OBJECT ORIENTED PROGRAMMING IN C/C++ (ACE1004)

Introduction

Professor: 최 학남

xncui@inha.ac.kr

Office: high-tech 401

Course overview

- Objectives
  - ✓ C++ 명령어의 문법 및 의미를 이해한다.
  - ✓ 주어진 문제를 이해 능력 배양
  - ✓ 주어진 응용 프로그램 개발 능력 배양
  - ✓ 프로그램 개발 도구의 사용 능력 배양
- Scope
  - ✓ 프로그래밍 언어의 6가지 기본문장(치환문, 입력문, 출력문, 조건문, 반복문, 함수)
  - ✓ 4가지 기본 데이타구조(단순변수, 배열, 구조체, 포인터)를 C++ 언어에 대해 차례로 설명하고 실습을 통해 익히도록 한다.
  - ✓ 구조체의 연장으로서의 클래스를 설명하고 실습을 통해 익히도록 한다

Text and Evaluation

- Text
  - ✓서명:C++ How to Program, 8th edition 저자: Deitel 출판사: Prenticehall 출판년도: ISBN
- References
  - ✓서명:jumping into C++ 저자: Alex Allain 출판사: Baker & Tayler 출판년도: 2013 ISBN: 9780988927803
- Lecture Type
  - ✓ Lecture (50%), Practice or Exercise (50%)
- **Evaluation**

