[프로그래밍 원리] 강의계획서

1 목표와 내용

- 학생들이 프로그램 작성의 기본 원리, 구성 요소, 프로그래밍 미학등을 습득하게 함으로써, 소프트웨어 시스템이 드러내는 복잡성을 손쉽게 다룰 수 있는 능력과 자신감을 익히게 한다. 더군다나, 프로그램이 기계를 사용하기 위한 도구라는 제한된 시각에서 벗어나 기계가 프로그램 실행을 위한 도구라는 시각을 갖추도록 보정해준다.
- 이 강의는 프로그래밍 연습(training)이 아니라 프로그래밍 교육(education)이다. 학생들이 특정 언어의 프로그래밍에 익숙하도록 연습받지 않는다. 보다 중요하게, 명료하고 효과적으로 생각할 수 있게 하는 프로그래밍 교육을 받게 된다.
- 대형 소프트웨어를 2명이상의 팀이 기획하고, 구현하고, 형상관리하는 과정을 프로젝트를 통해 익힌다.
- 대형 소프트웨어를 구성하는 프로젝트를 학기말에 요구한다.
  - 각 숙제들이 프로젝트 결과물의 각 부품들이 되도록 구성.
  - 소프트웨어 기획/구현/관리의 연습을 할 수 있도록 구성.
  - "Extreme programming" "Agile programming" "Programming pattern" "Literate programming"등의 접근법을 익힌다.

다루는 토픽은:

- 재귀와 반복(recursion and iteration)
- 함수로 요약하기(procedural abstraction)
- 데이타로 요약하기(data abstraction)
- 모듈과 계층구조로 요약하기(modularity and hierarchy)
- 물건중심의 프로그래밍(objects and imperative programming)
- 값중심의 프로그래밍(values and applicative programming)
- 타입을 갖춘 프로그래밍(types and typeful programming)
- 실행흐름의 관리(exceptions and advanced control)

2 자료

교과서: [컴퓨터 프로그램의 구조와 해석], 김재우 외, 인사이트, 2007

- *Structure and Interpretation of Computer Programs*, 2nd Ed., Abelson and Sussman, MIT Press, 1996

- *Mythical Man-Month: Essays on Software Engineering*, F. Brooks, Jr., Addison Wesley, 1995

- *Programming Pearls*, 2nd Ed., J. Bentley, 1999

- *Extreme Programming Explained*, Kent Beck, Addison Wesley, 2005

- *The Pragmatic Programmer*, A. Hunt and D. Thomas, Addison Wesley, 2000

& on-line/off-line 자료물들.

프로그래밍: 실습은 DrScheme (또는 MIT Scheme)과 OCaml 시스템을 이용한다. 실습관련 자료는 조교 홈페이지를 참고한다.

3 성적

숙제 40%, 프로젝트 50%, 실습 10%

- 성적은 절대 평가이다.
- 프로그램 숙제가 복제로 판정되면, 숙제의 모든 점수가 50% 감점 처리된다.
- 프로그램 복제여부는 CloneChecker에 의해 자동으로 감별된다.

4 진도 내용

wk1: what to teach, what to expect, policy

wk2: programming elements, primitives, combination, abstraction, evaluation, expression, values, syntax, semantics

wk3: abstraction with procedure, procedure application, typeful programming, scopes of names, syntactic sugars [project 1/4]

wk4: recursive definition, termination check, coverage check, abstraction with higher-order procedures

wk5: compound data, data type introduction/elimination, pairs for list, type for list, type-case analysis

wk6: data abstraction I, programming patterns I, abstraction hierarchy [project 2/4]

wk7: data abstraction II, programming patterns II

wk8: implementation choices, tagged data, defensive programming

wk9: imperative programming, mutating operations, machine-oriented programming [project 3/4]

wk10: understanding program execution in the enviroment model

wk11: value-oriented programming, typeful programming, automatic type checking/inference

wk12: module programming, parameterized modules

wk13: exceptions and advanced control, program correctness, inductive refinement [project 4/4]

□