

ASSIGNMENT # 1

In this assignment you will create a miniature phonebook database according to the following guidelines.

The purpose of this project is to review topics you should be familiar with from CIS-15; advanced array and string handling, pointers, *struct* types, dynamic memory allocation, function parameter passing, the preprocessing phase, macros, header files, separate and conditional compilation, and file input/output. Without a thorough understanding of these topics you will encounter many difficulties with the transition into C++ and with technical details (code) that are necessary to implement the concepts we will cover in class.

The following narrative is organized into several sections:

- General Overview – describes the setting of the problem in general.
- Program Specifications – describes the technical details of the program required.
- General Notes - some miscellaneous items about the program.
- Data File – initial data required for the program and what to submit.
- Program structure – a diagram representing file and function relations.

General Overview

The basic flow of this project consists of an initial setup phase, the processing of database operations and a program-terminating phase.

Setup Phase

The program loads existing phonebook records from a data file (Pbook.dat) into memory. Initially the data file will contain four records, but upon the second execution of the program and onwards, the number of records will be unknown. Before loading the records into memory the program should dynamically allocate space for these records, which are aggregated in an array of *struct* types, i.e. the phone book is stored in an array of phonebook records. During this phase the program should also sort the whole database, set a record counter and keep track of the number of records read from the data file.

Each record consists of three fields in the following format: *Last name, First name, Phone number.*

e.g. Taranova Vincent 212-6401781

ASSIGNMENT # 1

Phonebook Operations and Menu

The program will repeatedly present the user with the following menu of options:

- 1- Add Record**
- 2- Update record**
- 3- Search for Record**
- 4- Print Phonebook**
- 5- Terminate**

Enter your Choice ->

The user will choose one of the options and the program will process his request, executing the relevant operation. Upon completion of the required operation the screen should be cleared and the user should be returned to the main menu.

Adding records

When the first option (1) is chosen, the program presents the user with the following messages:

Enter Last Name ->

Enter First Name ->

Enter Phone Number ->

The user enters the relevant information and the program adds the record to the end of the database, i.e. the first available space in the array of phonebook records. Finding the end of the database is done by maintaining the record counter initialized during the setup phase.

After the record is added the record counter should be incremented and the whole database (array) should be sorted. Sorting the database is necessary to ensure a correct execution of binary search used by the 'Search Record' and Update Record' options.

Updating records

When this option is chosen (2), the program presents the user with the following message:

Enter Last Name ->

The user enters the relevant information and the program searches, using binary search for the record according to the last name. Once the location (in the array) of the record is found the program displays the whole record (last name, first name and phone number) and prompts the user to enter the new data.

ASSIGNMENT # 1

Enter First Name ->**Enter Phone Number ->**

After the user enters the new data the program updates the database (array) and displays the updated information using the following messages:

Updated record**Last Name:****First Name:****Phone Number:****Searching for records**

When this option is chosen (3), the program presents the user with the following message:

Enter Last Name ->

The user enters the relevant information and the program searches, using binary search for the record according to the last name. If the record exists in the database the program will display its information, otherwise it will display an appropriate message such as '**record not found**'.

Printing the phone book

If this option is chosen (4) the program will output either to the screen or to a file (up to you) the whole database of records in an appropriate tabular form, e.g.

| Record Number | Last Name | First Name | Phone Number |
|---------------|-----------|------------|--------------|
| 1 | Taranova | Vincent | 212-6401781 |
| 2 | | | |

If you choose to output to a file, the file should be called 'list.dat'.

Terminating Phase

When the user chooses the last option (5) the program should store the whole database in the data file ('Pbook.dat'), and then terminate. This is to ensure that modifications to the database during any session are maintained throughout the following sessions.

Note that the interaction (read/write) with the data file is performed only during the setup and termination phases. That is, once the database is loaded from the data file into memory, all work (adding records, updating, etc) is done inside memory, and only upon termination the data is saved back to the file.

ASSIGNMENT # 1

Program Specifications**Data structures**

You will need to create a new data type called **Pb_Rec** that will characterize a phonebook record. Your data type should contain three fields corresponding to **last** name, **first** name and **phone number**, which are all strings.