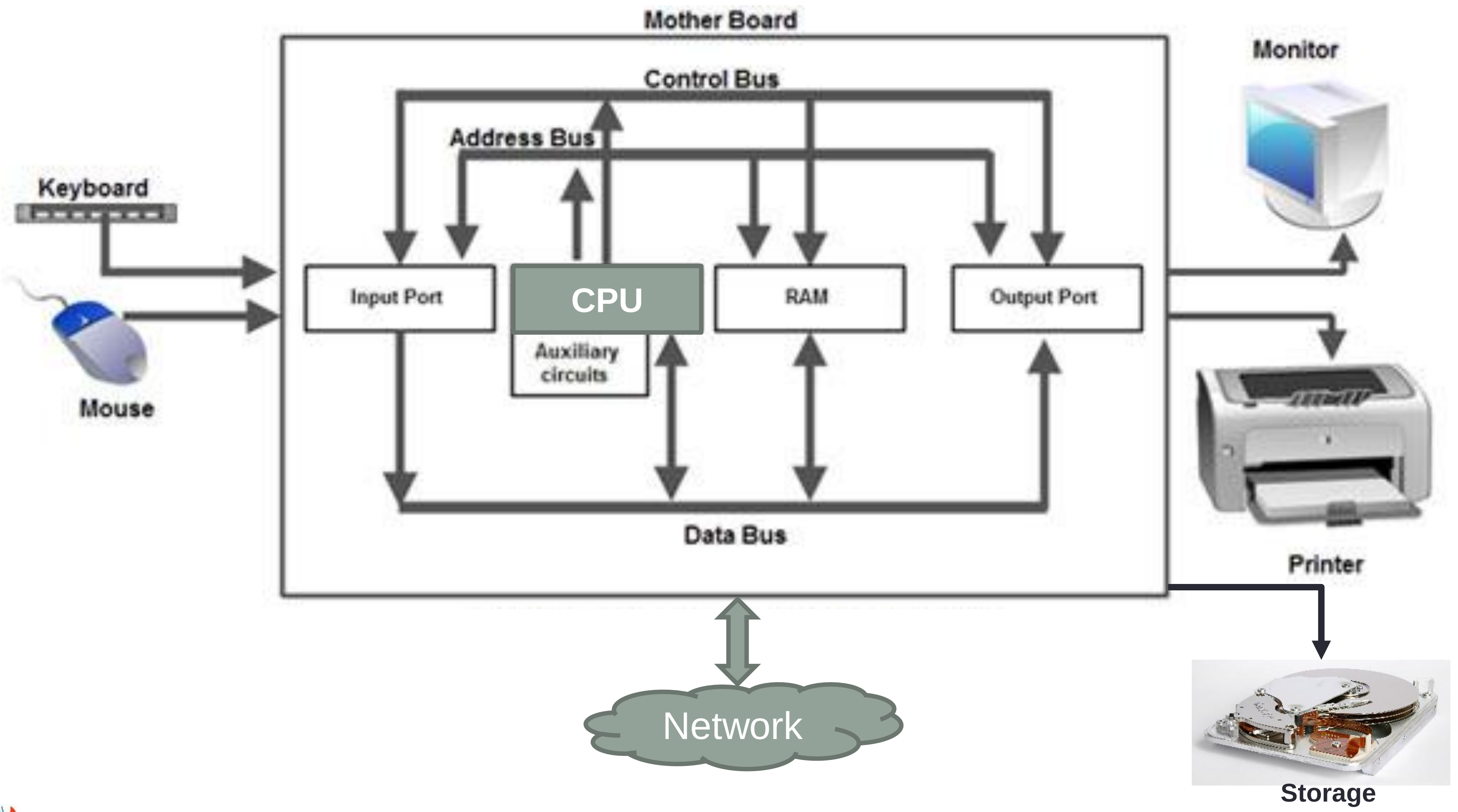
고학년과 저학년 따로 평가함

Midterm exam	Final exam	Attendance	Assignments	Quiz	Total
30 %	**30 %**	**10 %**	**20 %**	**10 %**	**100%**

