Creative Software Design

1 - Course Intro

Yoonsang Lee

Fall 2021

Course Information

- Instructor: Yoonsang Lee (이윤상)
  - yoonsanglee@hanyang.ac.kr

- TA: Jeongmin Lee (이정민)
  - j0064423@hanyang.ac.kr

- Course Hompage
  - The LMS course homepage at portal.hanyang.ac.kr (or learning.hanyang.ac.kr)
  - Slides will be uploaded to Lecture Contents **(강의콘텐츠),** probably ***just before the lecture***. So, **download lecture slides at the beginning of each lecture.**
  - If you want to study the lecture slides **in advance**, please refer to last year's lecture slides (They won't change much): https://cgrhyu.github.io/courses/2020-fall-csd.html

Live Online Lecture

- This semester's lectures and labs will be given in “live online lectures” until further announcement.
- All students are required to join lecture and lab sessions for each week on time.

- Questions:
  - In lecture sessions, you can ask questions using slido.com (will be explained later).
  - In lab session, TA will guide you on how to ask questions.

- Attendance check - Lecture session
  - Online quiz submission (will be explained later)
  - Session participation records
- Attendance check - Lab session
  - Minimum session participating time for attendance: **20%** of session duration

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 of C pointers / structures next week.

Schedule (subject to change)

Week	Topic	**Tue**	**Wed**	**Thr**
1	1 - Course Intro / 1 - Lab1 - Environment Setting, 1 - Lab2 - G++, Make, GDB		9/1	9/2
2	2 - Review of C Pointer, Const and Structure 2 - Lab - Gitlab	9/7	9/8	9/9
3	3 – Differences Between C & C++	9/14	9/15	9/16
4	No Class	9/21	9/22	9/23
5	4 - Dynamic Memory Allocation, References	9/28	9/29	9/30
6	5 - Compilation and Linkage, CMD Arguments	10/5	10/6	10/7
7	6 - Class	10/12	10/13	10/14
8	7 - Standard Template Library (STL)	10/19	10/20	10/21
9	Midterm Exam	10/26	10/27	10/28
10	8 - Inheritance, Const & Class	11/2	11/3	11/4
11	9 - Polymorphism 1	11/9	11/10	11/11
12	10 - Polymorphism 2	11/16	11/17	11/18
13	11 – Copy Constructor, Operator Overloading	11/23	11/24	11/25
14	12 - Template	11/30	12/1	12/2
15	13 - Exception Handling	12/7	12/8	12/9
16	Final Exam	12/14	12/15	12/16

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 for each lab session.

- TA and undergraduate mentors 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
  - Submit before the deadline!

- **NO SCORE** for copying
  - If A copies B's code, A and B will get 0 point.
  - If A, B, C copies the same code from the internet, they will all get 0 point.
  - Collaboration is encouraged, **but assignments must be your own work.**

Midterm / Final Exam

- It's best to take the offline exam, but if this is not possible, the exam will be taken online.
  - In this case, you will take a "monitored" online exam.

Grading

Midterm exam	40%
Final exam	40%
Assignments	15%
Attendance	5%

- You will get "F" for more than 5 absences in lectures or 10 absences in the labs.
- Absences from the midterm or final exam -> F

CAUTION: Penalty & ABF

- Grade penalty:
  - 4th year (senior) students: max grade A0

- 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.**

Grading Policy

- Basic principle: Separating the grades where there is a big gap between scores.

- Guideline:

A	25~30%
B	55%~65%
F	10%~15%

Language

- I will mainly use English in lectures.

- But **the most important goal is improving your understanding**, both for English and non-English speakers.
  - So, I'll **"paraphrase" the explanation in Korean for most slides.**

- In lab sessions, TA will try to use English.
  - In the official chat room (in Zoom sessions), you must ask questions in English.
  - But you can ask questions in Korean in personal chatting (in Zoom sessions).

- Now, let's have a brief summary for prev. slides in Korean.

