Interpreter data structure Project Report

An analysis of project design and implementation

Project team :

* Omar Samir
* Yehia Salah

Lecturer: Dr. Saleh Elshehaby

**Table of Contents**

1. **Abstract**
2. Implementation steps
3. **Teamwork methodology**
4. **Program details**
5. Overview
6. Main blocks
7. Workflow
8. **Functionality**
9. File load
10. BST
11. Evaluator
12. Heaps
13. Stacks
14. Console interface (main)
15. **Features**
16. The validation test
17. Containerization
18. Error handling
19. Code annotation
20. **Results**

1. **References**

**Abstract:**

The Interpreter implementation is based on the use of functions (caller/callee concept-modules), data structure (BST, Stacks) and headers. Our main objectives were:

* Containerization of the code thus guarantee the reusability of the modules.
* Error handling and cover all the possible cases
* Reaching the maximum optimization and performance possible.

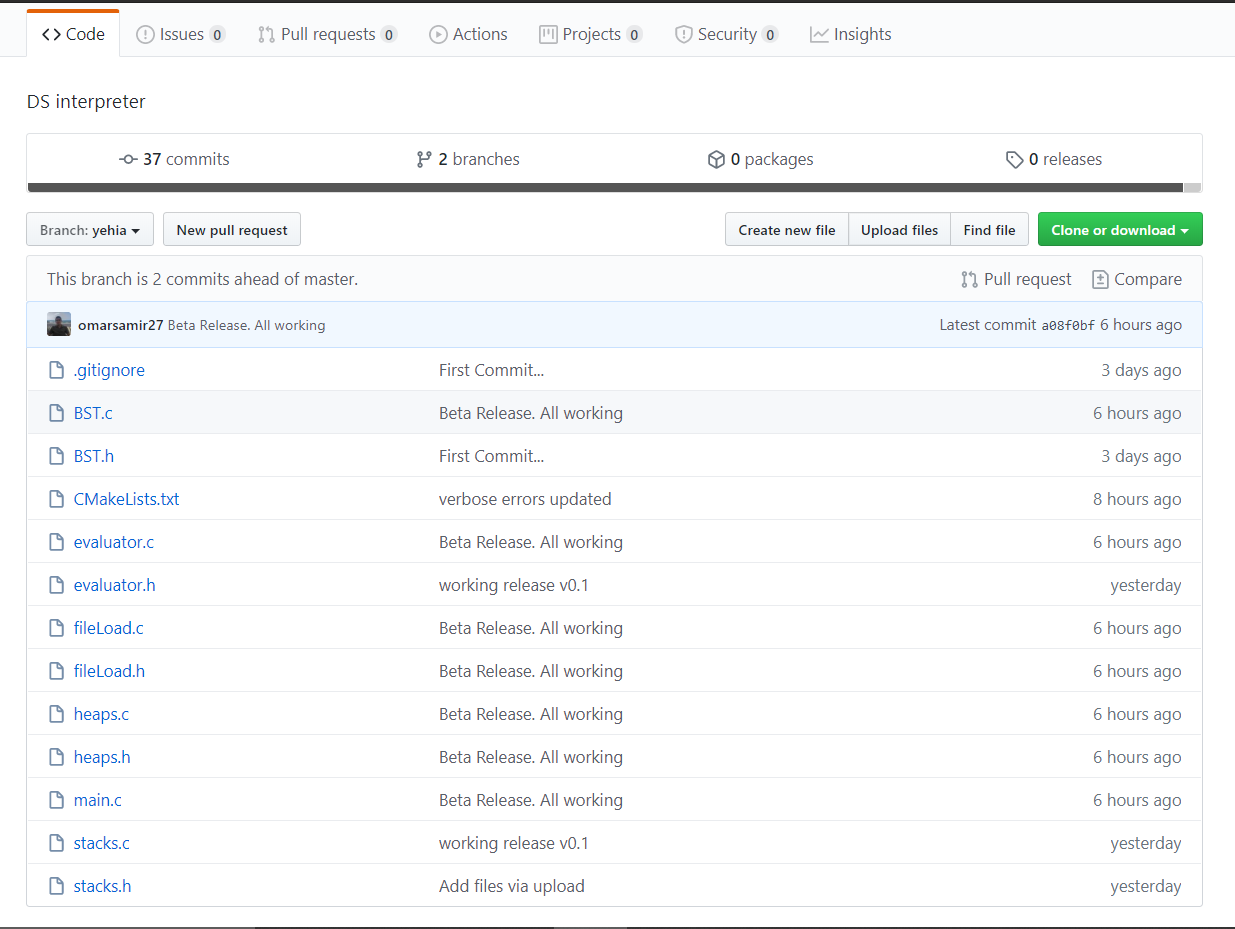
Implementation steps:

* Write pseudo code of the main modules and i/o charts.
* Split the project into main Files (sources + headers).
* The modules implementation of each file.
* Enhancing time complexity (reduced from O(n^2) to O(n\*logn )).
* Exceptions handling and covering all possible scenarios.
* Code annotation, improving program features and make it user friendly.

**Teamwork methodology:**

"Adding manpower to a late software project makes it later"

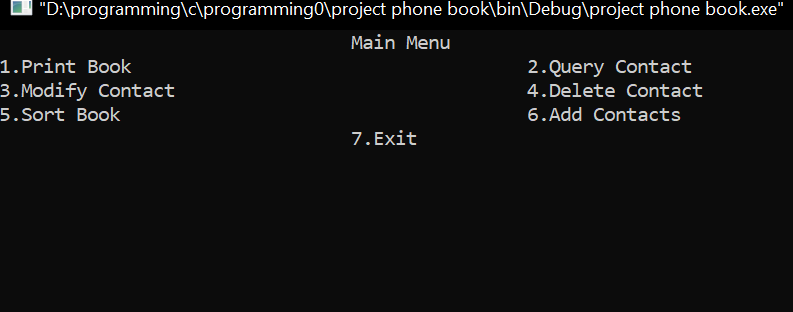
* Split code files on members (every member works individually on assigned file).
* Set the communications between the code containers and preset the functions prototypes.
* Combining all project files.
* Use of version control platform (Github) to deliver the code.