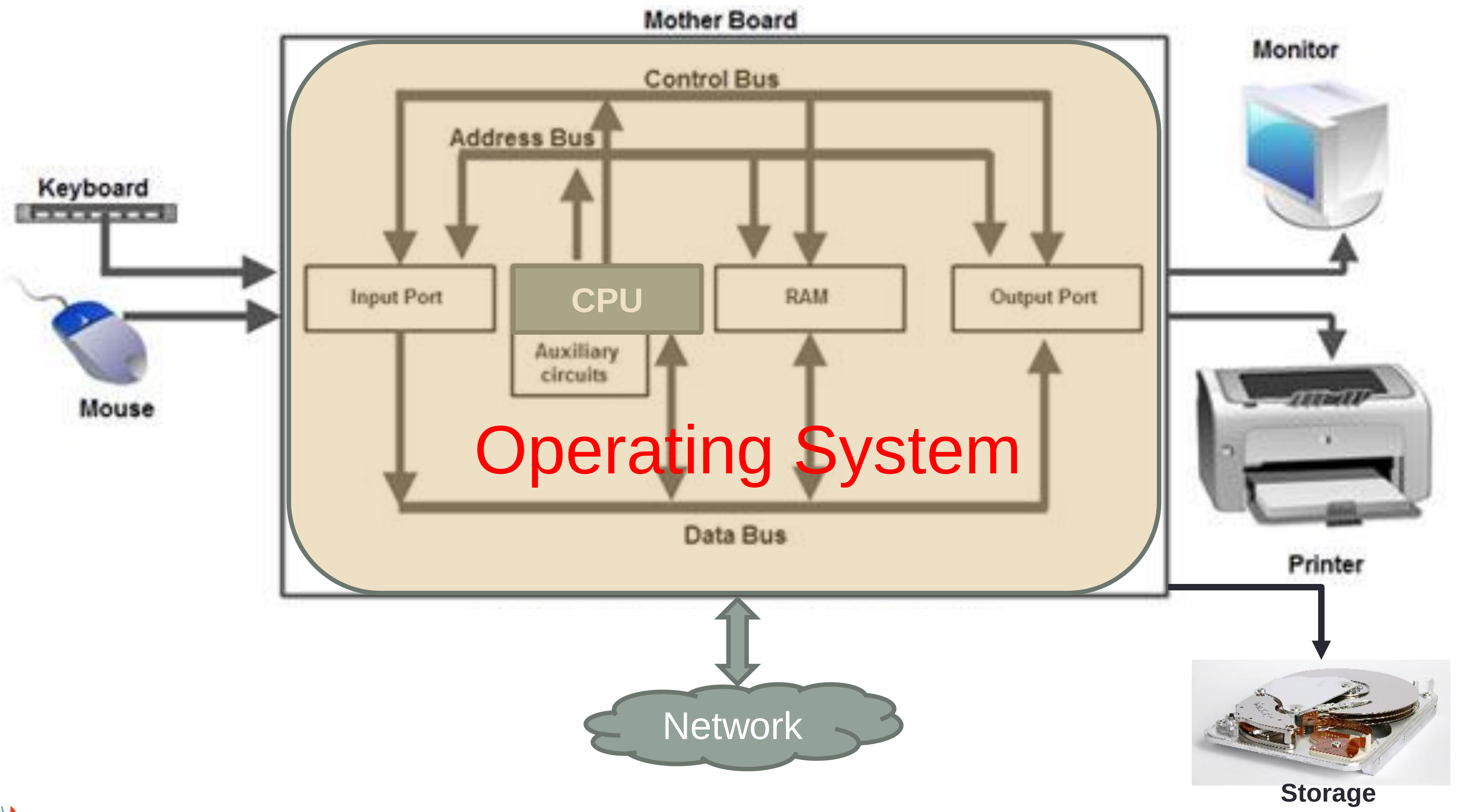
Class Schedule

주차	강의주제
1	Introduction
2	Data types
3	Control structure 1 – while, for, do-while
4	Control structure 2 – if, switch
5	Functions 1 – arguments(or parameters), local (global)variables
6	Functions 2 - function prototype, category of function
7	array 1 - declaration, initialization
8	중간고사
9	Array 2 - 1D, 2D, ND array, array with function
10	Structure 1 – declaration, initialization
11	Structure 2 – structure array, union,
12	Pointer 1 – definition, address of variable, declaration of pointer
13	Pointer 2 – pointer operation, pointer with structure
14	Memory allocation, file input and output
15	기말고사
16	보강주

