CPE 100 Sections International Sections
August 2020 Term
Programming Project 2: Simple Database Program

Last update: 20 October 2020

1. Assignment Objective

The objective of this assignment is to give students a chance to design, write and test a fairly large program that uses files, structures, and other C programming concepts introduced during the second half of the term.

This project will also give students an opportunity to create a program that has multiple source files. Your program must include **at least three C source files**. Suggestions for how to break up the program are provided below.

2. Assignment Description and General Requirements

A ***database*** is a way to store information about some kind of useful real-world objects, over a long period of time. Each type of object will have a set of properties or characteristics. The database stores the values for these properties. Usually there will be programs used with the database that allow a user to view or change the properties for a particular object.

Databases are normally implemented using one or more ***files*** so that the information they contain will be ***persistent*** (will not be lost when the computer is shut down or the database program is not running). "Relational" databases such as MySQL or Oracle use very complicated sets of files. This project will be much simpler and will involve what is called a "flat file database".

For this assignment, you will design and implement a database to store a particular kind of object, and a program to access and modify this database. Different problems will be assigned to each student. However, the general requirements for all the programs will be the same.

The database program ***must do all of the following***:

- Allow the user to **display** the current contents of the database (all objects) on the screen.
- Allow the user to **add** a new object (a new "**record**").
- Allow the user to **search for** an object that is in the database. The specific problem description explains what properties can be used for searching. In some cases, a search might return multiple objects. If multiple objects are returned by a search, the program should allow the user to choose one for further actions.
- Once the user has found and selected an object, allow the user to **modify** its properties or simply display the full set of current properties for the object.

The database program should operate in a loop, repeatedly asking the user which of the above operations he or she wants to do, until the user says that he or she is done. When the database program exits, it should **save the current contents of the database to a file**. (Or, if you prefer, the program can write a new version of the database file after each change to an object.)

Note that your database must be *persistent*. That means it must be possible to run your program a second time in order to view and modify information that was stored during a previous run. However, your program should create a new database if one does not already exist from a previous run.

Your database program **must be able to handle an unlimited number of records**. Theoretically, the user can continue to add information forever.

The program must **check all information** supplied by the user to make sure that it is valid for the particular property (e.g. valid phone number, valid date, etc.). Details of format and validation requirements are provided in each problem description. Read and follow these instructions carefully.

You can and should use the code you wrote for Project 1 as a base for the validation. You may also use my code from inputFunctions.c *(demos for Lecture 7) or* ioFunctions.c *(demos for Lecture 10) as long as you do not change* ***it.***

3. The structure of your program

The database program that you will write must be *modular*. It must divide up the functions into several different source files (at least three), each of which handles a different aspect of the program functionality. It is up to you to decide how to divide up the functionality. Some possibilities include:

- User interface, database reading and writing, data validation functions, data manipulation functions (add, modify),
  -- or --
- Record adding, record searching, record modification, record display, database reading/writing.

These are just some suggestions.

Your program *should not repeat the same or similar functions in multiple* ***.c*** *files*. For example, you should have only one function that reads the database, in a single **.c** file. Other functions, possibly in different **.c** files, should call this function when they need to read the database.

4. What You Need to Submit

You should upload two files.

1. A single *.zip, .tar* or a *.tgz* file that holds everything needed to build and run your project.

   a) The **makefile.**
   b) The **source code as .c and .h files**. All files should be in the same directory. It should be possible for me to go to that directory and simply type **make** to compile the project.
   c) A sample data base and any other necessary files.

You should name this file ***nnnnnnnnnnn_ProblemN.zip*** (or .tar, or .tgz), where ***nnnnnnnnnnn*** is your full student ID and ***N*** is the number of your assigned problem (1-10).

Do not submit .rar archives.

2. A *PDF file* with the following contents:

   a) A **title page** with your name, student ID, and a title which is the name problem you were assigned.
   b) ONE **flowchart** showing the overall program logic. This flowchart should not show the details within each functional branch of the program (e.g. adding, searching) but should make it clear where and when you are reading and writing the data base file(s).
   c) A **table** explaining how you broke your program into modules. This table should have ***one row for each .c module***. It should have the following columns: *Name of Module*, *General Description* (summary of the purpose of the module), *Functions* (list of all the functions in the module). The names of your functions should make it clear what each function does! If this is not obvious – you should rename your functions!
   d) An explanation of the **structure of your database file,** including a diagram or example.

The PDF file should be named ***nnnnnnnnnnn_ProblemN.pdf*** following the same structure as the code file.

The PDF file and code archive file are due at **noon on Friday December 18th**. Late submissions will receive a zero.

You will not submit a printed booklet for this project.

5. Further Instructions

About Submissions:

- All code must follow the coding standards for the course. This includes standards for naming, indenting, and

comments. You should review the standards before you begin writing your code.
- Try to keep your lines of code to 80 characters or shorter. You should break up longer lines with carriage returns.

