



CMSHN1201 Programming Workshop

Assignment 2

Paul's Bibliographic Emporium

Coursework : Exam weighting

Weighting towards all courseworks (%)

Learning Outcomes Being Assessed

! Modular programming techniques
! Software development techniques
! Program design techniques

Set by

Dr Paul Vickers (718) p.vickers@livjm.ac.uk

Moderated by

Mr Andrew Laws (603) a.laws@livjm.ac.uk

Hand-out date

1 November, 2000

Hand-in date

1 December, 2000

Expected hand-back date

18 December, 2000

Expected feedback date(if different from hand-back date)

4 December, 2000

Estimated number of hours of student effort

24 Hours

Outline of Problem To write a C program that will:
• Allow the user to maintain a shop's master file including updating of sales data.

Introduction

A book shop, *Paul's Bibliographic Emporium* maintains a computer file of books which it keeps in stock. The file is sorted into ascending order of ISBN within ascending order of publisher code. Paul, the owner, displays the books he sells in three sections labelled Fiction, Non-Fiction, and Computing. Consequently he categorises the records in his file in the same way. Each record in the file contains the following information on one line:

- the publisher's code (3 characters),
- the ISBN (10 characters),
- the book title (30 characters),
- the author's name (25 characters),
- the book's category (F, N, or C),
- the book's price in pence (an integer).

The file is terminated by a dummy record containing '[' characters in all alpha-numeric fields and zeros in all numeric fields.

Paul has recently installed a new computer system. This meant that his files had to be converted from the old system's format into that of the new. He chose to let his assistant Melv perform the task as he claimed to have some experience of working with computers. Paul also keeps another file on his computer which holds details of all sales which he has made to-date. This file (which has also been converted by Melv for the new computer system) holds the following information on one line:

- Publisher code,
- ISBN,
- sales to-date

The file is sorted in the same manner as the master file and is terminated similarly. Paul is interested in finding out how much revenue the different publishers generate for his shop and also how popular his computing titles are. Two reports are thus required: the first will simply show total sales (and the corresponding monetary value in pounds and pence) for each publisher, with grand totals at the bottom; the second will list the sales details (number sold and sales value in pounds and pence) for each computing title Paul stocks, with totals for each publisher and grand totals at the end. In each case suitable headings are required. Appendix A shows how the two reports might look.

You may assume that:

1. if a master file record has no corresponding sales file record, then sales for that title are nil (and should be reported as such);
2. if a sales file record has no corresponding master file record (i.e., it is reporting sales for a book that is not stocked by the shop) then that record will be copied to an error file.

In addition to the report-generating features, Paul also wants to be able to add books to his stock list, amend existing book records and delete records for books he no longer stocks. Therefore, a menu-based program is required that will allow Paul to:

- Merge the sales data file with the master stock list (held in memory as a linked list);
- Produce the two sales reports;
- Display the stock list on screen;
- Add a book to the list;
- Amend a book's details;
- Delete a book from the list;

The main menu, then, should look like:

```

Paul's Bibliographic Emporium
-----

1 : Merge Sales File
2 : Create Publisher Sales Report
3 : Create Computing Books Sales Report
4 : Display book list
5 : Add a book
6 : Amend a book
7 : Delete a book
8 : Quit

Select an action [1-8] [_]
```

Program startup and shutdown

When the program is run, the master stock list (master.dat) is read into a linked list. Records are placed in the list in the order in which they appear on the file. When the Quit option is chosen, the records in the linked list will be written back to the master file and the memory used by them freed up.

Menu option 1

If this is the first time since running the program that the option is selected, then the screen will be cleared, the sales file merged with the stock list, and a confirmation message displayed. If transactions for books that are not stocked are encountered, then an error report will also be generated and the number of errors found should be displayed. Trying to run the merge again will result in an error message being displayed as it should only be run once per execution of the program.

Menu options 2 & 3

Choosing this option simply invokes the functions that generate the publisher sales report and the computing books sales report (see Appendix A). Confirmatory messages should be displayed.

Menu option 4

This option causes the stock list to be displayed on the screen four records at a time. The user should be prompted to show each screenful.

