Object Oriented Programming

Kyung Hee University
Software Convergence
Prof. Daeho Lee

Text

- Richard L. Halterman, "Fundamentals of C++ Programming", Sept. 25, 2018
  https://www.dbooks.org/fundamentals-of-c-programming-1201/

Compiler (IDE)

- Visual Studio (Windows)
- Xcode

https://developer.apple.com/assets/elements/icons/xcode-12/xcode-12-96x96_2x.png

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRoPQQUiJ2S5kTMob6BB1ssutaYFB8mDlimCZiHkvh96lHhYXSnh75M3nYv0D-U4p8WWt0&usqp=CAU

1. Software Development Using C++

1. The Context of Software Development
2. Writing a C++ Program

Software

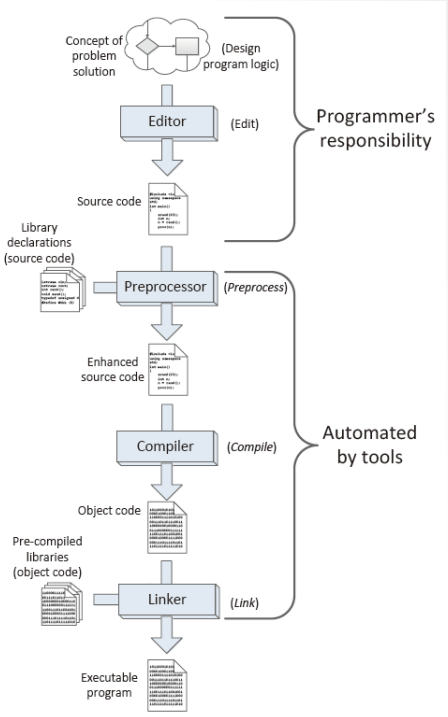
- Software: instruction set
- Binary numbers?
  - Code, encoding, decoding
- Programming languages
  - FORTRAN, COBLE, C, Python, Java, …

Programming Languages

- Machine language
  - Low-level language
  - Directly control a computer's CPU, ALU, registers, and memory
- Assembly language
  - Low-level language
  - Symbolic language
  - Assembler
- High-level language
  - Use natural language
  - Compiler, interpreter
  - C, C++, Python, Java, Fortran, COBOL, …

Development Tools

- Editors
- Compilers
  - Preprocessor
  - Compiler
  - Linker
- Debuggers
  - Checking coding errors (bugs)
- Profilers
  - Dynamic program analysis (memory, complexity, …)
- IDEs (integrated development environments)



Modern C++

- C++
  - Object-oriented language
  - 1972(C) → 1982(C++) → C++ 2.0 → C++ 98 → C++03
  - C++11, C++14, C++17, C++20, C++23

C++98	C++11	C++14	C++17	C++20
○ Templates ○ Exceptions ○ `iostream`-API ○ `std`-library string, containers, algorithms	○ Rvalue references with move semantics ○ Lambdas ○ Variadic templates ○ Uniform initialization ○ Type inference (`auto`) ○ Range-based `for` loop ○ `constexpr` ○ `std`-library APIs support move semantics, smart pointers, concurrency, hash-based containers, `atomic<>`	○ Binary literals ○ Generalized return type deduction ○ Generalized lambda captures ○ Generic lambdas ○ Relaxed `constexpr` restrictions ○ Heterogeneous lookup in associative containers ○ `std`-library `make_unique()`, transformation `_t` alias "shortcuts"	○ Structured bindings ○ `if` and `switch` with initialization ○ Compile-time static `if` `constexpr` ○ Aggregate extensions ○ Fold expressions ○ Mandatory copy elision ○ Class template argument deduction ○ `std`-library `optional<>`, `variant<>`, `any<>`, `byte`, `string_view` ○ File system library ○ Parallel STL algorithms	○ …

https://nohau.eu/wp-content/uploads/Whats-new_-bild.jpg

Simple C++ Programming (1)

```
int main() { // main function - comment
}
```

```
#include <iostream>    // preprocessing directive
int main() {
        std::cout << "This is a simple C++ program!\n";
}
```

- Preprocessing directive
- Library
- Main function
- Statement, ;

Simple C++ Programming (2)

```
#include <iostream>
using std::cout;
int main() {
        cout << "This is a simple C++ program!\n";
}
```

```
#include <iostream>
using namespace std;
int main() {
        cout << "This is a simple C++ program!\n";
}
```

Simple C++ Programming (3)

- namespace
  - Named scope that prevents name conflicts in large projects
- using
  - `using namespace` *`namespace_name`*
  - `using` *`namespace_name::name`*
- std::cout
  - Standard output stream
  - <<: insertion operator

```
#include <iostream>
int main() {
        std::cout << "This is" << " a simple C++ program!\n";
}
```


Simple C++ Programming (4)

```
#include <iostream>
int main() {
        std::cout << "This is a simple C++ program!\n";
        return 0;
}
```

```
#include <iostream>
void main() {
        std::cout << "This is a simple C++ program!\n";
}
```

General Structure of a C++ Program

```
// include directives
int main() {
  // Program statements
  // …
}
```