Course Description
CMPS 221

Programming Fundamentals

Catalog Description:
Introduces the fundamentals of procedural programming and an introduction to object-oriented programming. Topics include data types, control structures, functions, arrays, and standard and file I/O. The mechanics of compiling, linking, running, debugging and testing within a particular programming environment are covered. Ethical issues and an historical perspective of programming within the context of computer science as a discipline are discussed.

Prerequisite:
Satisfy ELM and pre-calculus readiness.

Corequisite:
Take CMPS 150 or CMPS 215 concurrently. Computer Science majors should take CMPS 215, if possible.

Units:
5

Instructor:
Marc Thomas

Goals:
To provide the student with a basic knowledge of computer programming fundamentals, including software design, algorithm development, and algorithm implementation methods.
To develop computer programming skills in a high-level programming environment.
To provide the student with the opportunity to work in the UNIX environment.
To provide an overview of the many areas of computer science.
(Laboratory) Become proficient in writing procedural programs and simple object-oriented programs in the C/C++ programming language.

Required Texts:

Recommended Reference Texts for Later Work:

Topics:
The first principle in learning to program is to write lots of short programs. Try things out yourself and don’t wait for an assignment or lab. Look at the sample programs in /usr/fac/marc/public_html/code/cs221 which you can also access from my web page http://www.cs.csusb.edu/~marc/classes.html and try modifying them for practice.

Introduction to the C/C++ programming language and how C++ is (essentially) a superset of C. The main module, header files, cin (standard input), cout (standard output), and cerr (standard error), prompts and constant strings, character variables char, integer variables int and double/float variables, the value of a
variable, the address of a variable, algebraic syntax, the if statement, blocking with { and }.

“Structured” programming, discussion of if ... else, nesting, while, pre and post-increment operators. Handling crashes (and core dumps) with a debugger. Some work with greatest common divisor. The separate stages of preprocessing, (two-pass) compilation, and linking.

More program control structures, for, switch, do ... while, break, and continue, logical operators for and, or, and negation, handling exceptions. Use of a sentinel, input and output redirection, type conversion. I/O in more detail, <iostream>, the relationship between I/O in C++ (object-oriented) and I/O in C (stream oriented). Use of

ios::sync_with_stdio(); for compatibility and use of printf().

Functions, both built-in and user defined, passing parameters by (first) value and (later) reference. The important Math Library Functions. Scoping, local vs. global, and program structure. One simple example of a recursive function. C++ specific features such as inline functions and function type overloading. Problems of overflow, especially with int variables. Examples with prime and composite numbers, solving quadratic equations, etc.

Introduction to data structures: the array. The use of arrays in mathematical and scientific programming, arrays of double-precision reals, simple sorting as data is being input. The problems of overwriting array boundaries. Some simple statistical programs with average, variance, median, etc. Implementation of character strings as null-terminated arrays.

Pointers, using pointers safely (this topic will be greatly elaborated upon in Data Structures and Algorithms). String processing. File input and output with <fstream>.

Introduction to structures, classes, encapsulation, member functions, constructors, and object-oriented design (this topic will be greatly elaborated upon in the course on Object-Oriented Programming — CMPS 222).

Laboratory:

The laboratory session will parallel the lecture, illustrating the principles and familiarizing the student with a real program development environment. Programs will be written in C/C++. There will be some discussion about differences, preferences, and options with regard to program development tools such as editors, compilers, debuggers, integrated environments, and the operating system platform.

Grading:

Two midterms will be given, each worth 25%. I do not give make-up midterms; for an excused absence I count the other grades proportionately higher. One final exam, comprehensive but emphasizing the later material, will be given. It is mandatory and worth 25%. Homework and lab work are together worth the remaining 25%.