Menu option 5

This option will start a dialogue with the user prompting him to enter the ISBN, publisher code, author, title, category, and price of a new book. Checks should be made to validate the ISBN before any further details are entered. If the ISBN is valid, then another check should be made to ensure that the book isn't already held on the stock list. If everything is ok, then the remaining book details should be entered and the book added to the list.

Menu option 6

Amending a book allows the user to change any or all of a book's author, title, and price. Also, the user should have the option of not amending any fields after he has chosen a book. Before amendment can take place a check should be made to ensure that a) a valid ISBN has been entered and b) the book exists on the stock list. The user should be allowed to amend the fields as often as he likes until he chooses to quit this option.

Menu option 7

The user should be prompted to enter an ISBN. If it is valid, and it matches a book in the list, then the user should be asked to confirm the book's deletion.

Your task

Your task is to provide Paul with a program that will satisfy the above requirements. You will work in teams of two on this project. I will act as the third member of the team and will also be contributing to the solution. I have already made a start on the project and have created a project file **books.ide** that uses the following files:

- *readme.txt* Contains hints and tips
- *menu.cpp* Contains the main menu
- *computingreport.obj* A compiled version of the computing report function
- *delete.obj* A compiled version of the record delete function
- *merge.obj* A compiled version of the sales merge function
- *listfunctions.cpp* Source code for the various list management functions
- *fileops.cpp* Source code for the file management functions
- *publisherreport.cpp* Source code for the publisher report function
- *typedefs.h* Global type definitions

menu.cpp

menu.cpp contains function main, and most of the variable declarations. I have already written the outline design for the whole program and have coded option 4. You must write the code for menu options 1, 3, and 7 and I will code options 2, 5, and 6.

computingreport.obj

This file contains a working version of the computing report function. You can use this while you code the necessary code to call it. Once that's done, then you need to replace this obj file with your own version of *computingreport.cpp* (that means, design and code the function).

delete.obj

You must also design and code the function that deletes a record from the stock list. A working version is provided in *delete.obj*, but again, you will eventually need to replace that with your own .cpp source code version.

merge.obj

Another function for you to write. This is the function that merges the data from the sales transactions file with the master stock list. Again, a compiled already-working version is available in *merge.obj* and you should replace that with your own *merge.cpp*.

listfunctions.cpp

This source file contains all the functions necessary for maintaining the stock list (except for DeleteRecord).

fileops.cpp

In this file are the functions used for reading records from the master and sales files.

publishereport.cpp

Here is where I have designed and coded the function that produces the publisher sales report (see Appendix A). You might like to use this as the basis for your design for the computing report function.

typedefs.h

This contains some global type definitions that might be used by any of the program's functions.

readme.txt

This file contains various bits of information that will help you with compiling your project file.

Guidance**Input**

The program is interactive and involves lots of prompts and user data entry. A problem with data entered at the keyboard is that one must press the ENTER/RETURN key to submit the values. The problem is that that causes a special end-of-line character also to be inserted into the data-read-buffer which, if not dealt with, will screw up any following input. The following example shows how use of an extra **char** variable solves the problem. Use this technique for accepting user input in the main menu:

```
int number ;
char dummy ;
...
...
printf ("Enter a number) : ") ;
scanf ("%d%c", &year, &dummy) ;
```

This puts the end-of-line marker into the char variable *dummy* hence removing it from the keyboard buffer.

Output

To make the program more pleasant to use, we shall be using the C screen layout functions *clrscr*, *clreol*, and *gotoxy*. Examples of their use are provided in the bits of the program I have already coded.

Online resources

All the files associated with the coursework can be downloaded from the module web site as well as sample master and sales files. There you will also find a fully working version of the program, called *books_pv.exe*. Running the fully-working version will give you a feel for what the specification really means. Make sure you copy the program **and** the two data files (*master.dat* and *sales.dat*) to your own folder before attempting to execute the program.

Getting help

If you have any problems, then make full use of the staff in the laboratory session. Also, as I'm part of your team, make use of me. Obviously, I won't do the work for you, but I can often steer you in the right direction. Above all, try to enjoy it. Programming is fun! It can also be mind-numbingly frustrating at times, but hey, you didn't come to university just to let your brain atrophy.