About Your Solution

There are many possible ways to write this kind of program. You can use any format for your database files that you want. The files can be text or binary. You can decide when to write data to your database, either when the program exits or any time there is a change. You can store data in arrays while you are working with it, or use the file itself as the main storage method. I am happy to discuss the possibilities with you.

Think about making your solution easy to use. For example, if the user wants to modify a record, don't make her retype all the information, just the properties she wants to change. If there's an error, try to explain in your message what the problem is. Don't display so much information at once that the user has to do a lot of scrolling to see it.

If you have questions, please email me and ask! I'll answer you (if you are the first person to ask) and if the question seems like something other people might want to know, I'll also answer on the Facebook page.

6. Grading

This project counts as 15% of your course grade.

Grades will be based on the following criteria:
- 6 points based on the correctness and completeness of the design (breakdown into modules, logic and data structures)
- 6 points on correctness, completeness and structure of the code and the testing.
- 3 points on programming style (readability, variable names, comments, indenting, etc.)

I have tried to assign problems so that you can use at least one of the validation functions you wrote for Project 1.

For the other validations, **do not copy code** from other students. If I find any suspicious code in your project, you will get a zero on the project, plus lose 10 points off your course grade. That is, you will lose a total of 25% of the possible points in the course.

Do not risk this! Do your own work, 100% ! Furthermore, **do not share your code with your friends**, even if you want to “help”. If I can identify the source of copied code, both the copier and the person who is the source will be punished for cheating!

7. Detailed Specifications for Each Problem

I will assign problems to each student. You must implement the problem assigned to you.

All properties of the object are required for every object in the database, unless the instructions say “can be blank”.

If a property is indicated as “must be unique”, this means that you must check the entered value against all other records in the database. If the new record property or modified property is the same as any existing record, you should give an error message and ask again.

The instructions also indicate what are the possible properties for searching. A string entered by the user to specify a search is called a “search key”. If a property says “partial match possible”, this means you must allow searching based on the first few characters of a property. For instance, bank account number in Problem 1 allows partial matches. This means you can search for a bank account number using “209” and you should get all records with account numbers that begin with “209”. Then you should let the user choose which one she wants to view or modify. On the other hand, if you enter a full bank account number as the search key, you should get at most one match.

Note: For bank accounts, phone numbers and other properties with optional punctuation (like dashes), I recommend that you store the numbers without the optional punctuation so all data will be consistent. When you start a search, you should remove the punctuation from the search key as well.

Problem 1: Banking System Customer Database

This database will manage a set of **bank customers**. Each customer will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Account number	Project 1 (11) – must be unique	209-18772-022 12845189905	Yes
Date/time opened	Project 1 (3) – must not be future	2017-12-09 14:22	Yes
Customer name	Project 1 (8)	Mrs. Janet Jones	Yes
Nationality	Two letter code must match one of list in: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements	TH US	
Type of account	One of: “CURRENT”,”SAVINGS”, “FIXED_DEPOSIT”		
Initial deposit	Positive integer	20000	

Must allow search by:

- Account number - partial match possible
- Nationality

Problem 2: Address Book Database

This database will manage a set of **contacts**. Each contact will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Name	First and/or last – alphabetic only	Jack Jenny Jones	
Birthday	Project 1 (2) – must be in the past	21-01-1987	Yes
Email address	Project 1 (5) – must be unique	seg.goldin@gmail.com	Yes
Phone (international)	Project 1 (7)	+1-413-3672009	Yes
Address	Allow alphabetic, numeric, spaces and / only. May have any number of parts.	21/5 Pracha Uthit Road	
Gender	M, F or O		

Must allow search by:

- Name – partial match possible
- Email – partial match possible

Problem 3: Thai Hotel Database

This database will manage a set of **hotels**. Each hotel will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Hotel Name	Any string with alphabetic characters, digits and/or spaces	The Meridian Suk15 Hotel	
Date Built	Project 1 (4) – must be in the past	21 JAN 1978	Yes
Hotel services code	Project 1 (10)	TWP$$	Yes
Thai mobile phone	Project 1 (6)	0835621111	Yes
Province code	"TH-nn" or "TH-S" Must match one of list here: https://en.wikipedia.org/wiki/ISO_3166-2:TH	TH-50 (Chiang Mai) TH-S (Pattaya)	
Star rating	Integer between 1 and 5	3	

Must allow search by:

