Fall 9-1-2000

CS 221.01: C/C++ Programming

John J. Blum
The University Of Montana

CS 221 – C/C++ Programming
Course Syllabus
Fall Semester 2000

Instructor:	Mr. John J. Blum
Teaching Assistant:	TBA
Office:	Social Sciences 404
Office Hours:	No set hours. Meet by appointment.
Email:	jblum@cs.umt.edu
Textbook:	C++ How To Program, 3rd Edition Dr. Harvey M. Deitel & Paul J. Deitel Prentice Hall www.bookpool.com

Course Synopsis: Welcome to the fundamentals of C & C++ programming. The intent of this course is to familiarize students with analyzing and solving problems using programming constructs inherent in C & C++. Students are expected to have successfully completed their responsibilities in CS131 & 132, Fundamentals of Computer Science I & II respectively, which formed a basis to this course. This course is the continuation for CS majors in their studies of Computer Science.

Before delving into the object-oriented realm of C++, we will build our foundation in C emphasizing structured procedural programming. A short review of basic programming constructs serves only to better understand design principles and proper program construction using C. We will use C to perform many of the low-level tasks (memory manipulation, stream processing, etc.) essential to your studies in other courses. Other applications of C can be seen in the development of operating systems (i.e. Linux), compiler implementations (i.e. javac), real-time systems (i.e. CAD - Logistic Systems, Missoula, MT), and many other examples without a doubt.

Following C will be extensive study of object-oriented concepts and design using C++. Students will learn to model and represent problems using objects. Two important features of objects are the attributes (data members) and behaviors (functionality) it possesses, and how they are used to represent a solution to the problem. Learning C++ will also reinforce what knowledge should have been acquired during CS131 & 132, and to have a new found appreciation of Java.

The course will use the textbook as a guide to teach the course. Information used in class that is not contained in the book will be provided in handouts. Assignments are based from, but not limited to, the textbook. 1/3 of this course will be taught in C, and the remaining 2/3 taught in C++. The course is designed to prepare students for CS344 (Operating Systems), CS446 (Computer Graphics), CS488 (Network Communications), etc.

Course Objective: The course emphasizes applying C & C++ to situations that make it difficult, but not impossible, to achieve in other languages (such as Java). Therefore, the

course will strive to point out to students the differences and similarities between C++ & Java. Students will learn the importance of design by separating specification from implementation. This separation will be accomplished through the use of headers files to specify the interface of our applications and source files (*.c & *.cpp) to implement the interface.

- To understand basic problem solving techniques.
- To understand how to construct programs modularly from small pieces called functions.
- To be able to use pointers effectively and correctly. Students will learn basic memory manipulation through use of pointers.
- To be able to process various data types coming from input streams and redirect it to the appropriate output stream format.
- To understand the notions of data abstraction and abstract data types (ADTs).
- To understand the software engineering concepts of encapsulation and data hiding.
- To understand how to redefine (overload) operators to work with new types.
- To understand how inheritance (single, multiple) promotes software reusability.
- To understand the notion of polymorphism.

This list may even be more extensive depending upon time and students comprehension of the material.

The power of C++ will be realized when students can identify the problem at hand, and successfully represent the problem through object-oriented techniques. These techniques of solving problems are built around relationships between objects. In object-oriented programming, objects share two-kinds of relationships with one another, HAS-A & IS-A. Remember this, I will talk about relationships again.

This course encourages students to become good programmers by becoming artistic programmers. I would wish everyone good luck, but you will not need it. In my experience, luck is a waste of programmer's time, it eventually runs out.

Homework and Exams: There will be approximately 10 programming assignments. These assignments are worth approximately 25% of the grade, or 100 points. That means each assignment is worth 10 points apiece. You are encouraged to do every assignment on time; it will prepare you for the tests. I may substitute some of these assignments for in-class lab exercises. There will be one midterm and a final. The midterm is worth 100 points also 25% of the grade, and the final worth 200 points constituting half of your grade. Late assignments will be docked 10% for each week they are late. No assignments will be accepted after the last week before finals week. Final exam week is December 18 – 25.

Programming Assignments:	25%, 100 points
Midterm Exam:	25%, 100 points
Final Exam:	50%, 200 points

Cheating: I encourage group collaboration and assisting others. However, any students caught cheating will be confronted by me personally and most definitely prosecuted to

the fullest extent of the rules outlined in the **Student Conduct Code**. I should not have to say anymore on this matter.

Drop/Add Deadlines:

September 25	Last day to add/drop by Dial Bear or Cyberbear
October 16	Drop/Add (No $$$ back)
December 15	Last day for drop petitions

I will try to give you a good indication of your standing in class. Students are encouraged to visit with me anytime he/she is wondering their status.

Final: 10-12 Wednesday