CIS 160-01

## 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 CIS 150
- To study algorithms to manipulate strings, dates, and other 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 Book

*Computing Concepts with C++ Essentials* by Cay Horstmann, Wiley, 1997

*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.

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

Class attendance is important, since the main points of the course will be highlighted and details will be demonstrated in the class. Generally, a lab session requires that you complete and submit a lab exercise. Readings will show you background and further explanations on the concepts and techniques covered in the class. Exercises will help you to reinforce key ideas covered in the class and to prepare you for tests. (Many of the test questions will be based on such Exercise questions). Since you learn by doing, Project Assignments and Lab Exercises will be the most important elements among your responsibilities in the course.

## Project Assignments

**Six programming projects are scheduled for the term (in addition to 11 Lab Exercises). Since learning the basics or 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.**

**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%.** *An assignment which is turned in after the start of the class on the due date is considered one day late.* **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.**

**We will try the following procedure for submitting project assignments.**

1. **Hand in a hardcopy of the source code at the start of the class on the due date.**
2. **Submit an executable file as attachment to an e-mail (to be sent to: rmaruyam@chaminde.edu). The e-mail's time stamp will be used to determine the timeliness of an assignment submission.**

*Return to:* [Top of Page]

L *s*

**Certain class times will be used as lab sessions. A lab exercise, which is designed to illustrate selected points covered in the class, should be completed and submited after each lab session. Lab exercises will be availabe well before they are due. You should come prepared for each lab so that you are familiar with the context and the objectives of each exercise. Lab exercises which are not completed within the class time must be completed outside the class and handed in by the due date.**

*Return to:* [Top of Page]

*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.**

| | | |
|---|---|---|
| **Lab Exercises** | **(20 x 11)** | **220 pts** |
| **Programming Projects** | (30 x 6) | **180 pts** |
| **Tests** | **(60 x 3)** | **180 pts** |
| **Final Exam** | | **140 pts** |
| **Total** | | **720 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 7 completed projects is a necessary condition for a passing grade.**

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

## Programming 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 to 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 587f)* 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.**
i. **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:* [To 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 Page]

### Attendance

**Regular class attendance is important, since you are responsible for all topics covered in the class. It is especially important for this course because of the density of the course content. (One class is 10% of the course.) 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.**

All students are expected to attend the lab portion of the course. Although the lab times will be designed primarily for individual work, intstructions on general interest will be presented during the lab from time to time.

*Return to:* [Top of Pagel

*Last updated on 1/11/99. Please send comments or questions to* rmaruyam@chaminade.edu