- Hotel Name - partial match possible
- Province code

Problem 4: Movie Database

This database will manage a set of **movies**. Each movie will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Movie Name	Any string with alphabetic characters, digits and/or spaces	Toy Story 4 Fantastic Beasts and Where to Find Them	
Release date	Project 1 (2) – must be in the past	03-07-2009	Yes
Movie site URL	Project 1 (15)	http://toy-story-movie4.com	Yes
Category code	Similar to hotel code (Project 1 – 10) – any combination of D (drama), R (romance), A (action), C (comedy), H (horror)	D ACH	Yes
Gross revenue in millions of baht	Positive integer	130 (means 130 million baht)	
Average user rating	Floating point between 0 and 10	7.4	

Must allow search by:

- Movie Name - partial match possible
- User rating – should return all movies **with ratings greater than or equal to the search string**

Problem 5: Immigration Database

This database will manage a set of **people entering Thailand**. Each person will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Passport number	Project 1 (9) – must be unique	US-321211111	Yes
Entry date	Project 1 (1) – must be in the past	14/03/2554	Yes
Name	Any combination of letters, digits and spaces	Harry Michael March Wiranto	
Address in Bangkok	Project 1 (13)	120/3 Thonglor 10200	Yes
First visit?	One of Y, N, or U (unknown)		
Visa length in days	Integer value 10 to 365	30	

Must allow search by:

- Passport number - partial match possible
- Entry date

Problem 6: Hospital Database

This database will manage a set of **patients**. Each customer will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Passport number	Project 1 (9) – must be unique	US-321211111	Yes
Name	Any combination of letters, digits and spaces	Harry Michael March Wiranto	
Date of Birth	Project 1 (1) – must be in the past	09/10/2549	Yes
Today's date	dd/mm/yyyy Buddhist era (same format as D.O.B.)	[You can and should create this ***automatically*** – do not ask!]	
International phone	Project 1 (7)	+86-89-89998899	Yes
Gender	M, F or O		

Must allow search by:

- Passport number - partial match possible
- Name – partial match possible

Problem 7: E-commerce Customer Database

This database will manage a set of **customers**. Each customer will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Name	Any combination of letters, digits and spaces	Harry Michael March Wiranto	
Email address	Project 1 (5) – must be unique	seg.goldin@gmail.com	Yes
Password	Project 1 (17)	my$Pass13%	Yes
Last order date	Project 1 (4)	09 MAR 2017	Yes
Country of residence	Two letter code must match one of list in: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements	TH US	
Address	Any combination of letters, digits, spaces and periods	P.O. Box 200 Bangkok 10111	

Must allow search by:

- Email address – partial match possible
- Name – partial match possible

Problem 8: CPE Computer Database

This database will manage information about **computers owned by CPE**. Each computer will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Date purchased	Project 1 (2) – must be in the past	18-10-2000	Yes
IP Address	Project 1 (14) – must be unique	202.44.12.67	Yes
Owner name	Any string of letters, digits and spaces	Ajarn Sally Tao	
Owner mobile phone	Project 1 (6)	0534210999	Yes
Room number	Four digit string starting with either “10” or “11”	1044 1121	
Type of computer	One of “DESKTOP”, “LAPTOP”, “SERVER” or “TABLET”		

Must allow search by:

- IP Address
- Owner name – partial match possible

Problem 9: Ghostbusters Database

This database will manage information about **supernatural events**. Each event will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Date and time of supernatural event	Project 1 (3) – must be in the past	2000-10-18 12:34	Yes
Name and title of person reporting event	Project 1 (8)	Mr. Harry Jones Ajarn Sally Goldin	Yes
Thai mobile number of person reporting	Project 1 (6)	0534210999	Yes
Event code	String including 'G' (ghost), 'V' (vampire), 'D' (demon), 'Z' (zombie) and/or 'W' (werewolf), in any order. Each letter can appear only once in the string.	G WVD ZG	
Number of people killed	Positive integer, can be 0	3	
Results of investigation	One of "SUCCESS", "FAILURE" or "UNKNOWN"		

Must allow search by:

- Event code – search key should be one character (e.g. 'G'), return all records with event code that includes this character
- Reporting name – partial match possible

Problem 10: CPE Student Database

This database will manage a set of **CPE students**. Each student will have the properties below.

Property	**Format/validation rules**	**Examples**	**Project 1?**
Student ID	Project 1 (12) – must be unique	61070501123	Yes
Student name with title	Project 1 (8) - modified to only allow titles Mr., Mrs., Miss, Ms., not Dr. or Ajarn	Mr. Harry Jones Miss Natucha Omsakul	Yes
Student date of birth	Project 1 (1) – must be in the past	05/09/2550	Yes
Student home province	"TH-nn" or "TH-S" Must match one of list here: https://en.wikipedia.org/wiki/ISO_3166-2:TH Or can be "IN" for international students	TH-50 (Chiang Mai) TH-S (Pattaya)	
Latest GPA	Floating point number 0 to 4.0	3.56	
Student gender	M, F, or O		

Must allow search by:

- Student ID – partial match possible
- Home province