

CS 300: Programming Languages

Marshall University, Spring 2007

Course Syllabus

Contents

1	Course description	2
2	Prerequisites	2
3	Class meeting time and location	2
4	Instructor information	2
5	Course topics at a glance	2
6	Course goals	3
7	Measurable student learning outcomes	4
8	Course assessment and grading criteria	5
9	Due dates for assigned work	5
10	Instructional materials	6
11	Teaching and learning resources	6

1 Course description

In this course, you will learn the structure and features of high-level programming languages and understand how they have been implemented in several actual programming languages. Emphasis is not on learning a particular language (or set of languages) in great detail, but on the process of learning how to learn new programming languages. However, you will write simple programs using the following languages: C/C++, MATLAB, Python, C#, SAS, and possibly Mathematica.

2 Prerequisites

- Intermediate-level proficiency in programming using an object-oriented language such as Java or C++.
- Access to WebCT Vista from your home will be quite helpful.

3 Class meeting time and location

- 9.30 - 10.45 AM, Tuesdays and Thursdays, GH 211.

4 Instructor information

Name: Dr. Venkat N. Gudivada, Associate Professor, Engineering & Computer Science.

Phone and email: 304-696-5452; gudivada@marshall.edu. Please use WebCT Vista email for course related inquiries.

Office Location: GH Room 205A.

Office Hours: MW 2.30 PM - 3.30 PM and TuTh 1.30 PM - 3.30 PM. Other times by appointment.

5 Course topics at a glance

- A survey contemporary programming languages and their features
- Syntax and semantics
- Binding, scope rules, and type safety
- Abstract data types, algebraic types, and type inheritance and inference
- Expressions and operators
- Control structures
- Method/subprogram implementation
- Concurrency control
- Exception handling
- Generic programming
- Functional programming
- Object-oriented programming

6 Course goals

- Describe the syntax and semantics of contemporary programming languages
- Explain fundamental concepts including expressions, operators, operator precedence and associativity, expression evaluation, fundamental control structures, binding times, scope rules, abstract data types, type inheritance and type safety.
- Compare and contrast various programming languages in terms of their structure, and features; determine their suitability in an application context.
- Describe language runtime support for method implementation, invocation, and return; and discuss parameter passing techniques.
- Explain concurrency concepts and describe language support for concurrent programming.
- Explain exception handling concepts and language support for processing exceptions.
- Compare and contrast procedural, object-oriented, generic, functional, and declarative programming paradigms.
- Write simple but substantial programs using languages based various programming paradigms: C/C++, MATLAB, Python, C#, and possibly Mathematica.

7 Measurable student learning outcomes

A high course grade in CS 300: Programming Languages requires that the student demonstrate most or all of the following:

1. **Explains** the technical and pragmatic reasons for the proliferation of programming languages.
2. **Articulates** the distinguishing characteristics different programming paradigms and language constructs required for supporting them.
3. **Clearly distinguishes** between programming language syntax and semantics, specification schemes for specifying them, and their role in learning new languages and impact on programming productivity and software maintenance.
4. **Understands** the issues related to early and late binding, how scope rules govern the visibility of variables and program units, and issues in type safety and type casting.
5. **Understands** the notions and applications of abstract data types, algebraic types, and type inheritance and inference; compares and contrasts them.
6. **Demonstrates** knowledge of techniques for expression evaluation, and role of operator precedence and associativity in expression evaluation.
7. **Understands** the nuances in the variations of the fundamental control structures and is **knowledgeable** in choosing a right control structure for a given context.
8. **Demonstrates** an understanding of the need for and issues in concurrency control and **programmatically illustrates** concurrent access to shared resources.
9. **Demonstrates** an understanding of the need for and issues in runtime exceptions and **programmatically illustrates** exception raising and exception processing.
10. **Articulates** the functions of language runtime system and explains how it: determines visibility of variables and program units; implements method invocation and return; performs memory management; and supports concurrency and exception handling.
11. **Recognizes** the need for generic programming, **gained** basic understanding of developing generic software components.

Successfully developed non-trivial programs in C/C++, MATLAB, Python, C#, and Mathematica.

8 Course assessment and grading criteria

The course assessment is based on the following measures:

- Several programming assignments: 50%
- Individual student conference with the instructor (should occur by 30-Mar-2006): 5%
- Two midterm exams: 30%
- Final exam: 15%

Assignment of letter grade

<i>Score</i>	<i>Letter Grade</i>	<i>Remarks</i>
≥ 90	A	Achievement of distinction
$\geq 80 \ \& \ < 90$	B	Competent and professional work
$\geq 70 \ \& \ < 80$	C	Below average performance
$\geq 60 \ \& \ < 70$	D	Patently substandard work
< 60	F	Unsatisfactory work

Note that A grades are awarded only to those students who have demonstrated distinctive performance in the course.

9 Due dates for assigned work

Due dates for various assigned work are indicated either on the handouts or WebCT Vista. No late submissions will be accepted.

10 Instructional materials

Required textbook Kenneth C. Louden. *Programming Languages (Second Edition)*. Thompson (Brooks/Cole), 2003. ISBN: 0-534-95341-7.

Additional resources [Textbook author's Web site](#)

11 Teaching and learning resources

It is important to visit WebCT Vista for up-to-date information about the course. It hosts all the course materials including assignments, handouts, lecture notes, and reading materials. Also, you will use the Vista for submitting all your work.

References

- [1] Kenneth C. Louden. *Programming Languages*. 2nd edition. Thompson (Brooks/Cole), 2003. ISBN: 0-534-95341-7.
- [2] Tony Gaddis and Barret Krupnow. *Staring out with C++ (brief edition)*. Pearson education, 2007. ISBN: 0-321-41291-5.
- [3] David Hill and David Zitarelli. *Linear algebra labs with MATLAB*. 3rd edition. Pearson education, 2004. ISBN: 0-13-143274-5.
- [4] Mark Lutz. *Programming Python*. O'Reilly, 2006. ISBN: 0-596-00925-9.