COURSE DESCRIPTION

Dept., Number	CSci 251	Course Title	Programming for Engineering and Sciences
Semester hours	3	Course Coordinator	Charles M. Jenkins, Instructor

Current Catalog Description

Algorithm development and structured programming techniques; numerical and graphical applications related to engineering and scientific problem solving.

Textbook

Chapman, S.J. *Fortran 95/2003 for Scientists and Engineers, Third Edition*. McGraw–Hill Higher Education, Boston, Massachusetts, 2008. ISBN 978–0–07–319157–7.

References

Course Outcomes

This course introduces students to computer programming using a general–purpose programming language. It emphasizes structured problem–solving in an engineering and scientific context. By the end of this course, students should be able to develop complete, understandable, maintainable programs. Specifically, students should be able to:

1. Explain the rudiments of a computer's architecture.
2. State clearly the purpose of a computer program.
3. Define the inputs and outputs of a program.
4. Develop algorithms.
5. Develop a plan for testing a program.
6. Identify and mitigate risks to the successful implementation of a computer program.
7. Identify and eliminate errors in a computer program.
8. Implement algorithms in computer program.

Relationship between Course Outcomes and Program Outcomes

This course is primarily taken by students in Chemical, Civil, Electrical, Geological, and Mechanical Engineering, as well as students in Forensic Chemistry. It is only rarely taken by Bachelor of Science in Computer Science students and may be applied to their degree requirements only under exceptional circumstances with the permission of the Department's Undergraduate Committee. The course outcomes contribute to the program outcomes as shown below:

1. Explain the rudiments of a computer's architecture. Outcome a
2. State clearly the purpose of a computer program. Outcomes b, c
3. Define the inputs and outputs of a program. Outcomes b, c
4. Develop algorithms. Outcome b, c
5. Develop a plan for testing a program. Outcome c
6. Identify and mitigate risks to the successful implementation of a computer program. Outcomes c, i, j
7. Identify and eliminate errors in a computer program. Outcome c
8. Implement algorithms in computer program. Outcome c

Prerequisites by Topic

This course has a corequisite of Math 261, Unified Calculus and Analytical Geometry I

Major Topics Covered in the Course

1. The rudiments of a computer's architecture.
2. A structured approach to solving problems with computer programs.
3. Documenting a programming.
4. Formatting a program.
5. Performing calculations involving the following operations:
    a. Arithmetic (e.g., addition, multiplication, modulo division, and absolute value)
    b. Logarithmic and exponential
    c. Trigonometric (e.g., sine and cosine)
    d. Boolean (e.g., AND, OR, and NOT)
6. Executing a sequence of actions
7. Selecting a course of action ("branching")
8. Repeating a course of action ("iteration")
9. Declaring and using floating point, integer, and Boolean variables
    a. Scalar (single–valued)
    b. Array (multi–valued)
    c. Constants
10. Reading unformatted input from the keyboard and from files (using file redirection)
11. Writing formatted and unformatted output to the screen and to files (using file redirection)
12. Designing and coding subprograms to create larger, more complex, well-structured programs.

Assessment Plan for the Course

To demonstrate their mastery of the course topics, students:

1. Complete five or more homework assignments and two or more graded laboratory exercises
2. Take four quizzes
3. Complete five or more computer programming assignments of increasing difficulty
4. Take a three–hour practical exam involving the creation of an entire computer program to solve a problem posed by the instructor

The course is overseen by the School of Engineering's Engineering Computing curriculum committee, for which the usual instructor for CSci 251 (or another full-time computer science faculty member) serves as Chair. The instructor administers a Student Perception of Course Effectiveness survey in each section of the class. The Engineering Computing committee recommends appropriate changes to the course in response to the survey results and in response to the School's curricular requirements.

How Data in the Course are Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	1		Software design	1	
Data structures			Concepts of programming languages	1	