As mentioned earlier, the phonebook is loaded and processed in memory. For this you will need to create an array of **Pb_Rec** types. This is done during the initialization phase that is discussed next.

Initialization

In the main function you will declare an uninitialized pointer that will later point to the array of phonebook records. This pointer is passed (pointer to pointer, why?) to an initialization function. The function will first dynamically allocate space for an array of **five** Pb_Rec type elements. It will then start reading records from the data file and store them one at a time in the array. If more space is required, the function should **reallocate** the space used by the array and reserve space for **five** more elements. This procedure is repeated until all records are read from the data file. Using this method a very small amount of memory is wasted on unused space in the array.

After the database is loaded into the array, a sorting function should be called. The sorting function should sort the phonebook (array) by last name. The initialization function should also keep track of the number of records read, and upon termination return this value to the main function.

The main function and the menu

The main function should include two essential variables: a pointer to the array of records and the record counter. These variables are passed to every other function. The initialization function and the adding records function will modify these variables, therefore their address should be passed; a pointer to the pointer of the array and a pointer to the record counter.

The first function that is called from the main function is obviously the initialization function, once control is returned from this function, the main function will repeatedly present the user with the menu and process his choices until the user chooses to terminate the program. Each time the menu is displayed the screen should be cleared.

The menu should be a function that displays the menu and returns the user's choice to the main function that will process the choice using the *switch* command.

ASSIGNMENT # 1

Searching and Sorting

These should be two separate functions that are not embedded into any other functions. Search should be performed using *Binary Search* (utilizing recursion) and not linear search. Sorting can be performed using any method familiar to you, *Quick Sort* recommended.

Adding records

This function will present the user with the messages mentioned earlier. After the user enters the required information (last name, first name and phone number) the function should first check if the array contains room for the new record. If so, the new record should be appended to the end of the array, that is, the first available empty space. E.g. if the array contains 7 records the new record should be added to the 8th position in the array. Finding the correct position is done by maintaining the record counter. If the array does not contain room for the new record the function should **reallocate** the memory used by the array and reserve space for **five** more elements, utilizing the same procedure used by the initialization function. After more space is allocated; the new record is appended to the end of the array. E.g. if the array contains space for 10 records and all 10 spaces are already occupied, the function will reallocate the array and add 5 more elements. Now the array contains space for 15 records and the new record will be added to the 11th position, leaving 4 elements unoccupied.

Once again, after the new element is appended, the function should call the sorting function that will sort the phonebook (array) by last name. Also, do not forget to increment the record counter after the new record is added.

Updating and searching records

These will be two separate functions that will call the *Binary Search* function after the appropriate messages are displayed and data is entered. In no case will the binary search function be embedded into these functions. These functions should also be able to cope with a scenario in which a record is not found. You will need to adjust the binary search function to accommodate such a scenario.

Modular programming

Your program should be broken into three files, implementing ‘modular programming’ and utilizing the ‘*separate* and *conditional compilation*’ techniques. The first file referred to as the ‘*driver*’ will contain only the **main** function. The second file referred to as the ‘*implementation*’ will contain all the **functions** used by the program. The third file is the ‘*header*’ file that contains all your global definitions such the Pb_Rec data type.

ASSIGNMENT # 1

General Notes

- Use pointer notation only – all array handling operations should be done using pointers and not indices.
- Executing the program – you should execute the program at least twice adding at least 5 records each time, i.e. after the first run you should have at least 9 records in you database and after the second run, at least 14 records. This is to ensure that your memory allocation routines work correctly.
- What to submit – all source code and data files. In addition you will need to submit a list of all records in the phonebook after each run, illustrating that all new and modified records were saved back to the data file (Pbook.dat). You can do that by printing the data file. You will also need to submit a sample of output illustrating that all phonebook operations (menu options) are working correctly.
- Your program should be commented extensively, uncommented programs will not be graded!

Data File (initial data)

| | | |
|----------|---------|-------------|
| Taranova | Vincent | 212-6401781 |
| Williams | Serena | 718-9515657 |
| Turing | Alen | 914-2256901 |
| Marly | bob | 917-3351289 |

(format: one record per line)

ASSIGNMENT # 1

Program Structure