# Class Schedule

## Spring Semester, 1998-99

| Wk | Date | Notes | Text Book | Exercises | Due | Lab |
|---|---|---|---|---|---|---|
| | 1/11 | **1. Introduction** | 1-30 | R1.3, R1.12, R1.16 | | |
| | 1/13 | * Programming concepts<br>* IDE | | | | |
| | 1/15 | Program Structure | | | | Lab |
| 2 | 1/18 | *Fr. Chaminade/Martin Luther King Day (No Classes)* | | | | |
| | 1/20 | **2. Data Types**<br>* Numeric types | 37-53 | R2.1, R2.6, R2.7 | | |
| | 1/22 | * I/O statements<br>* Arithmetic Expressions | | | PA 1 | |
| | 1/25 | * Strings | 66-73 | R2.8, R2.9, 82.11, 82.13, R2.16, 82.17 | | |
| | 1/27 | **3. Objects & Classes** | | | | Lab 2 |
| | 1/29 | * OOP Concepts | | | | |
| | 2/1 | * Class string | 90-99 | R3.1, R3.3, R3.7, R3.9 | | |
| | 2/3 | * Class Time<br>* Class Employee | | | | |
| | 2/5 | | | | PA 2 | |
| 5 | 2/8 | *Test No. 1* | | | | |
| | 2/10 | **4. Decision Structures**<br>* Pseudocode<br>* If-Then-Else<br>* Relational Operators<br>* Multiple Ifs | 128-144<br>154-162 | R4.1, R4.2, R4.5, R4.6, R4.7, R4.10 | | Lab 3 |
| | 2/12 | * Nested If<br>* Logical expressions<br>* Truth Table<br>* DeMorgan's Rule<br>* Boolean Expressions | | | | |
| 6 | 2/15 | *President's Day (No classes)* | | | | |
| | 2/17 | **5. Functions**<br>* Built-in functions | 174-189 | R5.1, R5.2, R5.6 | | Lab 4 |
| | 2/19 | * User-defined functions | | | PA 3 | |
| | 2/22 | * Parameters | 189-195 | R5.8, R5.9, R5.10 | | |
| | 2/24 | * Procedures<br>*Reference Parameters | | R5.13, R5.18, R5.20, R5.21 | | Lab 5 |
| | 2/26 | * Top down Design | | | | |
| 8 | 3/1 | **6. Iteration Structures**<br>* While Loop | 232-240<br>248-252 | R6.2, R6.3, R6.6, R6.11, R6.12, R6.20 | | |
| | 3/3 | * For Loop<br>* Common Loop Patterns<br>* Trace Table | 262-270 | | | F |
| | 3/5 | *Test No. 2* | | | | |
| 9 | 3/8 | * Nested Loops<br>**Classes** | 322-335<br>338-343 | R8.2, R8.4, R8.5, R8.8, R8.12, R8.13, | | |
| | 3/10 | * Terminology<br>* Encapsulation<br>* Interface | | | | Lab 6 |
| | 3/12 | * Member Function<br>* Constructors<br>* Using Classes | | | PA 4 | |
| 10 | 3/15 | **Arrays** | Array Read in¢s | R9.1, R9.3, R9.5, R9.8, R9.12 | | |
| | 3/17 | * Declaring Arrays<br>* Array Parameters | | | | Lab 7 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 3/19 | * Algorithms Involving Arrays | | | | |
| | 3/22 | *Spring Recess (No classes)* | | | | |
| | 3/26 | | | | | |
| 12 | 3/29 | * Array of Objects<br>**Files**<br>* Introduction to Files | 428-429 | R10.1, R10.2, R10.7, R10.8 | | |
| | 3/31 | * Standard I/O<br>*File Operations | | | | *Lab 8* |
| | 4/2 | *Good Friday (No classes)* | | | | |
| 13 | 4/5 | * File Operations<br>* Examples | 452-457 | R11.1, R11.2, R11.3, | PA S | |
| | 4/7 | Test No. 3 | | | | *Lab 9* |
| | 4/9 | **Modules**<br>* Divide and Conquer<br>* Header Files | | R11.6, R11.9, R11.10,' R11.12 | | |
| 14 | 4/12 | * Implementation Module<br>* Examples | | | | |
| | 4/14 | | | | | |
| | 4/16 | | | | | |
| 15 | 4/19 | **Sorting**<br>**Searching** | **491-496** | | | |
| | [4/21 | **List Class** | | | | *Lab* 11 |
| | 4/23 | | | | | |
| 16 | 4/26 | **Recursion** | 210-213 | | | |
| | 4/28 | | | | PA 6 | |
| | 4/30 | | | | | |
| 17 | 5/4 | *Final Exam:* 10:30 -12:30 | | | | |

