William Paterson University of New Jersey
Department of Computer Science
College of Science and Health
Course Outline

I. Course Title:
CS 382

Programming Languages

3 credits

II. Course Description:

Design issues relevant to the implementation of programming languages. Topics include in-depth study and comparison of mechanisms for sequence control, data structure implementation and run-time storage management; conceptual study of programming language syntax, semantics, and translation; survey of major programming paradigms including procedural, functional, object-oriented, and logical; introduction to language constructs that support distributed and parallel computing.

III. Prerequisites: CS 280 and CS 342 with grades of C- or better

IV. Course Objectives:

1. To study in depth the concepts underlying programming languages and comparatively analyze them
2. To develop an understanding of the characteristics (lexical elements, syntactic structures and semantics) of programming languages.
3. To develop an understanding of programming language features and their applicability in solving different types of problems.

V. Student Learning Outcomes:

After the completion of this course, a successful student will be able to do the following:

1. Describe the major programming domains with their characteristics and for each at least one of the most commonly used languages.
2. List the language evaluation criteria and the characteristics that affect them.
3. Describe and read the characteristics of a programming language in terms of its lexical elements, and syntactic structures.
4. Describe the major categories of programming languages with their characteristics.
5. Describe the major methods for implementing high-level programming languages.
6. Provide the regular expression that defines a simple regular language
7. Use CFG and BNF to specify the syntactic structures of programming languages.
8. Given a CFG or BNF defining a language, provide the derivation of the parse tree of a syntactic structure.

9. Explain bindings and binding times.
10. Describe the scoping rules of variables and program structures.
11. Trace a program (using the different memory segments of the program) with functions' calls.

The course will also reinforce the following students learning outcomes of the university:

a) Effectively express themselves in written and oral form. Measure: exams and projects.
b) Demonstrate ability to think critically. Measure: exams, surveys, and projects.
c) Locate and use information. Measure: projects.
d) Demonstrate ability to integrate knowledge and idea in a coherent and meaningful manner. Measure: exams, surveys, and projects.
e) Work effectively with others. Measure: projects.

VI. Course Contents:

1. History and overview of programming languages: Early language; evolution of procedural languages; non-procedural paradigms and languages.
2. Representation of data types: Choice and representation of elementary data types; specification and representation of structured data types; bindings.
3. Sequence control: Expressions (order of evaluation and side-effects); statement; subprograms and coroutines; exception handling.
4. Data control, sharing, and type checking: Mechanism for sharing and restricting access to data; static vs dynamic scope; parameter-passing mechanisms; type checking.
5. Run-time storage management: Static allocation; stack-based allocation and its relationship with recursion; heap-based allocation.
6. Lexical elements of a programming language: Tokens; regular expressions; finite state automata and scanners.
7. Programming language syntax: Syntactic structures; contex-free grammars, pushdown automata and parsers.
8. Programming language semantics and translation systems: Static semantics; run time semantics; interpreters vs compilers; optimization. Symbol table handling.
9. Programming paradigms: Overview of functional, logic and object-oriented paradigms; designing programs with these paradigms; example programs and applications.
10. Distributed and parallel programming constructs.

VII. Teaching Methods:

1. Classroom lectures
2. Exercises/homework/lab assignments/projects discussions
3. Open Lab. Sessions

VIII. Evaluation:

1. Periodic examinations and final examination
2. Homework assignments.

4. Programming assignments in various representative programming languages.

IX. Textbook:

Concepts of Programming Languages, 7th Ed., Robert W. Sebesta, Addison-Wesley, 2006.

X. Bibliography:

Programming Languages: Principles and Paradigms, 1st Ed., Allen B Tucker & Robert Noonan, McGraw-Hill, 2002.

Programming Languages, Kenneth C. Louden2nd Ed., Course Technology; 2002.

Programming Languages: Design and Implementation, 4th Ed., Terrence W. Pratt & Marvin V. Zelkowitz, Prentice-Hall, 2001.

Concepts in Programming Languages, John C. Mitchell & Krzysztof Apt, Cambridge University Press, 2001.

Comparative Programming Languages, 3nd Ed., Robert G. Clark, Addison-Wesley, 2001.

Programming Language Pragmatics, Michael L. Scott, Morgan Kaufmann, 2000.

Structure and Interpretation of Computer Programs, 2nd Ed., Harold Abelson, Gerald Sussman, & Julie Sussman, McGraw-Hill, 1997.

Programming Languages: Concepts and Constructs, 2nd Ed., Ravi Sethi, Addison-Wesley, 1996.

The Study of Programming Languages, Ryan Stansifer, Prentice Hall, 1994.

The Anatomy of Programming Languages, Alice Fisher & Frances Grodzinsky, Prentice-Hall, 1993.

Essentials of Programming Languages, Friedman, D., M. Want & C.T. Hayes, McGraw-Hill, 1992.

Object-Oriented Design with Applications, Booch, Grady, Benjamin/Cummings Pub. Co., 1991.

Programming Languages: An interpreter-Based Approach, Kamin Samuel, Addison Wesley, 1990.

XI. Prepared by: Dr. Gilbert Ndjatou.

XII. Original Department Approval Date: Spring 1997.

XIII. Revised by: Dr. Gilbert Ndjatou on Spring 2005, previously April 1, 2000.

XIV. Department Revision Approval Date: Spring 2005.