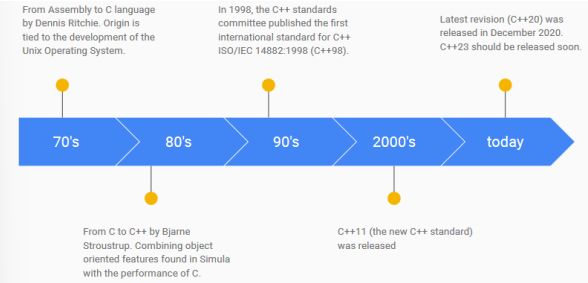
- Basic for C++
- C++ from C perspective
- C++ from Java perspective

Resources

- Recommended books:
  - The C++ Programming Language by Bjarne Stroustrup
  - Accelerated C++: Practical Programming by Example by Andrew Koenig
  - C++ Primer by Stanley Lippman and Barbara Moo
- C++ tutorials on the Internet (www.cplusplus.com, www.learncpp.com)

History of C++

- General purpose programming language
- Evolved from the C programming language
- Multi-paradigm language
- Low level memory manipulation
- Standard template library for data structures and algorithms



C++ impact and popularity

- Quickly gained popularity for system-level programming and application requiring performance
- Used in various domains
  - Game programming
  - System programming / Embedded systems
  - High-performance computing
  - Financial softwar

C++ today

- Remains one of the most widely used programming languages
- Known for its efficiency, flexibility, and a large ecosystem of libraries and frameworks
- Ongoing development with community contributions and standard updates
- Legacy code and projects written in C++ continue to be maintained and expanded

C++ vs C design

- C++ derives directly from C, so it is C plus plus more stuff
- Object Oriented via Classes

  - In C data and functions are separated
    In C++ they are encapsulated within a class where data can be hidden
- Generic programming via Templates
- Redesigned memory management via new/delete
- Operator (and functions) overloading
- Namespaces
- Reference variables
- Exception handling

C++ vs Java design

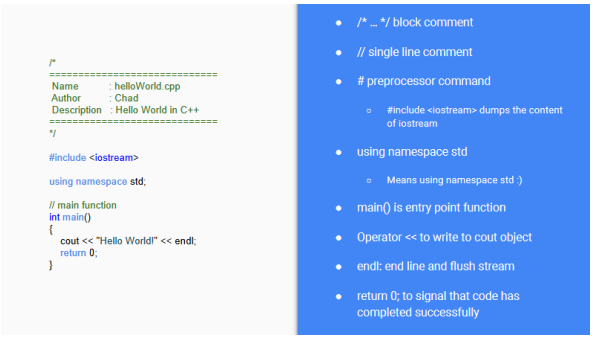
- C++
  - Compiled language
    - Runs as native binary on a target
  - Compatible with C code (very few rare exceptions)
  - Allows multiple programming paradigms without discrimination
  - Allows multiple inheritance of classes
  - Allows manual low level memory management via pointers
- Java
  - Interpreted language
    - Runs through a virtual machine
  - Uses JNI (Java Native Interface) to call C/C++ code
  - Allows multiple programming paradigms but strongly favors Object-Oriented
  - Single inheritance for classes
  - No pointers and provides automatic garbage collection mechanism

Compiling and running C++ code (1/2)

- C++ code can be contained
  - in a single file or
  - it can span over multiple files
  - **Common practice to separate header files (.h) containing declarations from implementation files (.cpp)**
  - Remember that, unlike C++, Java requires a different file for each public class and requires that the name of the file matches the name of the class
- C++ common file extensions:
  - .cpp, .c++, .c, .cxx, .cc, .hpp ···
- Use any file editor or IDE to write C++ code

Compiling and running C++ code (2/2)

- C++ code need to be compiled in order to run as an executable
- gcc / g++ under Linux, MSVC under windows, Clang under OSX
- Example: **g++ example.cpp -o example**
- Usually when we compile we invoke two operations (actually 3 if we consider pre-processing):
  - Compilation: transforms source code to object file (intermediate step between source code and final executable file)
  - Linking: producing one executable file from multiple object files
- Common practice is to use special script called Makefile for compiling complex projects
- www.cpp.sh : simple frontend gcc compiler for quick coding and debugging



Data Types (1/2)

- Same as C
  - int : sizeof(int) = 4 (4 at least, on 64-bit system)
  - float : sizeof(float) = 4 (usually 4, which is a 32-bit floating point type)

  - char : sizeof(char) = 1
  - double : sizeof(double) = 8 (which is a 64-bit floating point type)

Sizeof results may vary depending on compiler and operating system (32-bit vs 64-bit)

Data Types (2/2)

- C++ only
  - string : sizeof(string) = 8 in general on a 64-bit system
  - bool : sizeof(bool) = 1 in general but it is implementation dependent so might differ from 1
  - auto (since C++11): type automatically deduced from initializer
    * Do not confuse with C auto modifier which is the default for all local variables

Sizeof results may vary depending on compiler and operating system (32-bit vs 64-bit)

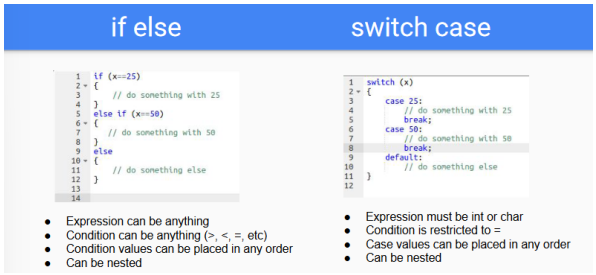
Operator Review

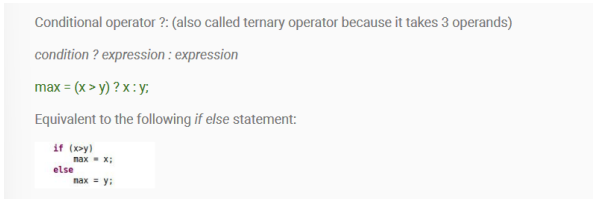
- Assignment operator (=)
  - To assign (from right to left) a value to a variable
  - int x = 42;
- Mathematical operators (+, -, *, /, %)
  - Arithmetical operations: add, subtract, multiply, divide, modulo
  - int x = 13%3;
- Relational operators (==, !=, <, <=, >, >=)
  - Test based on comparison
- Logical operators (&&, ||, !)
  - AND, OR, NOT
- Bitwise operators (&, |, ~, ^, «, »)
  - AND, OR, NOT, XOR, left shift, right shift

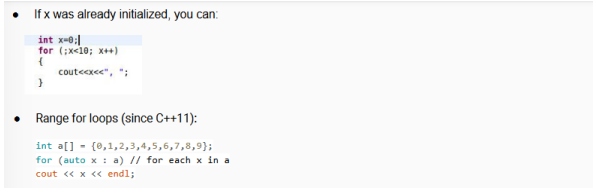
Flow control

- Conditional execution
  - if (condition) ···else ...
  - switch (expression) case constant ···
- Loops (iterate over the same code multiple times)
  - for (initialization; condition/termination; increment/decrement)
  - for (element:array)
  - while (condition) { ···}
  - do { ···} while (condition)

Relational or logical operators can be used to evaluate conditions







- What would the following for loop do?
  - for(; ;)
    * That would be Infinite loop