FD99

Course Description

Introduction to Object-Oriented programming using the C++ programming language, with examples drawn from business and other applications. Students should be acquainted with basic programming concepts, such as data types, variables, decision structures, and loop structures. Various aspects of the C++ syntax will be introduced throughout the course. Prerequisite: `CIS 150` **or equivalent**

Course Objectives

The objectives of this course include the following:

- **To continue studying the basic concepts and techniques of programming begun in** `CIS 150`
- **To study algorithms to manipulate predefined and user-defined classes**
- **To introduce the concepts and techniques of Object-Oriented programming, using business-related and other familiar applications**
- **To introduce the C++ programming language as a vehicle for implementing algorithms and for developing Object-Oriented programs**

Return to: [Top of Page]

Text Bo

Teach Yourself C++ in 21 Days, Third Editon, **by Jesse Liberty. SAM's Publishing, 1999. ISBN:** `0-672-31515-7`

Return to: [Top of Page]

Course Requirements

The following is a summary of what is expected of you for the course. Refer to the section on Grading for further details.

- **Readings and Exercises**
- **Project Assignments**
- **Class Attendance**
- **Tests**
- **Final Exam**

In this course you are expected to be more independent than usual, and majority of your efforts will be directed toward readings and projects. We will meet regularly, for lectures, demonstrations, questions and answers, exercises, and tests.

Project Assignments

Eight programming projects are scheduled for the term. Since learning the basics of programming in

C++ is a major objective of the course, these projects are considered an indispensable part of the class. Each project is assigned points (maximum = 30 pts) according to its completeness, style, and timeliness. Refer to separate paragraphs for the project grading criteria. Programming assignments are available several weeks in advance of the due dates (at midnight via e-mail).

All programming assignments which are late by 1 to 7 calendar days will incur a penalty of 30% (i.e., 21 pts max, instead of 30). Assignments which are later than 7 days will receive a maximum of 30%. If you foresee a valid reason that could cause you to hand in your assignments late--e.g., TDY, special assignment, etc.--you must get permission for special arrangements *before the programming assignment is due.*

Be conscientious in completing your assignments, since they are indispensable to learning programming fundamentals. Do not hesitate to seek help when you get stuck. When you are seeking help in debugging your programs, always accompany your questions with a hardcopy of your program listing or a copy of your algorithm written in pseudocode.

Programming assignments (PAs) should be submitted via e-mail in the following manner:

- Send source code (with appropriate identification) and executable program of an assignment as e-mail attachment to: *rmaruyam@chaminade.edu*. If you are on the local system, *rmaruyam* would be sufficient (and faster).
- Include in the *Subject* box in the e-mail message the title CIS 160 PA No. x, where x is the PA number. (CIS 160 is important.)

Tests

There will be three tests and one final exam. Refer to the Class Calendar for their dates. When you foresee that you will not be able to make these dates for legitimate reasons, make prior arrangements with the instructor. There will be no make up test for unexcused absences.

Return to: [*Top of* Page]

Grading Guidelines

The determination of the final course grades will be guided by the following distribution of course elements.

Project Assignments	**(30 x 8)**	**240 pts**
Tests	**(60 x 3)**	**180 pts**
Final Exam		**120 pts**
Total		S40 pts

The following guidelines will be used in determining the final grades.

A: = 90% B: = 80% C: = 70% D: = 60% F: < 60%

* A **minimum of 6 completed projects is a necessary condition for a passing grade.** *Return to: To of* Page]

Project Grading Criteria

Your programming projects will be evaluated according to the following criteria:

1. **Accuracy -- does the program perform as it was intended?**
   Completeness -- does the program satisfy all the requirements stated in the original problem?
2. **Programming Style -- does it follow the style guidelines recommended for this class? Refer to the Programming Style guidelines for a list *of* common points to remember.**
3. **On Schedule -- was the project submitted on time?**

Although you are encouraged to seek help from your TA and the instructor or discuss solutions with your classmates, the code you submit must be the result *of* your own work. *Chaminade University General Catalog* defines *plagiarism* **as "the offering *of* work *of* another as one's own." (p. 51) Copying the work *of* others is considered a serious breach *of* contract in this class. Please read the section titled Class Policies regarding the penalties for plagiarism.**

Return to: f Top **of** Page]

g Guidelines

In a class like this, where you are learning the fundamentals *of* programming, you must do your programming work by yourself. Generally speaking, however, software development is a group activity, where each member of a development team must be able to communicate with others at all levels. Your programs will be more readable and understandable, both t0 you **and to others (and to your grader, in this case), if the code is organized and written in a uniform manner. Although standards for programming styles and conventions can sometimes seem subjective, a certain degree *of* discipline and uniformity is necessary in any software development activity. Your source code should follow a certain style, such as documenting comments, use *of* upper and lowercases for identifiers, indentation with control structures, blank space between symbols, and blank lines between statements. The following is a highlight *of*** some ***of* the common points. Read the section in your textbook on *Programming Style Guidelines (p* 587ff) for a more complete description of style conventions we will be using in this class.**

a. **Tabs are set every 3 spaces.**
b. **Variables and function names begin in lowercase.**
c. **In general, each declaration occupies one line.**
d. **There are** spaces **after keywords and between binary operators (e.g., c = a + b; not c=a+b;).**
e. **Braces must line up.**
f. **Indent the body of a decision structure or a loop structure.**
g. **No "magic numbers" should appear in your program.**
h. **Vertical spacing--insert a blank before and after a loop structure or a decision structure. Separate major sections in the program--e.g., input section, process section, output section--with a blank line.**
   Comments should help the programmer. Avoid commenting too much or commenting too little. Major code sections in a program deserve explanations.
j. **In general, global variables should be avoided.**
k. **Every program should begin with documenting comments. See handouts for examples.**

Return to: *[Top of Page]*

Submitting One's Own Work

Each student is expected to write his or her own programs. Although modern programming practices require extensive teamwork, one of the main goals in this class is that each student learns the basic programming skills by individual practice. You must distinguish between consulting your friends and discussing problems with them from copying other people's work. The penalty for copying someone else's program or parts thereof is a grade of F for all parties involved for the first offense; and an F in the course for the second offense.