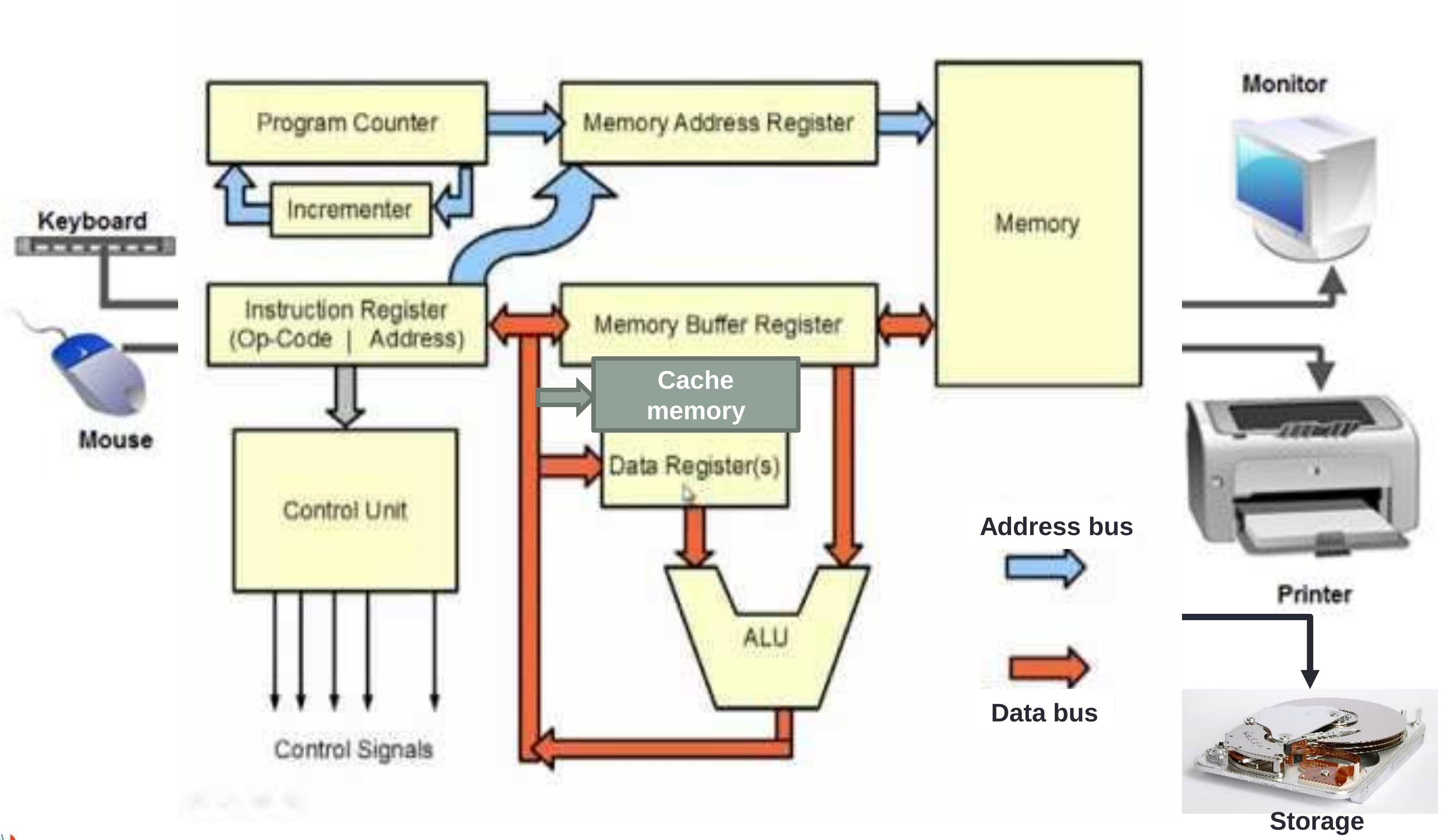
Components in a computer system



Components in a computer system

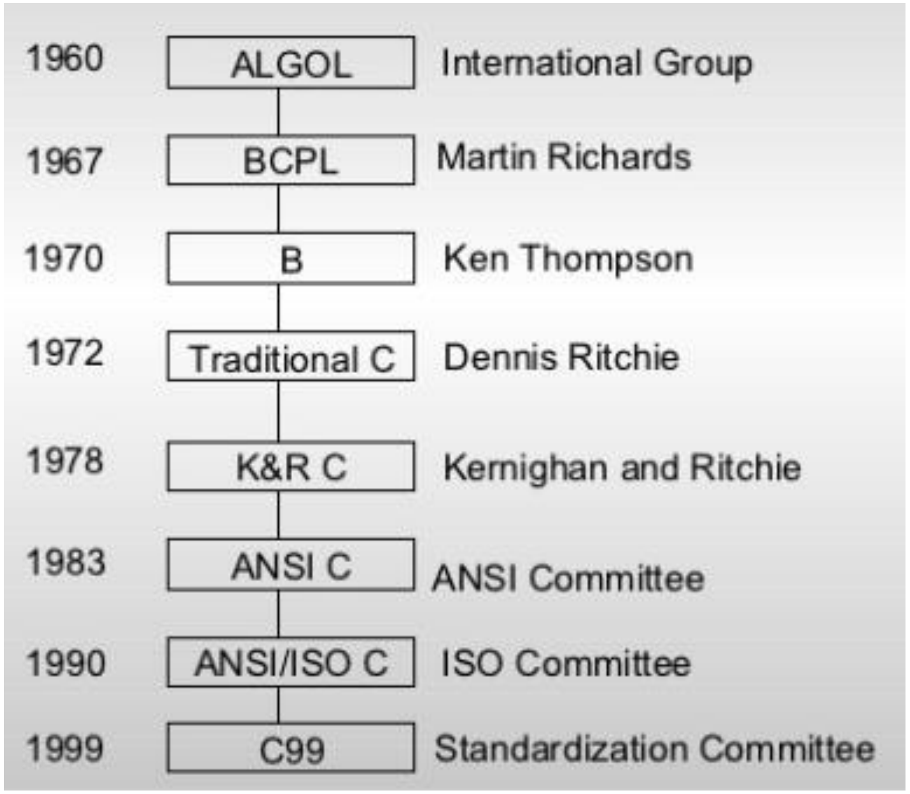


Inside CPU


Program Counter
Memory Address Register
Memory
Monitor
Keyboard
Incrementer
Instruction Register (Op-Code | Address)
Memory Buffer Register
Mouse
Cache memory
Data Register(s)
Control Unit
Address bus
Printer
ALU
Data bus
Control Signals
Storage