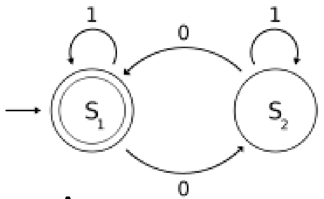
Computer Science

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

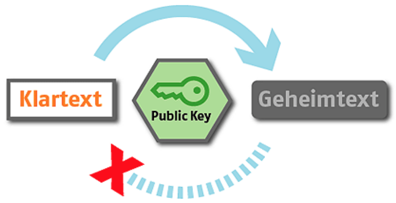
It is the systematic study of the feasibility, structure, expression, and mechanization of the methodical processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to information, whether such information is encoded in bits and bytes in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

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

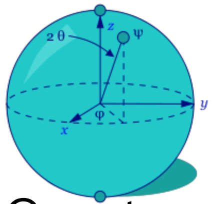
Areas of Computer Science



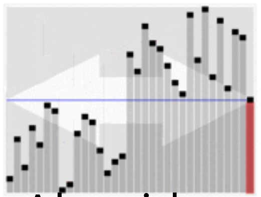
Automata theory

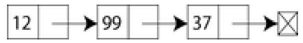


Cryptography



Quantum computing


Algorithms

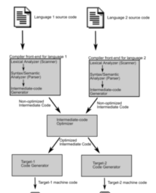


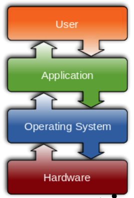
Data structure

```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename, label)
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for  in n, childenumerate(ast[1:]):
            children.append(dotwrite(child))
        print ,'    %s -> {' % nodename
        for  in :namechildren
            print '%s' % name,
```

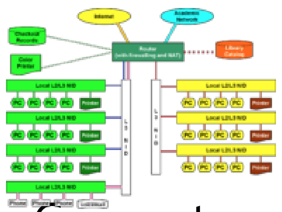
Programming language


Compiler design



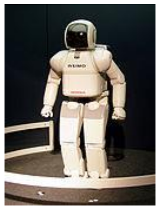
Operating system

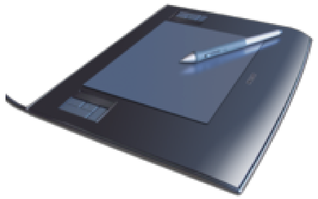
EMPLOYEE
ID : NUMBER
F_NAME : VARCHAR
L_NAME : VARCHAR
SALARY : NUMBER
ADDRESS_ID : NUMBER
FK_EMP_ID
EMP_DATA
EMP_ID : NUMBER
MGR_ID : NUMBER
YEAR_OF_SERV : NUMBER
Database

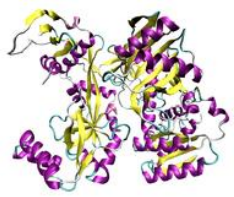

Computer network

Machine learning

Computer vision


Robotics


HCI


Bioinformatics

Computer graphics

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

Programming

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

Let’s revisit the basics related to programming.

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

Programming Language

A formal language designed to communicate instructions to a computer.

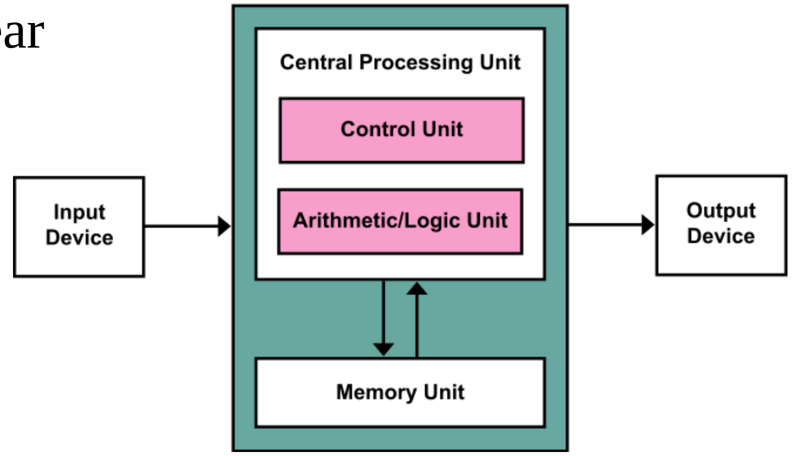
- Syntax and semantics.
    - Language syntax and constructs.
    - Type checking - strongly typed languages.
- Standard library and running environment.
    - Tightly related to the operating system and compiler.