****

**Program details:**

1. ***Overview:***

The programs id divided into six main files:

****\*picture console

1. **Console interface (main)**

Contains the main of the program and the console interface code as pic. above.

1. **File load**

Load the data from the file and return a BST and heap to main.

1. **Evaluator**

Rebuild the RHS string and process an omni-check and evaluate it unless corrupted.

1. **Stacks**

Convert RHS from infix to postfix and evaluate it.

1. **BST**

Store the variable name and its key using bst, sorted alphabetically

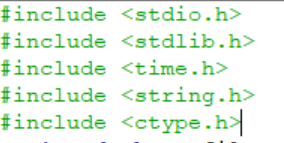
1. **Heaps**

Convert the bst into array and heapsort it.

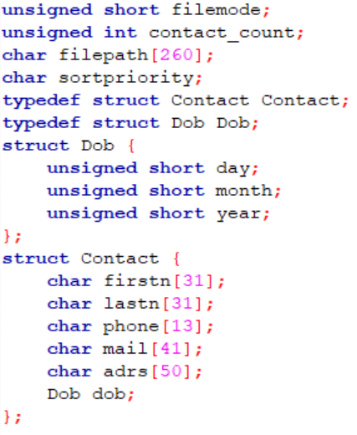
1. ***Main blocks:***

Program is split into 4 main parts:

***1.Headers***

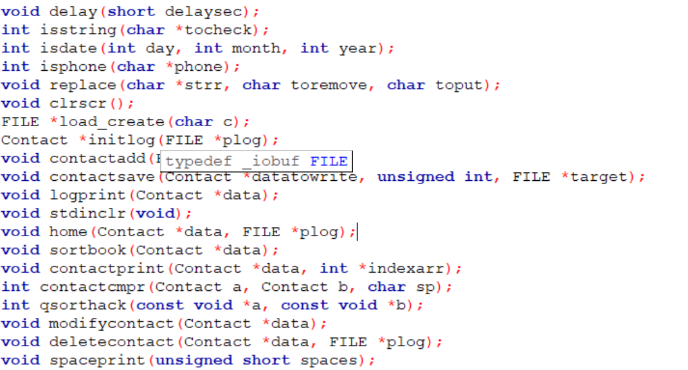
* The shown headers were used to ease the implementation of the program and preventing reimplementing non necessary functions.

***2.Global variables***

* Global variables have global scope so the can be “seen” from all functions which ease many processes in the program.

-Every variable will be discussed in the next sections

***3.Functions prototypes***

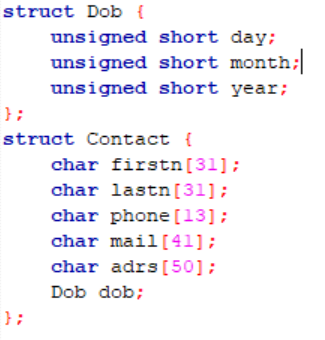
* A function prototype is simply the declaration of a function that specifies function's name, parameters and return type.(used to so that the Compiler check the parameters type and the names of functions ,and ease the reading of the program for the developer.)

***4. The main function and the implementation of other functions***

1. ***Managing data:***

***Structures:***

For the life of the program, all the data are stored(temporarily) in Structures, as follows:

* The Dob (date of birth) struct (standard size 6 bytes)

Unsigned short is used due to its range.

* The contact structs (standard size 172 bytes)

All fields’ lengths are the internationally average length (+null char).

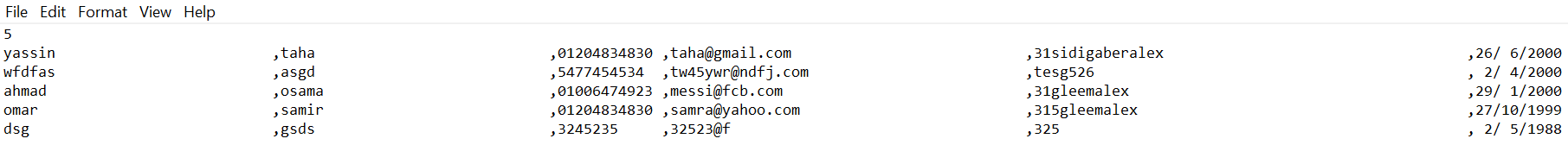
Chars are used to facilitate the handling of input of the user (using the string.h, ctype libraries)

\*\* Dob is an independent struct to facilitate the sorting process.

***Files:***

File (.txt) is used as data storage.

The Text file is preformatted to be random accessed by using the first line in the file as records’ counter and standardizing line length to be 180-byte.



