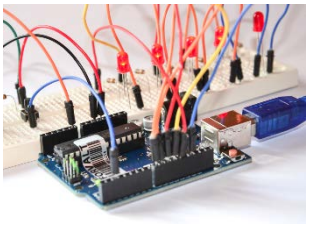
Programming for Scientists and Engineers

Fall 2018

Engineering 40

Section #55104



Instructor: Dr. John Heathcote
Office: FEM-1B (in the math study center)
Phone: 638-0300 ext. 3215
e-mail: john.heathcote@reedleycollege.edu

Class Times: MW 2:00-4:50pm
Room: FEM 3

Welcome to ENGR 40 , Programming for Scientists and Engineers! This will be a fun class where we can apply what we learn about computer science and programming to solve engineering problems! I look forward to working with you in this very hands-on course!

Office Hours: MW, 10:00-11:50AM, F, 11:00-11:50AM

Prerequisites: MATH 4B

Catalog Description: This course introduces the use of C++ programming language to solve engineering and applied science problems. It includes a systematic development of program structure, specification, testing and debugging. Lab assignments include traditional program development as well as the interface of software with the physical world.

Textbook: Gaddis, **Starting Out with C++, from Control Structures through Objects**, Brief Version, 8th Edition, Pearson.

Grading:	70%	Programs, Written Assignments, and Projects
	30%	Tests and Final Exam

Programs, Written Assignments, and Projects:
The major goal of this class is to learn to program. As such, a large portion of your grade will come from programs that you will write and execute during your lab periods. In addition, you need to understand the terminology and design of computer programs. So, during your textbook reading assignments, you will be asked to complete written assignments based upon your reading. Finally, as a way of displaying your ability in programming, you will be asked to complete longer (and hopefully more interesting) programs. These projects will also be a large part of your grade.

Tests and Final Exam: There will be approximately four tests during the term. Each test will cover 2-4 chapters. During these tests, you will be asked to show your understanding of computer programming terminology and topics as well as your ability to write computer programs.

Grading Policies: Late assignments will not receive full credit. If you are unable to take a test on the assigned date, you must contact the instructor ahead of time to arrange a different time. (If you are sick or if something else happens on the day of the test, call or email the instructor!)

Grading Scale:		
	90-100%	A
	80-89.9%	B
	70-79.9%	C
	60-69.9%	D
	<60%	F

Accommodations for Students with Disabilities:
If you have a verified need for an academic accommodation or materials in alternate media (i.e., Braille, large print, electronic text, etc.) per the Americans with Disabilities Act (ADA) or Section 504 of the Rehabilitation Act, please contact me as soon as possible.

Add Date:	Friday, August 31	Last day to add a course
Drop Date:	Friday, October 12	Last day to drop this course
Holidays:	Monday, September 3	Labor Day Holiday
	Monday, November 12	Veterans Day Holiday
	Thursday-Friday, November 22-23	Thanksgiving Holiday
Final Exam:	Monday, December 10th, 2:00-3:50PM	

Student Learning Outcomes:

Upon completion of this course, students will be able to:

1. Use a high-level programming language to design, implement, test, and debug programs that use each of the following fundamental programming constructs: basic computation, simple input/output, standard conditional and iterative structures, user-defined functions, arrays, pointers, classes, external data files, and the use of interfaces with the physical world.
2. Use pseudocode and a high-level programming language to design, implement, test, and debug algorithms for solving simple problems.
3. Summarize the evolution of programming languages and describe the software development life-cycle.
4. Demonstrate different forms of binding, visibility, scoping, and lifetime management.
5. Demonstrate how one may combine software and hardware components in order to respond to physical phenomena and manipulate the physical world.
6. Solve application problems in science and engineering.

Objectives:

In the process of completing this course, students will:

1. Analyze and explain the behavior of simple programs
2. Modify and expand short programs that use standard conditional and iterative control structures and functions.
3. Design, implement, test, and debug programs that use each of the following: basic computation, simple input/output, standard conditional and iterative structures, user-defined functions, arrays, pointers, classes, and external data files.
4. Choose appropriate conditional and iterative constructs for a given programming task.
5. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
6. Describe the mechanics of parameter passing.
7. Create algorithms for solving simple problems.
8. Use pseudocode and flowcharts to describe algorithms.
9. Use strategies that are useful in debugging.
10. Summarize the evolution of programming languages illustrating how this history has lead to the paradigms available today.
11. Describe how data is stored in a computer's memory.
12. Explain the value of declaration models, especially with respect to programming-in-the-large.
13. Identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size.
14. Discuss type incompatibility.
15. Demonstrate different forms of binding, visibility, scoping, and lifetime management.
16. Defend the importance of types and type checking in providing abstraction and safety.
17. Use basic development tools and IDE's for simple software development interfaced with the physical world.
18. Understand and modify software developed by others and evaluate the quality of the modifications.
19. Demonstrate the interplay between software and the physical world.
20. Follow simple software QA procedures.