History of Programming

- The origin of all modern programming language is **ALGOL** introduced in 1960's.
  - ✓ ALGOL is a structured programming language.
- In 1967, Martin Richards developed **BCPL**(Basic Combined Programming Language) for system software.
- In 1970, Ken Thompson at AT&T Bell Lab created a new language called **B** for UNIX OS.
  - ✓ Both BCPL and B were "typeless" languages.

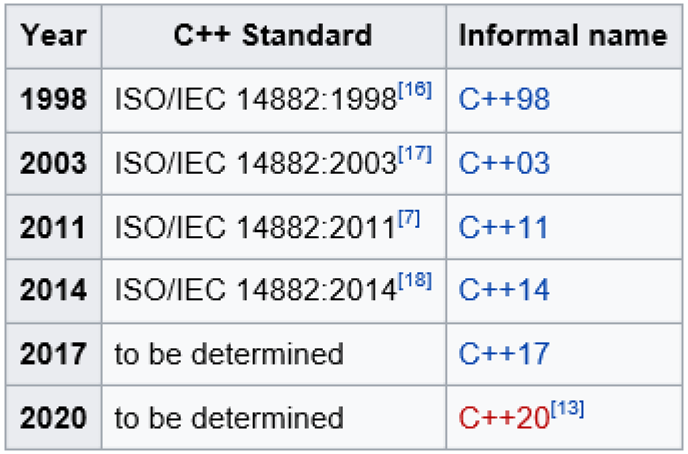


History of C/C++

- In 1972, Dennis Ritchie at Bell Lab developed **C** language for UNIX OS.
  - ✓ Added new features and concepts like “data types”.
- Since then, **C** has been recognized as a standard programming language by ANSI (**ANSI C**) and ISO (**C90**),then **C99**.
- In 1979, Bjarne Stroustrup developed “**C with classes**” to be an object-oriented version of **C**.
- **C++** was named by Rick Mascitti in 1983.
- The first **C++** compiler made available in 1985.
- In 1999, ANSI/ISO **C++** standard approved.

- More recent derivatives: Objective C, C#
- Influenced: Java, Pearl, Python (quite different)

Bjarne Stroustrup



Year	C++ Standard	Informal name
1998	ISO/IEC 14882:1998[16]	C++98
2003	ISO/IEC 14882:2003[17]	C++03
2011	ISO/IEC 14882:2011[7]	C++11
2014	ISO/IEC 14882:2014[18]	C++14
2017	to be determined	C++17
2020	to be determined	C++20[13]

Features of C

- Few keywords
- Structures, unions – compound data types
- Pointers – memory, arrays
- External standard library – I/O, other facilities
- Compiles to native code
- Macro preprocessor
- used for various system programming
- C lacks (부족한 점)
  - ✓ exceptions
  - ✓ range-checking
  - ✓ garbage collection
  - ✓ object-oriented programming
  - ✓ polymorphism

