Creative Software Programming

1 - Course Intro

Jongwoo Lim
Fall 2020

Course Information

- Instructor: Jongwoo Lim (임종우)
  - jlim@hanyang.ac.kr
- TA: Jiwoon Bae (배지운)
  - maytryark@gmail.com
- Course Hompage
  - The course page at portal.hanyang.ac.kr (or learn.hanyang.ac.kr)
  - Slides will be uploaded to **Course Content(코스 콘텐츠)** – **Lecture Slides** as soon as it is ready, but they may be updated until just before the lecture.

Course Overview

- In this course, you will
  - Learn the fundamentals of C++ language
    - key concepts of object-oriented programming such as classes, inheritance, and polymorphism
    - references, pointers, dynamic allocation
  - Practice programming skills by writing many exercise programs
  - Practice using development tools and editors in Unix/Linux environment.

References

- Beginner's book:
  - C++ Primer Plus (6th edition), Sthephen Prata

- For deeper understanding:
  - Effective C++ (3rd edition), Scott Meyers
  - More Effective C++, Scott Meyers
  - Effective STL, Scott Meyers
  - Effective Modern C++, Scott Meyers

Prerequisites

- Introduction to Software Design
  (소프트웨어 입문 설계)

- C programming language

- However, we'll have a brief review lectures about C pointers / structures at the beginning of this course.

Schedule (subject to change)

Week	Topic	Tue	Wed	Thr
1	1 - Course Intro / Lab - Environment Setting	9/1	9/2	9/3
2	2 - Review of C Pointer and Structure	9/8	9/9	9/10
3	3 - Review C Pointer&Const, Difference btwn C & C++	9/15	9/16	9/17
4	4 - Dynamic Memory Allocation, References	9/22	9/23	9/24
5	5 - Compilation and Linkage	9/29	9/30	10/1
6	6 - Class	10/6	10/7	10/8
7	7 - Standard Template Library (STL)	10/13	10/14	10/15
8	Midterm Exam	10/20	10/21	10/22
9	8 - Inheritance	10/27	10/28	10/29
10	9 - Polymorphism 1	11/3	11/4	11/5
11	10 - Polymorphism 2	11/10	11/11	11/12
12	11 – Copy Constructor, Operator Overloading	11/17	11/18	11/19
13	12 - Template	11/24	11/25	11/26
14	13 - Exception Handling	12/1	12/2	12/3
15	Make-up Class	12/8	12/9	12/10
16	Final Exam	12/15	12/16	12/17

Lectures & Labs

- Lecture (Tue) + Labs (Wed, Thu)

- Lecture (by instructor)
  - Traditional classroom-based learning.

- Labs (by TA)
  - Time for solving assignment problems by yourselves.
  - TA and undergraduate mentors will help you.

Assignments

- 1 assignment per each lab session.

- TA will help you to solve the problems.
  - You can ask questions!

- Lab1(Wed) assignment due: 23:59 on the day.
- Lab2(Thu) assignment due: 23:59 on next Tue.

Policy for Assignments

- **NO SCORE** for late submissions
  - Start early. Submit before the deadline!

- **F grade** for copying
  - If A copies B's code, both A and B will get F.
  - If A, B, C copies the same code from the internet or elsewhere, they will all get F.
  - Collaboration is encouraged, **but assignments must be your own work. Ask TA for help if you are stuck.**

About Laptop

- Lecture
  - The lecture slides contains many C++ code.
  - I recommend you to bring your laptop at lecture time so that you can compile & run code during the lecture.

- Lab
  - The lab is held in a laptop-only training room.
  - If you want to borrow a laptop, contact the TA by email until the lab in this week.
  - But, I strongly recommend you to bring your laptop at lab sessions.

Grading

Midterm exam	30%
Final exam	30%
Assignments	30%
Attendance	5%
Class attitude	5%

- To avoid F, you have to attend at least **9 lectures && 18 labs**
- Absences from midterm or final exam -> F

CAUTION: Penalty & ABF

- Grade penalty:
  - 3rd year (junior) students: max grade A0
  - 4th year (senior) students: max grade B+

- ABF course:
  - Final grade will be given as one of A, B or F.
  - At least 10% of students will be given F.
  - F is given only by your scores regardless of how much is left to graduate, **so 4th year students should be very careful to take this course.**

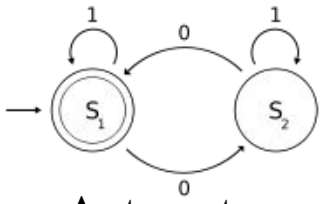
Computer Science

Computer science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications.