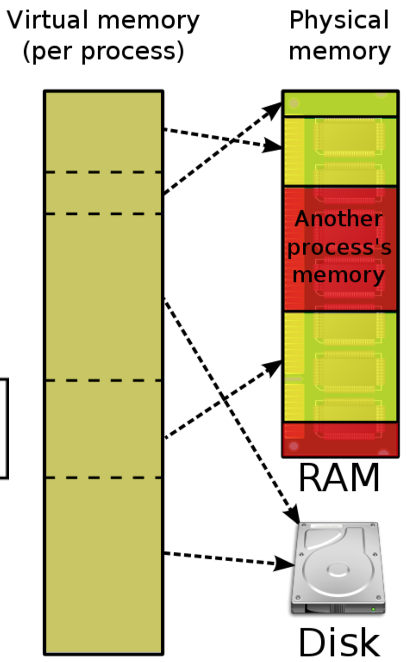
In this class, we learn/use ...

- 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
  - 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, ...
  - IDE (Integrated -) : Visual Studio, Eclipse, IntelliJ, ...

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 format 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.
- Source management (or version control) system - git, mercurial, ...
    - Share the code with multiple programmers 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

- After lecture, if you have questions, ask on the "Q&A Board" ("문의게시판") of the LMS course home.
  - TA will check and respond at least once a day.

- In lecture, we’ll use an online, anonymous Q&A platform - slido.com - to encourage questions.

Just Try Asking a Question!

- Go to https://www.slido.com/
- Join #**csd-ys**
- **Do not bookmark a slido event page** because new events will be created every week!

- Ask any questions **in English!**
  - You can use Google Translator if you have difficulty writing in English.

Questions – Slido.com

- In slido.com, you can
  - **Ask** your own questions
  - **Upvote** other questions

- We’ll use the slido Q&A **only during lecture time.**
  - Not after lecture time
  - Not in lab sessions
  - No written answers

- Please ask questions **anonymously**.
  - Just leave your name blank when post a question.

Quiz & Attendance – Slido.com

- 3 quiz problems for each lecture (using slido.com Polls).
- Very simple questions – you have to submit the answer in two minutes.
- **I'll check attendance using quiz submission.**

Quiz & Attendance – Slido.com

- You **MUST** submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Attendance checking:

Attendance	Number of submissions in the format **- 3 times &&** You are **in the classroom (session)**
Late	Number of submissions in the format – **1~2 times &&** You are **in the classroom (session)**
Absence	Number of submissions in the format – **0 times** || You are **NOT in the classroom (session)**

  - **3 lates count as 1 absence.**

Quiz & Attendance – Slido.com

- If submitting a quiz answer without attending the class (session) is detected,

- I think he or she has been also absent from the previous lecture.

- -> Counted as “Absence” for these two lectures

Just Try a Quiz!

- Go to https://www.slido.com/
- Join #**csd-ys**
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers **in this format** to be counted as attendance.

About Laptop

- Lecture
  - The lecture slides contains many C++ code.
  - During lectures, you can compile, run, and test the code on your laptop.

- ~~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.~~

Classroom Etiquette

- ~~**DO NOT negatively affect other students** in the classroom. For example,~~
  - ~~Doing other things (e.g. games) with your computer~~
  - ~~Using your phone for a long time~~
  - ~~Private conversation~~
  - ~~Sleeping on a desk~~

- ~~May be reflected in "Class attitude" in your grade~~

Lastly...

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

- If not, please consider taking other classes instead.

[Lab] Downloads in advance

- Before joining the first lab session (tomorrow), download the following files in advance

- VirtualBox: http://www.virtualbox.org

- Ubuntu Desktop image: http://releases.ubuntu.com/20.04

- You don't need to download them if you're already using Ubuntu on your laptop.

Next Time

- Labs in this week:
  - Lab1: Environment Setting / g++, make, gdb
- Next lecture:
  - 2 - Review of C Pointer, Const and Structure