BUYING SOLUTIONS

If you get really stuck with the design of the computing report, merge, or delete record functions, then you can "buy" an outline design from me. If you want to 'buy' a design, simply send me an email request and I will email you a skeleton .cpp file containing declarations and the outline design. Each purchase will cost you 10% of the marks available. So, if you buy all three outlines from me, then the maximum mark available will be 70%. It is important to tell me whether you're buying the outline as an individual or as a team as that will affect your marks. For example, if person A is writing the merge function, then person B might not want to lose 10% because person A bought the design for it.

It is important that if you buy a solution, you keep it to yourself. Don't give it to other teams as a) that makes you and them guilty of **collusion and plagiarism** and b) if you give it to them and we don't find out, it is **you** who loses marks, not them.

General Instructions

Resources Required

- | |
|--|
| ! Access to C compiler |
| ! Access to www.cms.livjm.ac.uk/paulvickers |

Assessment

Marks will be awarded for:

Assessment Criteria	Weight
<ul style="list-style-type: none"> • Compilation without errors • Producing the correct results with output exactly as specified. • Producing a correct design. • Structure of code matching outline design. • Use of meaningful identifiers and layout of code. • Use of correct data types • Good test plan and testing 	100%

To get an **A grade**, I will be looking for:

- ! A program that meets all the stated requirements
- ! Good use of interactive dialogue and screen layout to make the program user-friendly (see the sample program on the web site for an example of this)
- ! Clear and well-structured outline designs (where required)
- ! Clear and concise coding that matches the outline designs and which conforms to the standards used in the course text book
- ! Effective naming and labelling of identifiers (where new ones are introduced)
- ! Evidence of a thorough test plan with documentation of expected and actual results.

I will award **B grades** for submissions that meet the above criteria, but with some shortcomings or small omissions.

C grades will be given when more serious shortcomings are found (e.g., not meeting several requirements, poor design or coding) or when several of the above criteria are not met through omission.

A pass (**grade D**) can be expected when the program provides only a significant minority of the required features and/or has major flaws or omissions in the other criteria.

A **fail grade** will be awarded when a significant majority of the criteria are not met through major flaws and/or omission.

This work is to be completed in teams of **two**. When you hand in the work, you should use the attached cover sheet. On that you should put the names of both team members and you should also indicate the percentage contribution you each made. For instance, if Fred did 60% of the work and Jim did 40%, then indicate that on the sheet. You should also include a statement detailing what parts of the program were done by whom.

Appendix A

Sample Reports

Paul's Bibliographic Emporium Sales by Publisher		
Publisher	Sales to-date	Sales Value
APR	3	120.00
BLK	0	0.00
HCB	11	390.89
MAC	20	439.80
MGH	100	2000.00
MIT	20	309.80
ORB	105	933.95
Grand Totals	----- 259 -----	----- 4194.44 -----

Paul's Bibliographic Emporium Sales of Computing Titles by Publisher				
Pub.	Author	Title	Sales	Value
APR			Total for APR	0
				0.00
BLK			Total for BLK	0
				0.00
HCB			Total for HCB	0
				0.00
MAC	Culwin, Fintan	Java Foundation Classes	20	439.80
			Total for MAC	20
				439.80
MGH	King, Pardoe, & Vickers	First Course in Programming	100	2000.00
			Total for MGH	100
				2000.00
MIT	Selfridge-Field, Eleanor	Beyond MIDI	0	0.00
	Boulanger, Richard	The CSound Book	5	174.95
			Total for MIT	5
				174.95
ORB			Total for ORB	0
				0.00
			-----	-----
		Grand Totals	125	2614.75
			-----	-----

CMSHN1201 Programming Workshop Coursework Cover Sheet

Complete and attach this form to your coursework submission.

THIS IS NOT A RECEIPT.

	Name	Registration number	%Contribution
Team member 1			
Team member 2			

This work is for the attention of

Dr Paul Vickers, CMS

Checklist:

The grade we realistically think this piece of work is worth is:

This work is our own; we have properly attributed any contributions by others where they occur and we understand the regulations surrounding collusion, plagiarism and cheating

We have included the following items (tick boxes when complete):

Virus-free diskette containing C source code:

Listing on collated stationery of your source program

Test plan and expected and actual results:

Summary statement of who was responsible for what:

Signed

Date