Further details will be discussed in functionality section

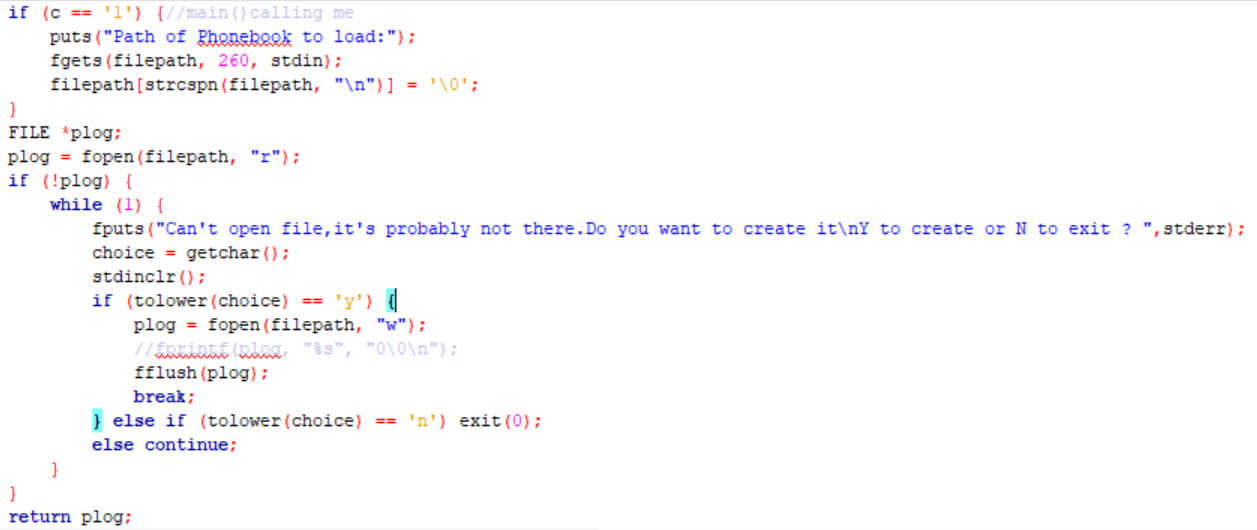
**Functionality:**

\*\* In-depth view of the code\*\*

1. ***Load\_create:***



It’s a function from type file, it’s objective is to prepare the required file and returns its pointer to the caller.



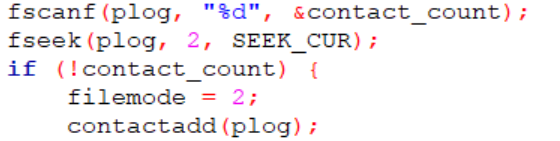
* This function is called by main or home.
* If it’s the first log to user, this function take the path file from the user (the main calls it with argument 1).
* In this module, the required file is open as read. (if the file path exists).
* In case the filepath isn’t available, the program asks the user if he want to create a new file or exist.
* Finally (if not exist) the function returns a file pointer to the caller.

1. ***Intitlog:***



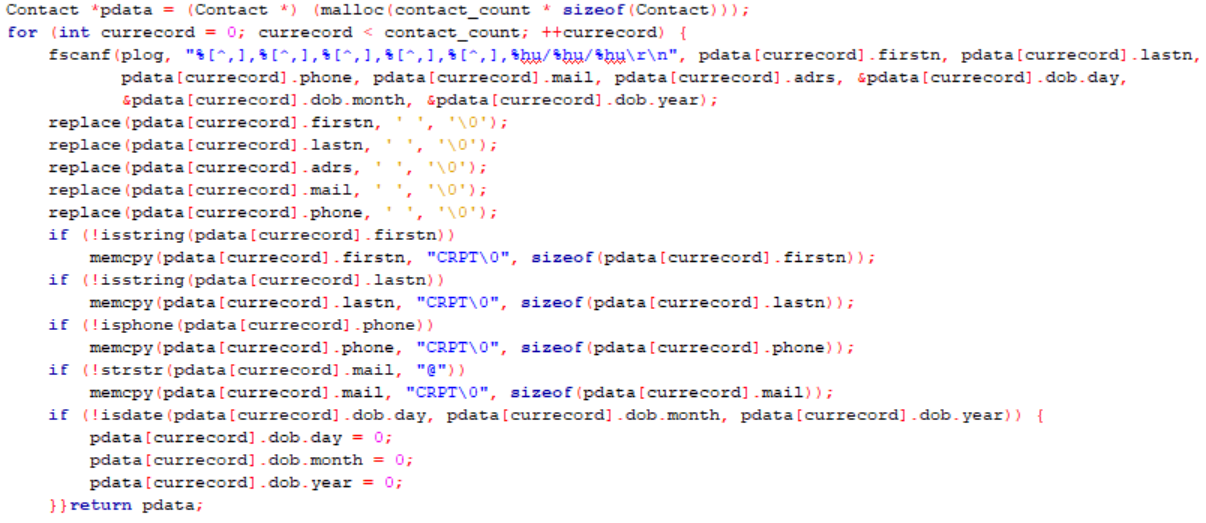
It’s a contact(struct) type function, it takes the file pointer as argument and returns a pointer to the struct contact (the filled/loaded struct).

**This function is divided into two blocks: -**



* The first part ensure that the contacts file has a header: the number of records in the file (view managing data section).
* If the header exists, then function fseek move the file pointer to the first record
* If the header isn’t available, the function calls contactadd(to format the file as required) and returns null.

**The second block:**

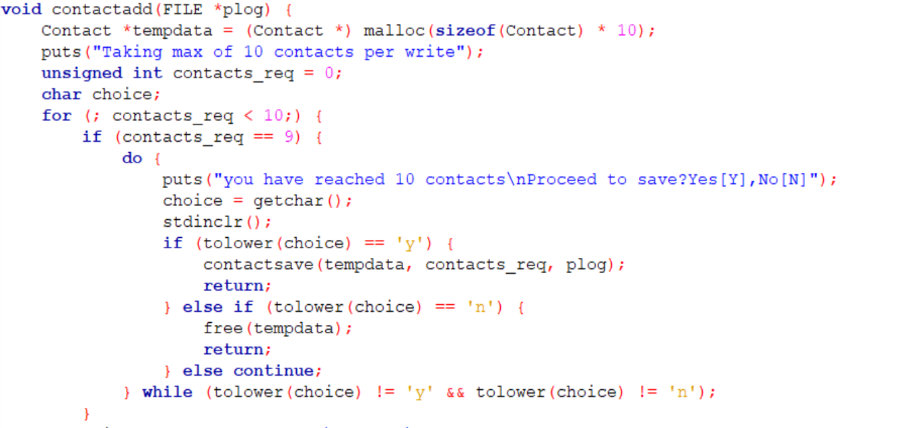


* This block reserve a memory in heap (size equal to size of contact struct\* number of contact).
* A for loop is used to loop on all the records in the file to scan them
* A function fscanf is used (with comma delimited argument) and scan every field and puts it in the required place in contact struct(pdata).
* Function replace is used (see features) to remove extra spaces from string (due to storage issue/standardized length of record).
* Function isstring/strstr/isdate check the validity of the fields in the record if not valid CORRUPT is put in this field so the user knows that there is a problem.