# Class Schedule

## Spring Semester, 1998-99

| Wk | Date | Notes | Text Book | Exercises | |
|---|---|---|---|---|---|
| 1 | 1/11 | **1. Introduction** | 1-30 | R1.3, R1.12, R1.16 | |
| | 1/13 | * Programming concepts<br>* IDE | | | |
| | 1/15 | * Program Structure | | | *Lab 1* |
| | 1/18 | *Fr. Chaminade/Martin Luther King Day (No Classes)* | | | |
| | 1/20 | 2. **Data Types**<br>* Numeric types | 37-53 | R2.1, R2.6, R2.7 | |
| | 1/22 | * 1/O statements<br>* Arithmetic Expressions | | | PA 1 |
| 3 | 1/25 | * Strings | 66-73 | R2.8, R2.9, R2.11, R2.13, R2.16, 82.17 | |
| | 1/2? | **3. Objects & Classes** | | | *Lab 2* |
| | 1/29 | * OOP Concepts | | | |
| | 2/1 | * Class string | 90-99 - | R3.1, R3.3, R3.7, R3.9 | |
| | 2/3 | * Class Time<br>* Class Employee | | | |
| | 2/5 | | | | PA 2 L |
| | 2/8 | *Test No. 1* | | | |
| | 2/10 | **4. Decision Structures**<br>* Pseudocode<br>* If-Then-Else<br>* Relational Operators<br>* Multiple Ifs | 128-144<br>154-162 | R4.1, R4.2, R4.5, R4.6, R4.7, R4.10 | *Lab 3* |
| | 2/12 | * Nested If<br>* Logical expressions<br>* Truth Table<br>* DeMorgan's Rule<br>* Boolean Expressions | | | |
| | 2/15 | *President's Day (No classes)* | | | |
| | 2/17 | **5. Functions**<br>* Built-in functions | 174-189 | R5.1, R5.2, R5.6 | *Lab 4* |
| | 2/19 | * User-defined functions | | | [&C3 |
| 72/22 | 2/22 | * Parameters<br>* Procedures | 189-195 | R5.8, R5.9, R5.10 R5.13, R5.18, 85.20, R5.21 | |
| | 2/24 | * Reference Parameters | | | Lab 5 |
| | 2/26 | * Top-down Design | | | |
| | 3/1 | **6. Iteration Structures**<br>* While Loop<br>* For Loop | 232-240<br>248-252<br>262-270 | R6.2, R63, R6.6, R6.11, R6.12, R6 .20 | |
| | 3/3 | * Common Loop Patterns<br>* Trace Table | | | |
| | 3/5 | *Test No. 2* | | | |
| | 3/8 | * Nested Loops<br>**Classes**<br>* Terminology | 322-335<br>338-343 | R8.2, R8.4, R8.5, R8.8, R8.12, R8.13, R8.14 | |
| | 3/10 | * Encapsulation<br>* Interface | | | *Lab 6* |
| | 3/12 | * Member Function<br>* Constructors<br>Using Classes | | | PA 44 |
| 10 | 3/15 | **Arrays**<br>* Declaring Arrays | Array Readings | R9.1, R9.3, R9.5, R9.8, R9.12 | |
| | 3/17 | * Array Parameters | | | *Lab 7* |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 3/19 | * Algorithms Involving Arrays | | | | |
| 11 | 3/22<br>3/26 | *Spring Recess (No classes)* | | | | |
| 12 | 3/29<br>3/31 | * Array of Objects<br>**Files**<br>* Introduction to Files<br>* Standard I/O<br>*File Operations | 428-429 | R10.1, R10.2, R10.7, R10.8 | | *Lab 8* |
| | 4/2 | *Good Friday (No classes)* | | | | |
| 13 | 4/5 | **File Operations**<br>* **Examples** | 452-457 | RI 1.1 R11.2 R11.3, | ***PA*** 5 | |
| | 4/7<br>4/9 | *Test No. 3*<br>**Modules**<br>* Divide and Conquer<br>* Header Files | | — — —<br>R11.6, R11.9, R11.10, R11.12 | *F* | *Lab9* |
| 14 | 4112<br>4/14<br>4/16 | * Implementation Module<br>* Examples | | | | *Lab 10* |
| 15 | 4/19<br>4/21<br>4/23 | **Sorting**<br>**Searching**<br>**List Class** | **491-496** | | | *Lab* 11 |
| 16 | 4/26<br>4/28<br>4/30 | **Recursion** | 210-213 | | PA 6 | |
| 17 | 5/4 | Final ***Exam:*** *10:30 -12:30* | | | | |