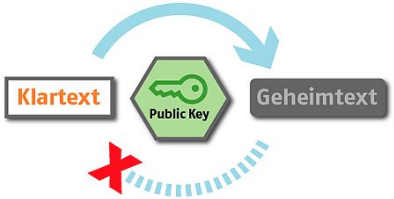
It is the systematic study of the feasibility, structure, expression, and mechanization of the methodical processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to information, whether such information is encoded in bits and bytes in a computer memory or transcribed engines and protein structures in a human cell.

http://en.wikipedia.org/wiki/Computer_science

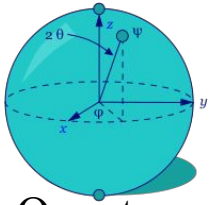
Areas of Computer Science



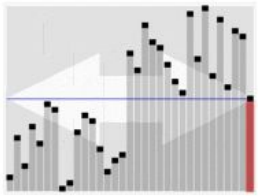
Automata theory



Cryptography



Quantum computing

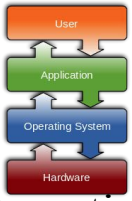


Algorithms

Data structure

Programming language

Compiler design

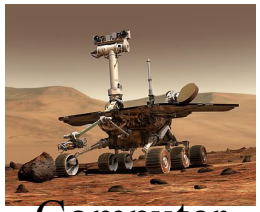


Operating system

Database

Computer network

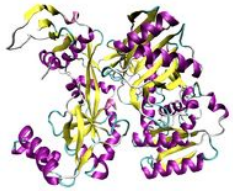
Machine learning



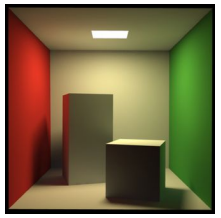
Computer vision

Robotics

HCI



Bioinformatics



Computer graphics

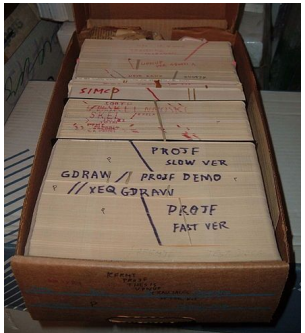
http://en.wikipedia.org/wiki/Computer_science

Programming

You already have learned how to make simple programs in Python and C, but let's revisit the basics.

- Programming language.
- Computer hardware.
- Operating system and development environment.
- Data structure and algorithm.
- Coding style, collaboration, source management (version control).

Programming is the comprehensive process that leads from an original formulation of a computing problem to executable programs. [wikipedia]



Programming Language

A formal language designed to communicate instructions to a computer.

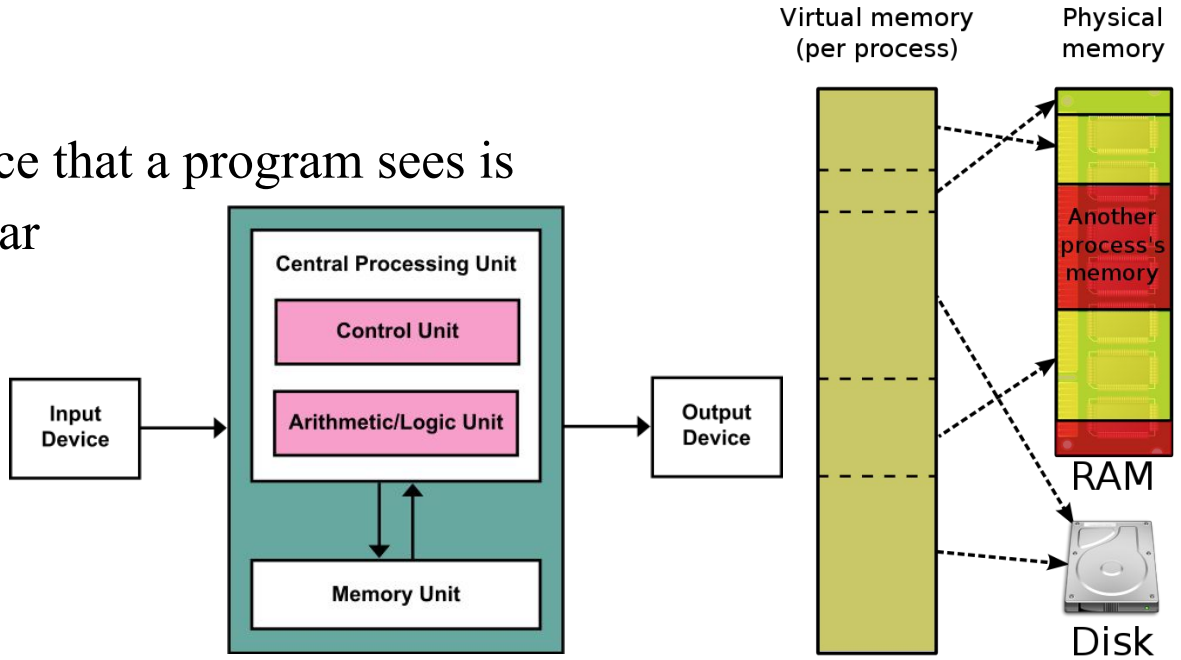
- Syntax and semantics.
  - Language syntax and constructs.
  - Type checking - strongly typed languages.
- Standard library and running environment.
  - Tightly related to the operating system and compiler.