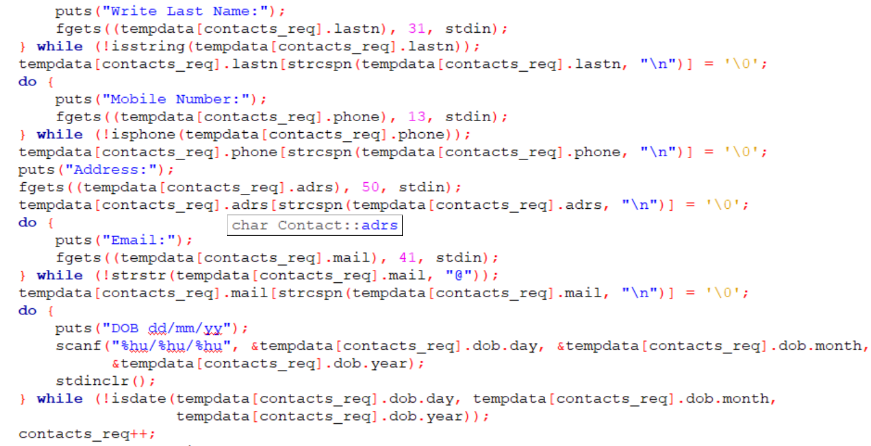
1. ***Contact add:***



Contact add is a void function that takes the pointer to the file as a parameter, it’s divided to 2 main blocks of codes. It saves new contacts in the heap (here as a buffer) and then calls savecontact function to save them to the file.

**1st block of code:**

* The function allocates the size of 10 records in heap.
* If the user reaches/adds 10 contacts, a prompt will be shown (save or discard the changes)
* If save, then the contactsave function is called
* If discard, then the allocated memory in heap is freed.

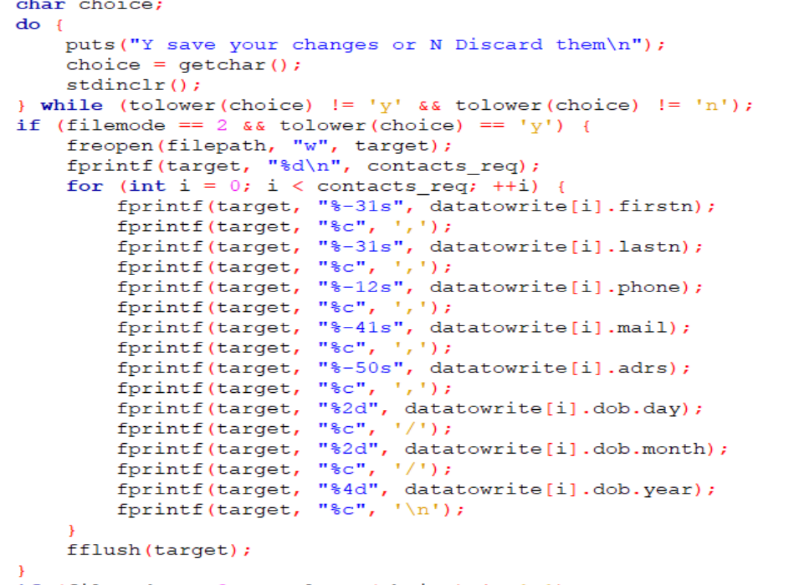
**2nd block of code:**

* This block of codes takes all the data from the user (name/address/phone/…)
* If the user enters the data in the wrong format, he must resubmit it (due to the do/while loops and the validation function)
* Contact\_records is incremented after every contact saved to the buffer(heap)
* Finally, the function calls savecontact to save the contacts into the file.

1. ***Contactsave:***

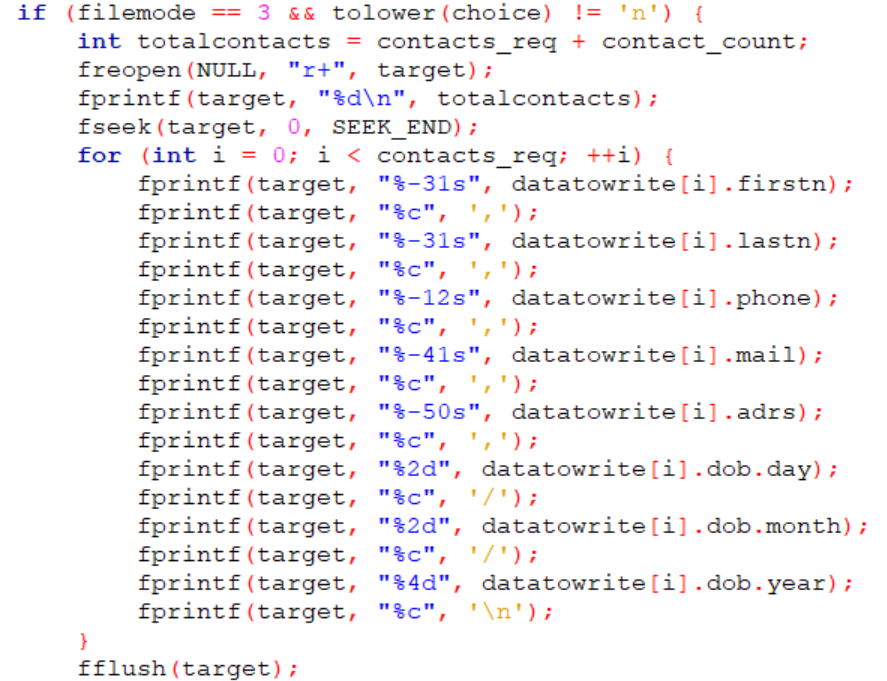


Contactsave is a void function that takes 3 parameter(pointer to structs that we want to save, number of records of these structs, pointer to the contacts file). It’s separated to 2 main blocks of code.

