

ISM-4234
Object-Oriented Application Programming
Fall 2004
Dr. Soltani & Dr. Gill
Assignments

Prepared by:
T. Grandon Gill

ISM-4234: Object-Oriented Application Programming

Assignment 1: Debugging and File Functions

The goals of this assignment are to ensure that each student:

- C Can set up a multi-file project in the Visual C++ .Net environment
- C Can fix some routine compiler and linker errors
- C Can trace through program execution to determine the source of logic errors

The overriding goal here is to help each student become familiar with the tools that will help him or her identify and address the problems that will be encountered in later exercises.

Instructions for Part 1, Debugging (40 points):

This exercise involves completing the Chapter 5 exercise in the textbook, Section 5.6. Prior to beginning the debugging exercise, you should walk through the Chapter 5 demonstration. Make sure you understand what each function is supposed to do before you begin.

- 1) Create a new project (a Win32 Console application using the “empty” option) called CustomerEx1 under whatever directory you have been using projects.
- 2) Copy the source files from Exercise 1, Debugging.zip into the project directory. This folder can be downloaded from Blackboard)
- 3) Add the four source files to the project
- 4) Build the project.

You should get a long list of compiler errors, resembling the list below. Fix these errors, identifying each in a spreadsheet with columns identifying the file, line number and nature of the bug/correction. For any runtime errors you encounter, place a breakpoint at or near the error and perform a screen capture *with the program paused at the breakpoint*. Place the captures in a document, to be handed in along with the completed spreadsheet. You do not need to hand in any source code.

Instructions for Part 2, dBase File Exercises (60 points):

These exercises, from the course textbook, Section 4.5, will get you used to file I/O and the dBase file format.

Lab Exercise 4.5.3 (p.268-270): dBase Record Viewer (30 points)

Implements a simple dBase file viewer. The source files required for the exercise are on the ISM3232 disk (from the bookstore) and have been posted to Blackboard.

Lab Exercise 4.5.4 (p. 270-272): dBase Structures (30 Points)

Involves reimplementing the same dBase file viewer using structures.

You should hand in the source code for the assignment, including a screen capture of your program displaying a record on the screen.

Please hand in all assignments in an 8-1/2 x 11 envelope.

You may work in groups of up to 4 or alone.

ISM-4234: Object-Oriented Programming Using C++

Assignment 2: Video Store Classes

Assignment 2 is intended to give you proficiency in writing a C++ class, taking advantage of inheritance. Your ultimate goal is to create a *VideoStore* class, a class maintains collections of employees, customers, films, and rental transactions.

Procedure:

The exercise is performed as a series of lab exercises at the end of three chapters in the Object-Oriented Programming Using C++ textbook. These exercises are performed as follows:

I. (25 Points) Chapter 8, Section 8.6: The Company Class

Constructs a class that maintains a collection of employees, including loading, serializing, import/export, edit and display.

II. (25 Points) Chapter 9, Section 9.4: The Store Class

Constructs a class that inherits from Company, but that also maintains a collection of customers (in addition to the list of employees) and supports loading, serializing, import/export, edit and display.

III. (50 Points) Chapter 10, Section 10.5: The VideoStore Class

Constructs a class that inherits from Store, but that also maintains a collection of films and rental transactions.

You will also need to perform several walkthroughs that precede the labs in order to get the code for various unit and collection classes (e.g., employee objects).

In addition to creating the classes, you should create a test data file (see Chapter 5, Section 5.5.3, for the procedure for creating and using such a file) that can be used to test each stage. Because of the way the program is designed, you should be able to add tests to the file as you add features.

To be handed in:

- I. Printouts containing your class source code for Parts I-III.
- II. The printed listing of the .txt file that you used for testing you components.
- III. The printed output that results from running your test file against your compiled program.

These should all be placed in an 8.5x11 envelope, with the names of all group members on the outside. Failure to follow these instructions will result in a penalty!

ISM-4234: Object Oriented Application Programming

Assignment 3: Simple Database

Objectives:

Assignment 3 is intended to give you proficiency in writing a C++ classes, taking advantage of inheritance. Like Assignments 1 & 2, it will not be oriented towards Windows programming (except for the extra credit final part), and is to be done as a console application.

Overview:

The core of the assignment is writing a CDB3File class, which should inherit properties from the STL *fstream* class. This class will allow you to load, change, and display dBase III formatted files. Using this as a base class, you will then create a CDB3Sorted class that sorts and filters data from a dBase III+ formatted file. Finally, you will create a CDB3Command class, which will allow you to send commands to your file using a command interpreter.

The assignment will be constructed in a series of stages:

- C *Header class:* You will create a CHeader class that reads and writes to the first 32 bytes of a dBase file (known as the header). This header includes information such as the type of file, the number of records, and the date it was last modified.
- C *Field class:* You will create a CField class that captures the definition information for each field in the table. This information includes field name, field type, field length and precision (for numeric fields).
- C *Field collection class:* You will create a CFieldSet class, which consists of a collection of CField objects composed with a single CHeader object. This class, effectively, will contain all the definition information associated with a .dbf file.
- C *Database class:* You will create a CDB3File class (that is composed with a CFieldSet object) that allows you to manipulate a single table in a .dbf format. In a series of steps you will be adding display, editing, adding, and deleting.
- C *Sorted database class:* You will create a CDB3Sorted class, inheriting from CDB3File, that allows you to manipulate a single table in a .dbf format. In a series of steps you to sort and select records from your data file.
- C *Filter Expression Evaluator class:* You will create a DBCalc class that is capable of validating and evaluating filter expressions. This class inherits from the LogicCalc class, which is presented as a lab exercise in the book (LogicCalc, in turn, inherits from BasicCalc, which you are given and has been discussed in class). You must therefore develop LogicCalc (Chpater 17) before proceeding to work on DBCalc.
- C *Command interpreter:* As the final part of the assignment, you will be creating a command interpreter class, CDB3Command, that inherits from CDB3Sorted and takes commands (as text strings) from the user and then executes appropriate member functions on the database object.

The entire exercise is explained and specified in detail in Chapter 22 of the course textbook. You should follow these instructions in completing the project. If you plan to vary from them, you should notify the instructor and obtain permission.

Approximate point counts are as follows:

- C *Header class: 5 points.*
 - C *Field class: 5 points.*
 - C *Field collection class: 10 points.*
 - C *Database class: 30 points.*
 - C *Sorted database class: 10 points for sorting*
 - C *Filter Expression Evaluator class: 20 points for filtering.*
 - C *Command interpreter: 20 points*
-

To be turned in:

- C Hard copy of your **header.h** and **header.cpp** files.
 - C Hard copy of your **field.h** and **field.cpp** files.
 - C Hard copy of your **fieldset.h** and **fieldset.cpp** files.
 - C Hard copy of your **db3file.h** and **db3file.cpp** files.
 - C Hard copy of your **db3sorted.cpp** and **db3sorted.h** files.
 - C Hard copy of your **db3command.cpp** and **db3command.h** files.
 - C Hard copy of your **logiccalc.cpp** and **logiccalc.h** files.
 - C Hard copy of your **dbcalc.cpp** and **dbcalc.h** files.
 - C Hard copy of **db3main.cpp**, with the test code you wrote for each section appropriately identified in the comments.
 - C Printed copy of your sample output files produced by using the RUN command in your command interpreter.
 - C If you do not complete the project, you should prepare a brief list of what you did, and did not, get working.
-