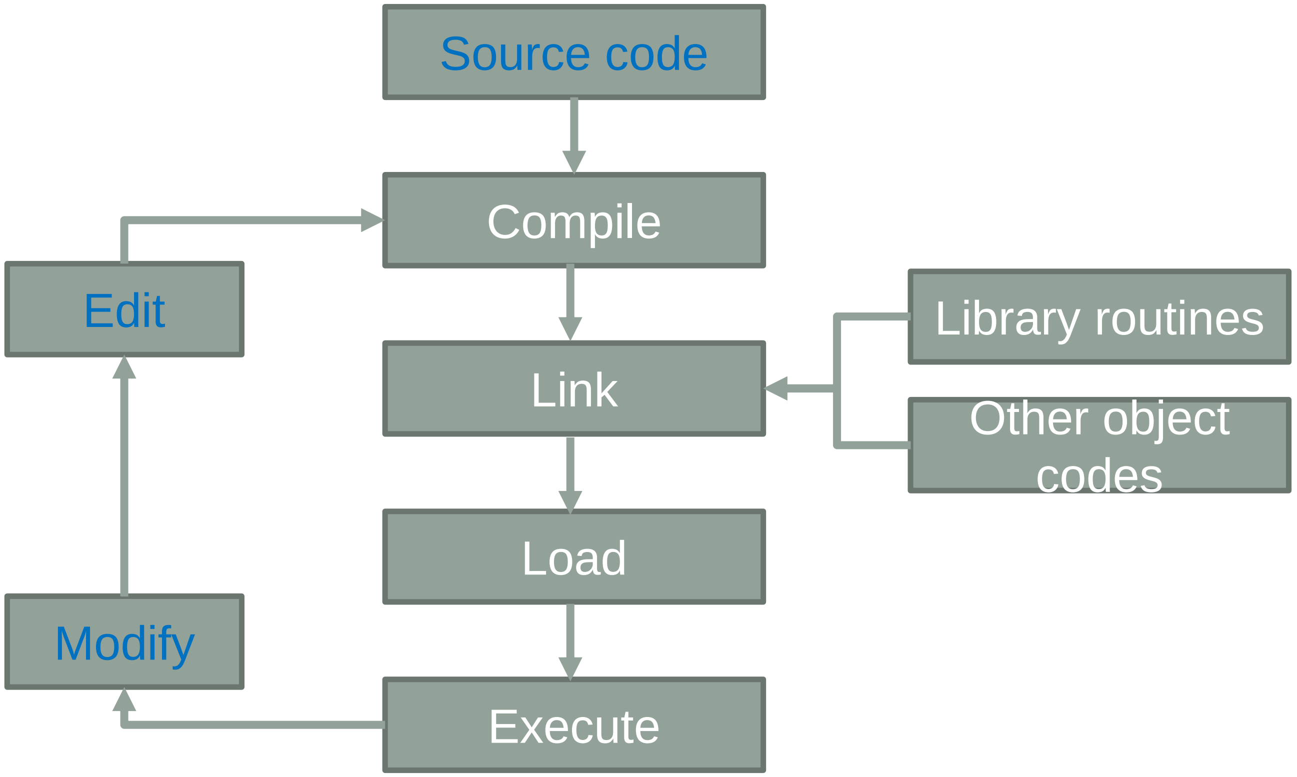
Warning!

✓ No range checking
✓ No type checking at runtime

Features of C++

- Compatible with **C** (almost) – Superset of **C**
- Extends **C** with object-oriented features
- Compile-time checking: strongly typed
- Classes with multiple inheritance
- No garbage collection, but semi-automatic storage reclamation

Program Development Cycle



Definitions

- **Source code**: original computer program written by a programmer
- **Object code**: binary code translated from the source code into machine language by compiler or assembler
- **Compiler**: a type of translator that verify if the source code obeys the programming language grammar (i.e. check if the program is syntactically correct)
- **Linker**: a type of program that combines the object code received from the compiler with files and objects from the library
- **Loader**: a type of program that loads the executable program into the main memory for execution

A Simple C++ Program

```
// Outputs, “Hello world!”
#include <iostream>
using namespace std;

int main() {
  cout << “Hello world!\n”;
  return 0;
}
```

Styles of Commenting in C++

```
// Everything after the
// double slash on the line
// is a comment

/* Everything between the
 * slash-star and the
 * star-slash is a comment
 */
```

Line-By-Line Explanations

```
// Outputs, “Hello world!”
#include <iostream>
using namespace std;

int main() {
  cout << “Hello world!\n”;
  return 0;
}
```

Comment line

Line-By-Line Explanations

```
// Outputs, "Hello world!"
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world!\n";
  return 0;
}
```

Imports the I/O library that std::cout resides in

Line-By-Line Explanations

```
// Outputs, "Hello world!"
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world!\n";
  return 0;
}
```