**1st block of code:**

* In this block of code, we choose filemode 2(write)
* The function reopens the file as write
* The first line in file will be its header (number of record)
* Using a for loop the record are written with the standardized length so the file can be random accessed.
* The file pointer is then flushed to the stream

**2nd block of code:**



* In this partition filemode is append (3)
* Function reopen the file as append and change the header to be number of total contacts (old +new)
* The function fseek is used to

move the file pointer toward the end and start writing the new records

* New records are written with the standardized form
* The file pointer is then flushed to the stream

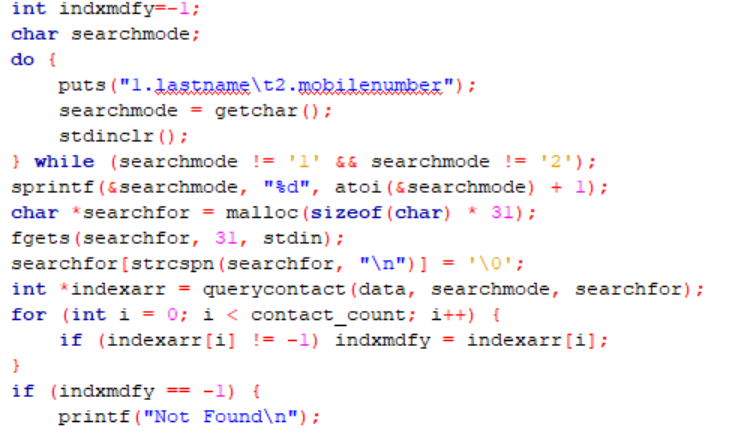
1. ***Log print:***



Log print is a void function that takes the contacts structs’ pointer as argument. This function prints all the records to the user on screen in the form of a table.

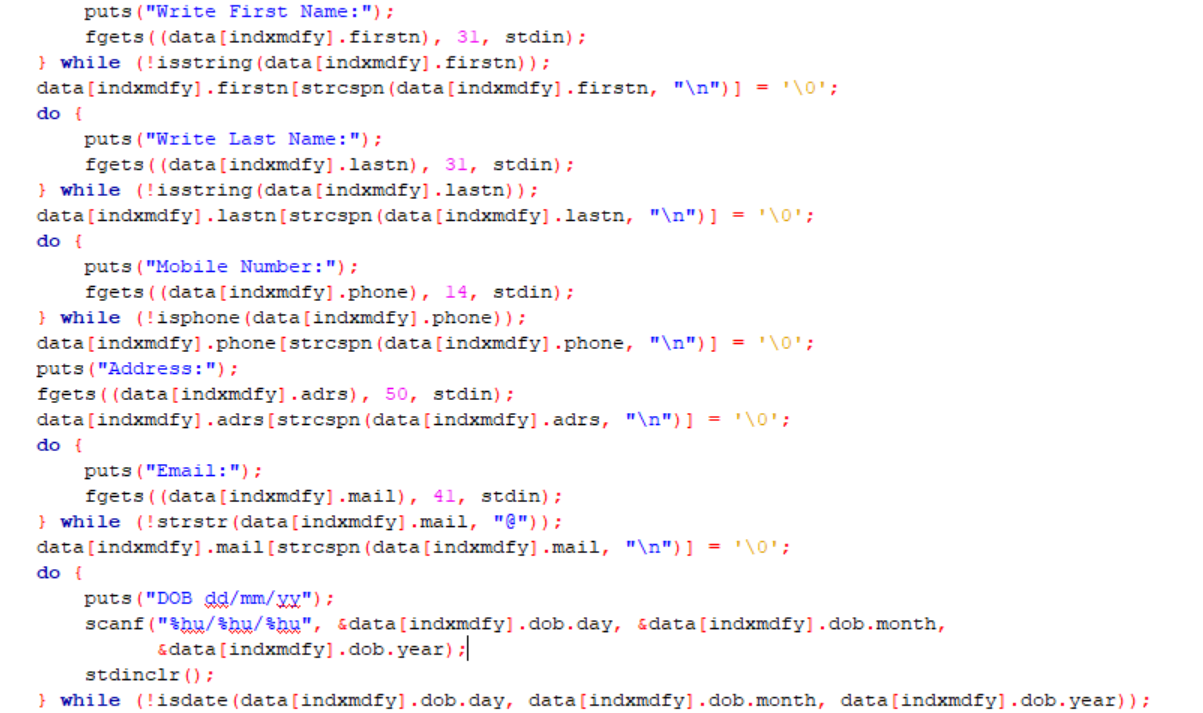
1. ***Modify contacts:***

Modify contact is a void function that takes the contacts structs’ pointer as argument. This function changes the name, phone… of a certain record. This module is divided into three main blocks of code (search, modify, save)

**1st block(search):**

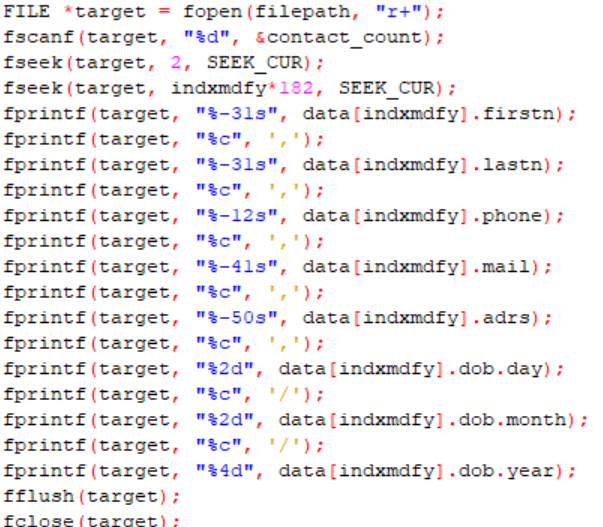
* The function asks the user whether field he wants to search by (last name /mobile).
* It scans the data the user enters and store in a string (because all fields of contacts is chars).
* The query function is called (it returns the record array index) [see query function]
* Then if found the indxmdfy stores the record number.
* If not puts not found.

**2nd block of code(modify):**



* In this part the selected record is modified.
* Again, every field is checked before it is saved into the record.

