Principles in Programming: Orientation & Lecture 1

Course Objectives

- Introduce various subjects
  in computer science
  through puzzles and problems
- Most problems came from ICPC

Introduction

- **Instructor: Euiseong Seo**
  - Associate professor, Software Dept.
  - Web page: http://csl.skku.edu/Euiseong
  - E-mail: euiseong@skku.edu
  - Office: #85564 (5F of Industrial Collaboration Bldg.)
- **Course Homepage**
  - http://csl.skku.edu/SWE2004S13
- **Lab**
  - In Wednesday class
  - #400222
  - Conducted by Three TAs

Course Elements

- **14 Lectures (once every week) (5%)**
- **10 Programming Labs (10%)**
- **3 Individual Programming Homework Assignments (30%)**
- **1 Team Project (15%)**
- **Midterm and Final Exam (40%)**
  - Most questions will be based on labs and assignments
  - Programming

Textbook

- **Programming Challenges by Steven S. Skiena and Miguel A. Revilla – Springer**
  - You can download from the SKKU library

Course Rules (1/2)♪

- **Cheating in exams**
  - machine check
  - will receive an "F" for the course
- **Late homework**
  - 10% penalty per day
- **Cheating on homework**
  - will receive a "0" point
- **1% penalty for missing a lecture class**
- **2% penalty for missing a Lab.**

Course Rules (2/2)♪

- **"not attending" a class includes**
  - not attending a class
  - being late to a class
  - leaving a class in the middle
  - chatting in class
  - having the mobile phone on in class
  - if you sleep, you die!!

Merit Awards

- **Best Homework (For Each Individual Programming Assignment)**
  - Extra 5% of the total point as bonus points
- **Best Team (for team assignment)**
  - Extra 5% for all the members♪
- **Top 5 Students**
  - A dinner at the end of the semester

Course Outline

Problem Solving

- Using C
- Basic Software Engineering

- Programming Patterns
- Problem Solving Techniques

- Practice

What You Need to Solve a Problem by Programming

- **Programming Language Skills**
  - Correct rules
  - Identifying rule violations
- **Programming Skills**
  - Training on programming patterns
  - Training on software engineering methods
- **Problem Solving Skills**
  - Logical thinking

Programming

- Programming Is To Use the "Dumb" Computer To Solve a Problem That A Human Cannot Solve Fast Enough.
- The Computer Needs "Very Very Very" Precise and Detailed Instructions.
- The Instructions Must Be in a Programming Language, Not a Natural Language.

Natural Language

What is 27.2 times 13.8 ?

Programming Languages

- Machine Languages
- Assembly Languages
- High-Level Languages

Machine Language

000000 00001 00010 00110 00000 100000

Add the registers 1 and 2 and place the result in register 6

100011 00011 01000 00000 00001 000100

Load a value into register 8, taken from the memory cell 68 after the location listed in register 3:

000010 00000 00000 00000 00100 000000

Jump to the memory address 1024:

Assembly Language

MOV r0, #0C

load base address of string into r0

LOAD: MOV r1,(r0)

load contents into r1

CALL PRINT

call a print routine to print the character in r1

INC r0

point to the next character

JMP LOAD

load next character

High-Level Language

```
float   length, width, area;

length = 27.2;
width = 13.8;
area = length * width;
```

♪

High-Level Languages

- Over 500 Languages (http://en.wikipedia.org/wiki/List_of_programming_languages_by_category)
- Basic, FORTRAN, COBOL, RPG
- (Algol, Pascal, PL/1), C
- C++, C#, Java (ADA, Smalltalk, Eiffel)
- Perl, TCL, Java Script, PHP, Python, Ruby
- SNOBOL, LISP, (Scheme)
- MATLAB, (APL)
- Shell, Awk, REXX
- SQL, (Prolog), XML, Xquery, XSLT, Postscript, OWL
- 4GL
- UML
- Verilog, VHDL

Executing Programs

1. **Compile**

   Converting programs written in a high-level language into an assembly language or a pseudo code

2. **Assemble**

   Converting programs written in an assembly language into a machine language

- **Interpret**
  - Running programs written in a high-level language with out compiling (one instruction at a time)

Programming Languages

- **"You Can Solve Any Problem Using Any Programming Language"**
- **But Different Languages Are Designed To Serve Different Purposes Better.**
  - FORTRAN for scientific computations
  - COBOL for business data processing
  - LISP for list processing
  - VisualBasic for user-interface programming
  - SQL, PHP for database applications
  - C++, Java for object-oriented software development
  - C for most modern enterprise/scientific applications

Problem Solving by Programming

- **Programming Is**
  - Translating very very precise instructions in some natural language (e.g., Korean, English,...) into some programming language (e.g., C, Java,...) to solve a problem that a human cannot solve easily.
- **So, Before You Program, You Need Very Very Precise Instructions on "How To" Solve the Problem.**
  - You need a "design".
- **Before You Know "How To" Solve the Problem You Need To Know Precisely What The Problem Is ("What To Do").**
  - You need to understand the requirements.

Problem Solving by Programming : Steps

(1) Understand In Precise Detail "What the Problem Is".

Requirements Analysis (Document)

(2) Understand Precisely "How To Solve the Problem".

Basic Design (Document)

(3) For Each Way, Write Down Very Precise and Detailed Instructions (in Korean or English, and Using Diagrams) On "How To Solve the Problem".

Detailed Design (Document)

(4) Choose the "Best" Way.

(5) Translate the Instructions Into a C Program.

Coding (Programming, Implementation) (Document)

(6) Test (Validate, Verify) the C Program

Test Cases (Document)