CSC312 Principles of Programming Languages :

Final Review

1. Introduction
2. Principle components of PL

3. Paradigms of PL design

1. Introduction

2. Principle components of PL

Syntax	Semantics	Name &Scoping	Type
Grammar	Def semantics	Binding	Type concepts
Derivation	Expr semantics	Scoping	Type system
Parsing	Subtile issues	Subtle issues	Example types
Two projects	Project		Project

3. Paradigms of PL design

Functional programming: Haskell ~~(project)~~

Parallel programming: OpenMP/Pthread (project)

λ Calculas: sytax, expression, substitution, reductions

1. Introduction (Ch. 1)

- *What is PL*
- *Main components*
- *Paradigms*
- *Properties for success*

2. Syntax (Ch. 2+3)

- *Grammar*
  - left/right regular grammar, regular expression, deterministic finite automaton (DFA), Chomsky hierarchy, BNF, EBNF, context-free grammar, ambiguous grammar
- *Derivation and Parsing*
  - leftmost/rightmost derivation, LL parser, LR parser, LL grammar, LR grammar, recursive descent parser, FirstSet computation, left dependence graph, parse tree, abstract syntax tree
- *compiler/interpreter structure, tokenization/lexing*

3. Semantics (Ch. 7+8)

- *Methods for specifying semantics*
  - State transitions, operational semantics, (axiomatize statements)
- *Expression semantics: short circuit evaluation, side effects*
- *Copy versus reference*
- *Meanings of various statements*

4. Name, Scope, Binding (ch. 4)

- *Binding of names: static v.s. dynamic*
- *Scoping: static v.s. dynamic*
- *Symbol table stack, referencing environment*
- *L-value, R-value, lifetime, visibility, overloading*

5. Type and Type systems (ch.5+6)

- *Type concepts*
  - type, ~~big/small endian~~, floating-point, type error, static/dynamic typing, type conversion (narrow, widen; implicit, explicit), type equivalence (structural, name), subtypes
  - polymorphism (3 common ways to realize it: overloading, inheritance, generics)
- *Type systems*
  - specification (stylized english, boolean functions)
  - example

6. Functional Languages (ch. 14)

- *View at Program: collection of functions*
- *Properties*
  - state free, referential transparency, lazy v.s. eager evaluation
- *Haskell*
  - Language
  - Special features: polymorphism, function prototype, type classes

7. Parallel Programming

- Pthread
  - thread creation, destruction
  - thread synchronization
    - locks, condition variables
- OpenMP
  - parallel constructs:
    - parallel regions
    - work sharing: loops, sections, tasks
  - synchronizations:
    - barriers
    - locks
    - critical sections

8. OO Language (ch. 13)

- *View at Program: collection of objects that interact.*
- *Foundation*
  - Procedural abstract, data abstract
  - Class and object model
    - OO language key features (encapsulation, virtual methods, inheritance)
    - methods (class methods v.s. instance methods), visibility
    - inheritance (is-a v.s. has-a)
    - polymorphism, template, interface, abstract class, reflection
- *Java*
  - Example