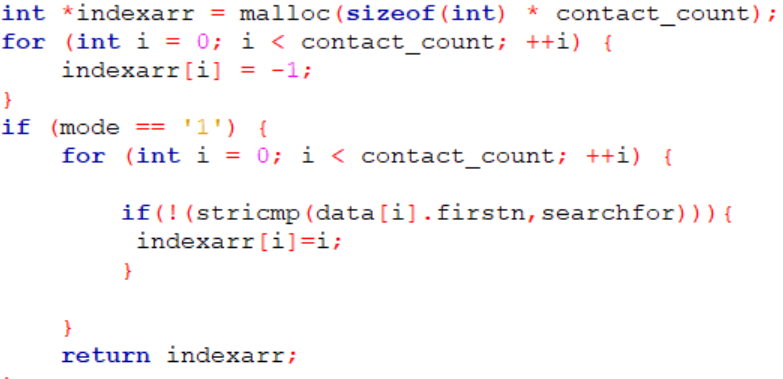
**3rd block of code(save):**

* The file is opened as append here
* The file pointer seeks the record number
* The record is saved to the files by the standardized form.

1. ***Contact Query:***



The contact query is an integer function that takes 3 parameters: pointer to records-a character that defines the mode of search and the string that we want to search for. It returns array of indices that matches your needed mode search target. It’s divided to 3 blocks of code with same concept:

* First, the function an array in the heap with number of elements equals to the number of the records and set all elements to be -1
* Depends on the mode the user chooses (1,2,3)

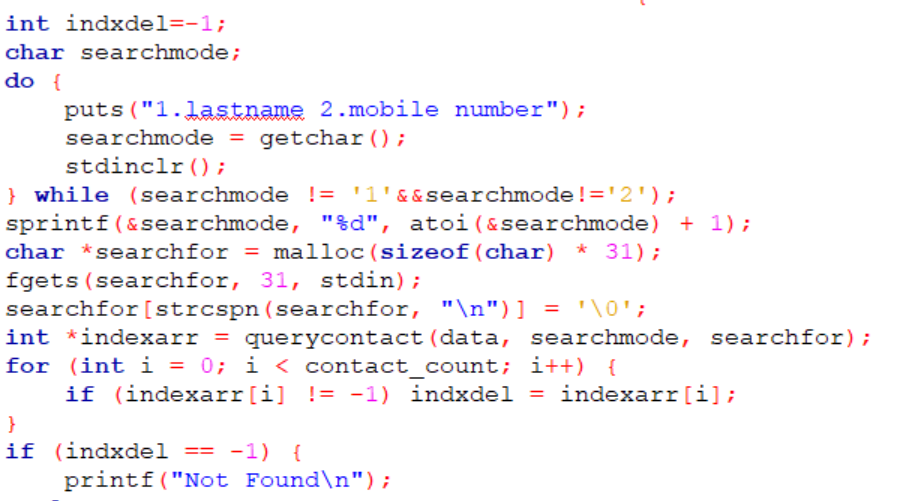
And by using a for loop the function loop on all records’ field

* Using a stricmp we compare the string entered by the user and the field (chosen field)
* If it matches, then the function assign number of this record to the index array.

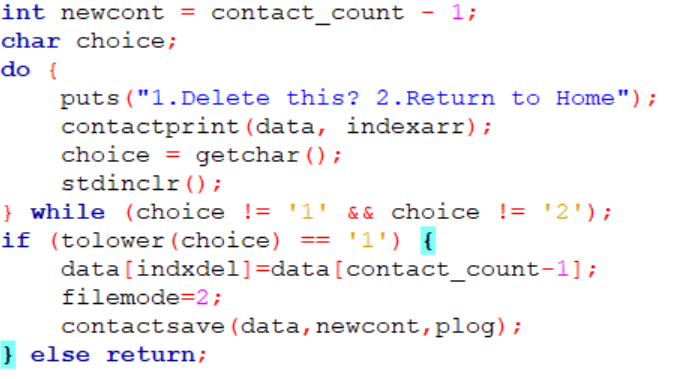
1. ***Delete contact:***



Delete contact is a void function that takes 2 parameters: the contacts structs’ pointer and file pointer. This module deletes a chosen unneeded record. This module is divided into two main blocks of code(search , delete).

**1st block (search code):**

* The function asks the user whether field he wants to search by (last name /mobile).
* It scans the data the user enters and store in a string (because all fields of contacts is chars).
* The query function is called (it returns the record array index) [see query function]
* Then if found the indxdel stores the record number
* If not puts not found

**2nd block (delete code):**

* A variable newcont is declared and the number of record minus the one deleted is assigned to it
* The contactsave function is called (in filemode2)
* Contactsave function overwrite the file.

1. ***Sort:***

The Data Array is sorted using the built in C function qsort ascendingly according to Firstname, Lastname, Date of Birth according to user input.Because qsort needs a comparator function to decide how to sort the elements and there are 3 sort modes ;we implemented function qsorthack as an intermediate between qsort and the the actual comparison function contactcmpr.

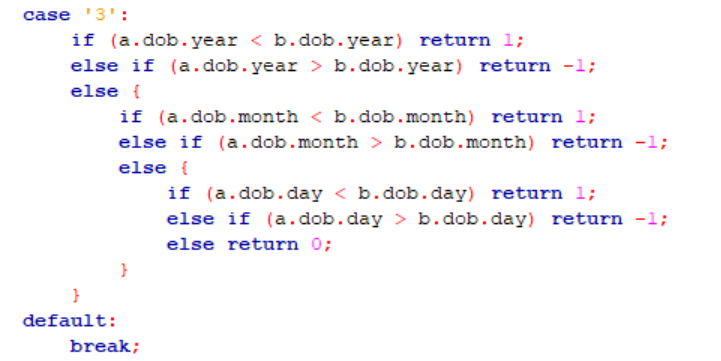
Contactcmpr has 2 modes:

***Case 1(comparing strings: names):***

* Using stricmp to compare between the 2 strings
* Stricmp loops on every character (same position) of both strings and compare their ASCII value
* Return 1 if string 1 is before string 2
* Return -1 if string 1 is after string 2
* Return 0 if string 1 is string 2

***Case 2(comparing integers: date):***

* Using a set of nested if conditions we compare each year, month and day
* Start by comparing years, if the 2 years are different, the function returns 1 or-1
* If both are the same year, the function enters the next if condition.
* Comparing 2 months, if the 2 months are different, the function returns 1 or-1
* If both are the same month, the function enters the next if condition and comparing the days.
* If the 2 days are the same the function returns 0.