Return to: **[Top of Panel**

Attendance

Regular class attendance is important, since you are responsible for all topics covered in the class. Please make sure that you have someone from whom you can obtain notes, in case you miss a class. Generally speaking, there will be no make-up tests. When there are legitimate reasons for not meeting scheduled test or assignment dates, arrangement must be made *before* **those scheduled dates.**

Return to: [Top **of Panel**

Last updated on 8/29/99. Please send comments or questions to rmaruyam@chaminade.edu

Class Schedule

Fall Semester, 1999-2000

Wk	Date	Readings	Exercises	Assmt. Due
1	8/30	Introduction (Ch. 1) * History of C++ (5-12) Program development (12-19)	Ch. 1 Ex: 1, 2, 3	
	9/3	1* C++ IDE (Integrated Dev. Env.)		*PA 1 (Introduction)*
2	9/6	***Labor Day (no classes)***		
	9/8	Analyzing a C++ program (Ch. 2) * A simple program with cout (23-27) * Comments (28-29) * Functions (30-33) Variables and Constants (Ch. 3) * What is a variable (35-39) Creating variables (39-44)	Ch. 2 Ex: 1, 2, 3 , 4	
	9/10	(skip typedef) Short & long integers (45-47) Characters (47-49) Constants (49-51) Ennumerated constants (51)	Ch. 3 Ex: 1, 2, 3, 4	
	9/13	Expressions and Statemets (Ch * Statements and expressions (58-60) * Arithmetic operators (60-63)	Ch. 4 Ex: 1,2,3,4, 5	
	9/15	* Special operators (63-68) * Type boole (68)		
	9/17	* Relational operators (69-70) * If stattement (70-75) * Nested If statements (76-79) * Logical operators (80-83) * Conditional operator (83-84)		P.9 2 *(Arithmetic Operations)*
4	9/20	Functions (Ch. 5) * Introduction to functions (89-95) Local and global variables (95-101)	Ch. 5 Ex: 1, 2, 3, 4, 5, 6	
	9/22	* Paameters as local variables (102-104) * Return values (104-106)		
	9/24	***Test No. 1***		
5	9/27	* Overloading Functions (109-111) Basic Classes (Ch. 6)		
	9/29	* Review of OOP (7-10) * Classes new type (127-130)	Ch. 6 Ex: 1, 2, 3, 4, 5, 6, 7, 8	
	10/1	* Class members (130-137) * Implementing methods (137-140) * Constructors and descructors (140-144)		*PA 3 (Function)*
6	10/4	* Const member functions (144-145)		
	10/6	* Interface and Implementation (144-149) * Classes using other classes (152-156)		
	10/8			
7	10/11	***Discoverer's Day (no classes)***		
	10/13	**Catch up day**		
	10/15	Program Flow (Ch. 7) * While loop (161-166) (skip ***continue & break)***	Ch 7 Ex: 1, 2, 3, 4, 5	PA 4 *(Class* **Worker)**

8	10/18	* do-while loops (170-173) (skip null statement) * for-loops (173-176)		
		* Nexted loops (179-181)		
	10/20	* Scoping in for-loop (181)		
	10/22	**Test No. 2 _**		
9	10/25	Pointers (Ch. 8) * What is a pointer? (201-207)	Ch. 8 Ex: 1, 2, 3, 4	
		Reference (Ch. 9)		
	10/27	* What is a reference? (235-237) * Reference to a class (241-242)	Ch 9 Ex: 1, 2, 3	
	10/29	* Passing Functions arguments by reference (242-244) (skip making swap() with pointers)		***PA 5 (while-loop)***
		Veteran's Day (no classes)		
		Implementing swap() with references (245-247) * Returning multiple values (248-254)		
	11/5	***Test No. 3***		
11	11/8	)Advanced Functions (Ch. 10) * Overloaded member functinos (269-271)	Ch. 10 Ex: 1, 2, 3	
	11/10	}* Constructors (274-277)		
	11/12	AArrays (Ch. 11) * What is an array? (341-347)		*PA 6 (Reference)*
12	11/15	* Initializing array (347-348) * Declaring array (348-349)		
	11/17	* array of objects (349-350)		
	11/19	* char arrays (359-362)		
13	11/22	* String classes (363-369)		
				PA 7 (Array)
		Thanksgiving Holidays (11/25-11/26, no classes)		
		Files (Ch. 11) Streams (527-533) Input using *cin* (533-546) Output with cout (546-548) Formatting (548-550) * File I/0 (556-558)		
		List Class Review		**P.4 8 /Class String)**
	12/14	**Final Exam: *10:30* - 11:30**		