Allows cout to be used without preceding it with std::

Line-By-Line Explanations

```
// Outputs, “Hello world!”
#include <iostream>
using namespace std;

int main() {
  cout << “Hello world!\n”;
  return 0;
}
```

The main function (every C++ program has exactly one)

Line-By-Line Explanations

```
// Outputs, "Hello world!"
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world!\n";
  return 0;
}
```

Outputs the line of text to standard output

Line-By-Line Explanations

```
// Outputs, “Hello world!”
#include <iostream>
using namespace std;

int main() {
  cout << “Hello world!\n”;
  return 0;
}
```

Returns control to the calling process with an error code of 0

Variable Declarations in C++

- Form:

```
    data_type identifier;
```

- Examples:

```
    int anInteger;
    bool aBool;
```

- C++ is **case sensitive**,
  - ✓ so **`x`** and **`X`** are completely different

Rules for Creating Identifiers

- Identifiers can be any length (use less than 32 characters to be on the safe side)
- First character must be a letter or underscore
- All characters after the first must be a letter, number or underscore
- Identifiers cannot be the same as C++ reserve words (like int, bool, etc.)

Commonly Used Primitive Data Types

Type	Purpose	Example
`int`	Represents integers	`int i = 69;`
`double`	Represents floating-point numbers	`double d = 6.9;`
`char`	Represents characters	`char c = ‘S’`
`bool`	Represents boolean values (`true` or `false`)	`bool b = true;`

C++ Console I/O Stream Objects

`cout`	Sends output to standard output (the console)
`cin`	Takes input from standard input (the keyboard)

Basic Math Operators

+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Modulus (remainder of integer division)

Precedence Rules for Arithmetic

- The `*`, `/`, and `%` operations get evaluated first
- The `+` and `–` operations get evaluated second
- When there is a tie, do operations from left to right

More Notes About C++ Math Operations

- In **`int`** division, any resulting decimal places are truncated (lopped off)
- You can perform a math operation on an **`int`** and a **`double`** (a double will be returned)

Incrementing and Decrementing

`++x`	Preincrement
`x++`	Postincrement
`--x`	Predecrement
`x--`	Postdecrement

Assignment Statements

Assignment	Result
`x += y;`	`x = x + y;`
`x -= y;`	`x = x - y;`
`x *= y;`	`x = x * y;`
`x /= y;`	`x = x / y;`
`x %= y;`	`x = x % y;`

Exercise #1

- Simple c++ style source code

```
#include <iostream>
using namespace std;

void main(){
   cout << "Korea";
}
```

Exercise #1

- Edit, compile, and run a program
  - ✓ Microsoft Visual Studio->Mircrosoft Visual C++
  - ✓ Create an empty project:
    - Select file->new->projects->win32 console application
    - Adjust "Location" for the project directory
    - Give a project name (for example "step1-1")
    - Select empty project
    - Check the creation of the indicated project directory
  - ✓ 3) Write a C++ source file
    - Select file->new->Files->c++ source file
    - Give a file name (for example: main.cpp)
    - Type the above C++ code
  - ✓ 4) Compile
    - Select build->build step1-1.exe
    - Check the creation of step1-1.exe in step1-1/Debug directory
  - ✓ 5) Run
    - Select build->execute step1-1.exe

Exercise #1

- Modifying step1-1 project
  - ✓ Start visutal studio
  - ✓ File->Open Workspace->select "dsw" file
  - ✓ Modify the code and compile/run

HW #1

- Edit, compile, and run the above example.
- Modify the code such that it prints
  - ✓ I am here

- Upload to the e-class (.doc or hwp)
  - ✓ Cover (include the course title + (HW #1), name, sid, date etc.)
  - ✓ Program source
  - ✓ Capture the results
  - ✓ Due date: before next lecture(one week) (e-class)

- Submit the hard copy
  - ✓ Due date: before next lecture(one week) (before the class)

Online complier

- https://www.onlinegdb.com/online_c++_compiler

- http://cpp.sh/