# Object Oriented Programming

*Kyung Hee University*

*Daeho Lee*

## Text

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

# *1. Software Development Using C++*

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

## Software

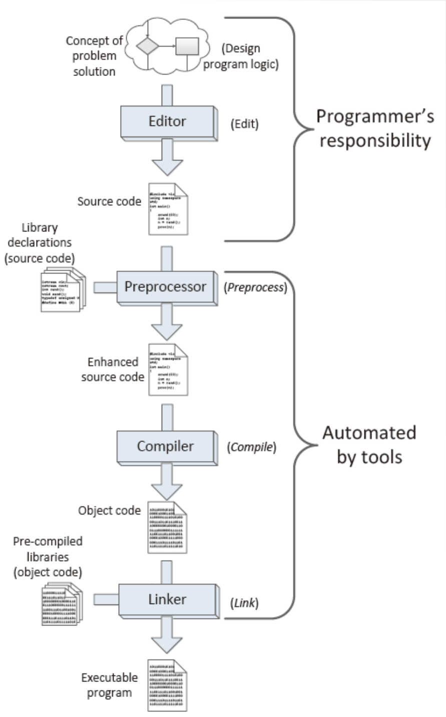
- Software: instruction set
- Binary numbers?
  - Code, encoding, decoding
- Programming languages
  - FORTRAN, COBLE, C, Python, Java, …

# 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

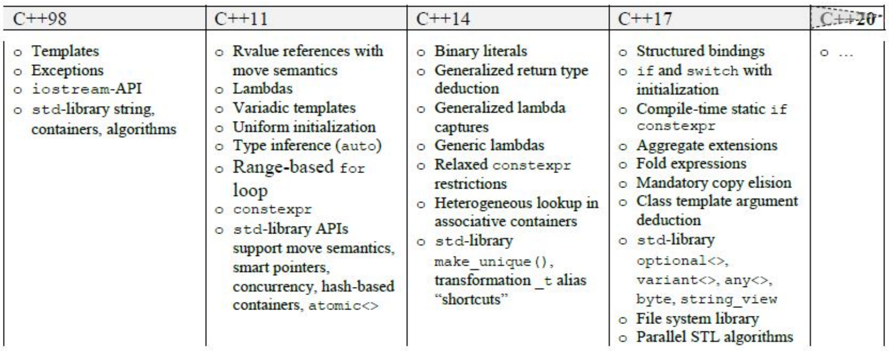
- Editors
- Compilers
  - Preprocessor
  - Compiler
  - Linker
- Debuggers
  - Checking coding errors (bugs)
- Profilers
  - Dynamic program analysis (memory, complexity, …)



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

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

- Visual C++ 2013, C++11
- Visual C++ 2015, C++11/14/17

# Simple C++ Programming (1)

```
int main() {
}
```

```
#include <iostream>
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

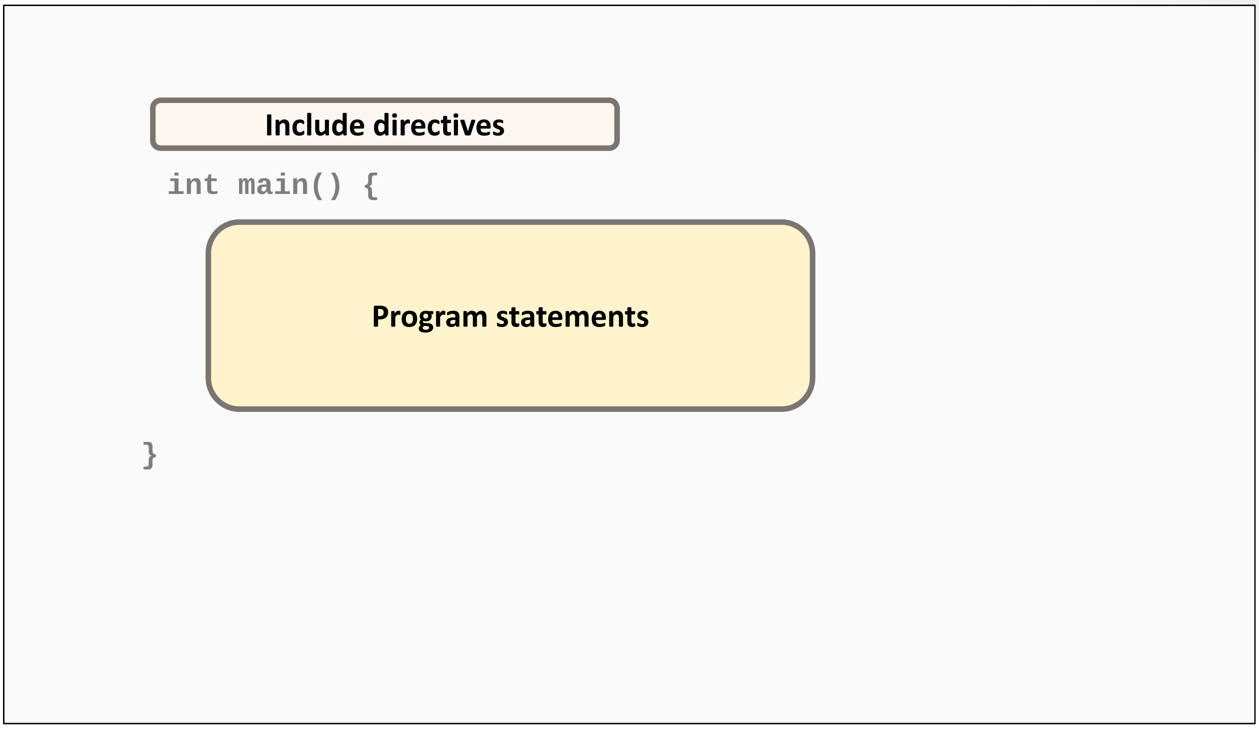
```
#include <iostream>
int main() {
        std::cout << "This is" << " a simple C++ program!\n";
}
```

# 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

```
}
```