In this class, we learn/use ...

- C++ programming language
- Linux operating system
- G++ compiler, build tools like make, debugger like gdb, and more

Computer Hardware

- Von Neumann architecture.
  - Main components are CPU and memory.
  - Both program and data are stored in the memory (stored-program computer).
  - When a program is executed, the instructions are fetched from the memory, then decoded for execution.
- Virtual memory.
  - The memory space that a program sees is a continuous linear address space.



Operating System and Dev. Environment

- The operating system is in charge of
  - resource management, including CPU and memory (time-sharing and virtual memory),
  - input/output (I/O) operations and file management, and
  - execution and termination of programs.
- Development environment
  - IDE (Integrated -) : Visual Studio, Eclipse, IntelliJ, CLion...
  - Text editor, basic text tools +
    Compiler, linker, debugger, profiler +
    Build system, source management tools, etc.
  - vim, emacs, nano, grep, find, diff, …
    g++, gcc, gdb, cgdb, …
    make, cmake, svn, git, ...

Data Structure and Algorithm

Programming starts with designing data structures and algorithms to solve the given problem.

- Data structure = how is the information stored?
  - Array (fixed-size, dynamically allocated)
  - Linked list (doubly-), stack, queue, deque,
  - Strings,
  - Trees (binary, B-), graphs, ...
- Algorithms = how is the information processed?
  - Sorting, searching, matching (regular expression),
  - Keeping a tree balanced, path finding in a graph, etc.

Collaboration and Source Management

Large programs are almost always built by multiple programmers over long period of time.

- Coding style = how to organize the code?
  - Nothing to do with the program performance, but
  - Very important for human programmers to easily collaborate.
  - Tabs vs spaces, 80-column or not, blanks before/after operators, braces in a new line or in the same line, etc.
- Multiple versions of code edited by multiple programmers.
  - Share the code and manage multiple versions.
  - Review the changes and merge them without breaking the existing system.
  - Release management, etc.

Interactive C++

- Interactive environment helps learning a programming language, libraries, and other tools.
- Cling
  - https://cdn.rawgit.com/root-project/cling/master/www/index.html
  - Linux and Mac OS X (Windows via its Ubuntu support)
- C++ Tutor
  - http://www.pythontutor.com/cpp.html#mode=edit
  - Visualizes the execution of c++ program (experimental)

Interactive C++

- Cling Demo
- C++ Tutor Demo

```
#include <iostream>

int main() {
  std::cout << "hello_world\n";  // Print hello_world.
  return 0;
}
```

Questions – email

- As I will upload the lecture videos every week, please ask questions using email to me or TA. We will try to answer them as quickly as possible, but allow a few days for reply.

- Include [CSP20] in your email subject, and your name and student id in the body. Try to be formal - emails are not chatting. You can write in Korean or in English for questions.

Lastly...

- If you agree on all the policies in the slides, see you this week's lab session!

- If not, please consider taking other classes instead.

Next Time

- Labs in this week:
  - Lab1: Environment Setting, Git / Gitlab, Vim
  - Lab2: g++, make, gdb

- Next lecture:
  - Review of C Pointer and Structure