**Features:**

***Data formatting:***

* Fgets and strscpn:

Fgets is a function that reads a line from the specified stream and stores it into the string pointed to by str. It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, it’s also safe. The problem faced that the user most probably stops, and press enter before the string ends, that makes the last character “\n”. we have used strscpn to replace the \n by \0 character.

* Getchar and stdinclr function:

Getchar is a function reads a single character from the standard input stream stdin and return it. The problem: getchar stores the char in the buffer, most probably the user after typing the character press enter; we have then to clear the stdin after getchar. We have implementing a stdinclr function for that purpose.



* Space print

Spaceprint is a void function that print spaces for log print so the contact can appear in table form.

* Replace

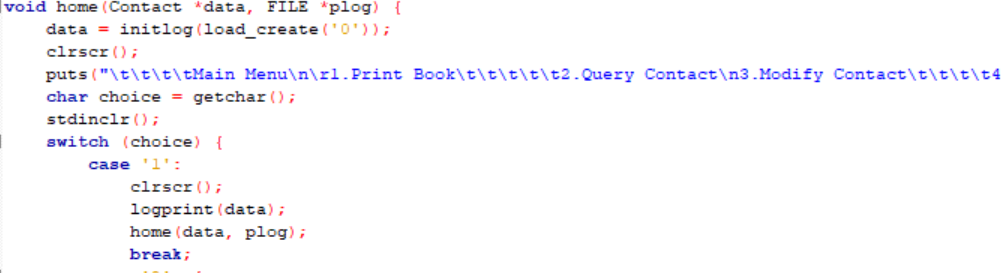
This function replaces a character in string by other character.

***User interface:***

* Home function:

We have implemented this function to act like the user interface function.

After the user choose to do any operation on the contacts the function call itself, back to main menu, until the user choose the exist option.



* Delay and sysclr:

We have used the delay and sysclr function to give the user a better user experience.

***The validation test:***

We have implemented several functions to check the validity:

* To check names: isstring

The isstring loop on the string to check whether there were any non-letters characters using isalpha.

* To check email: strstr

The strstr checks the existence of @.

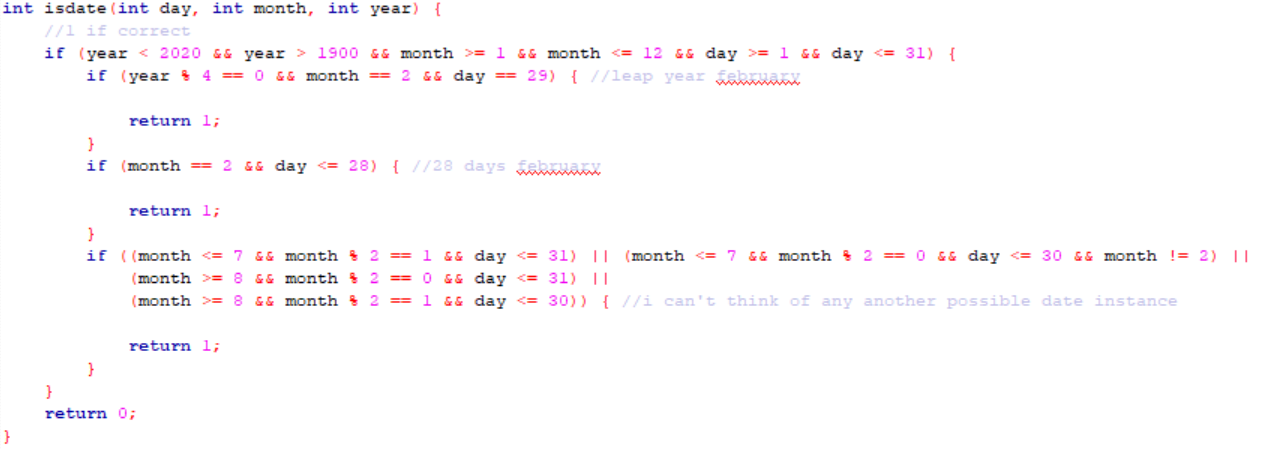
* To check phone number: isphone

The isphone loop on the string to check whether there were any non-digit characters using digit.

* To check date: is date

The isdate function checks if the year is between 1900 and 2020, the months are between 1and 12 and the days are between 1 and 31

Then we check the validity of dates with even and odd months.



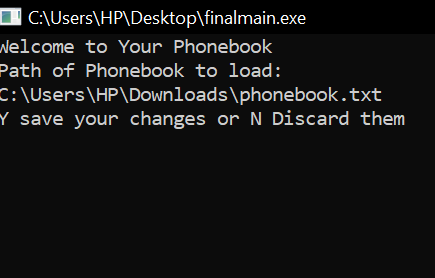
**Results: “FINALLY THE TESTING PART”**

The program works properly:

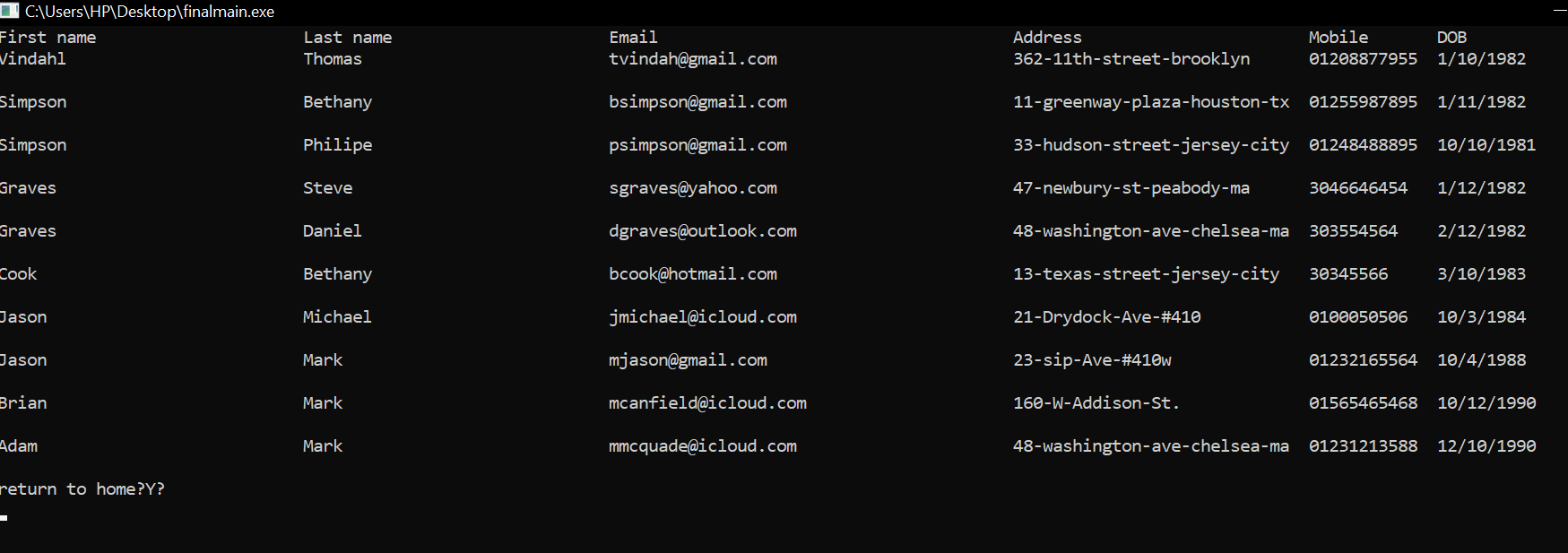
* ALL FUNCTION HAVE BEEN TESTED
* THE SEARCH OF A DUPLICATED CONTACT IS TESTED
* CREATING, READING AND APPENDING A FILE IS TESTED
* THE VALIDATION OF THE USER INPUT AND ERROR HANDLING ARE TESTED

AND HERE IS A BRIEF OF THE PROTOTYPE FILE TEST:

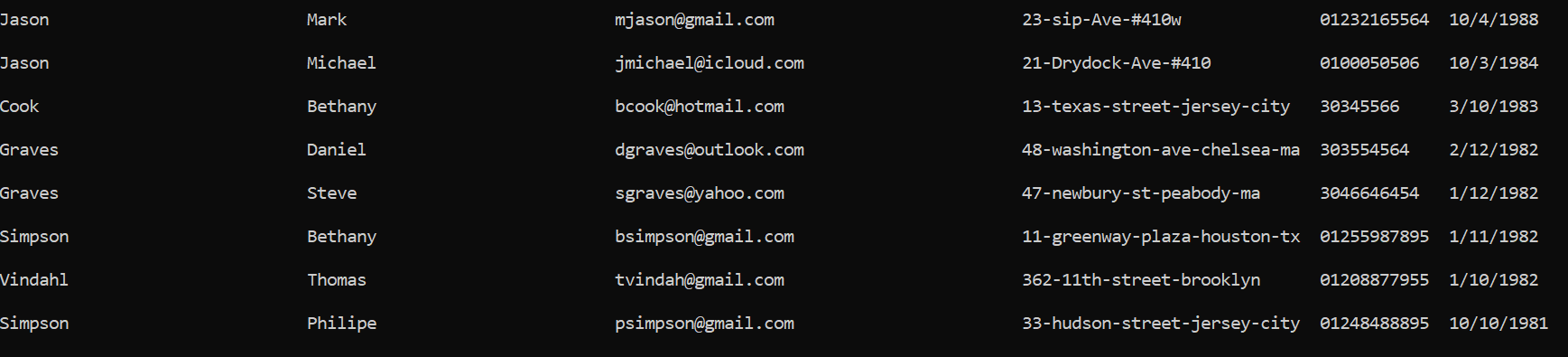
Welcome page



Print log



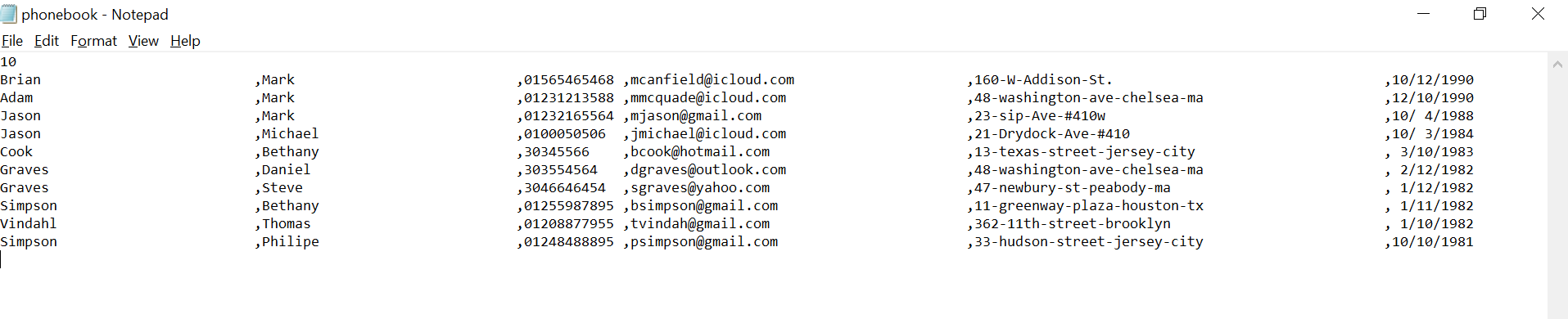
Sort log(by dates)



Query log



File saved data



**References:**

[**https://github.com/**](https://github.com/)

[**https://stackoverflow.com/**](https://stackoverflow.com/)

[**https://www.tutorialspoint.com/**](https://www.tutorialspoint.com/)