C- project

PROJECT
Phonebook

Presented by:

Mostafa Hakam	5525
Ayman Ahmed	5529
Ahmed Ashraf	5645

CONTENTS

Summary

We present a virtual phonebook program, Project PBook, that can be used on any PC to store, save and load contacts. Project PBook is a C program that can create and maintain a phone directory by loading and modifying text files that have contacts stored in a certain sequence. It can also search, add or delete contacts using certain functions.

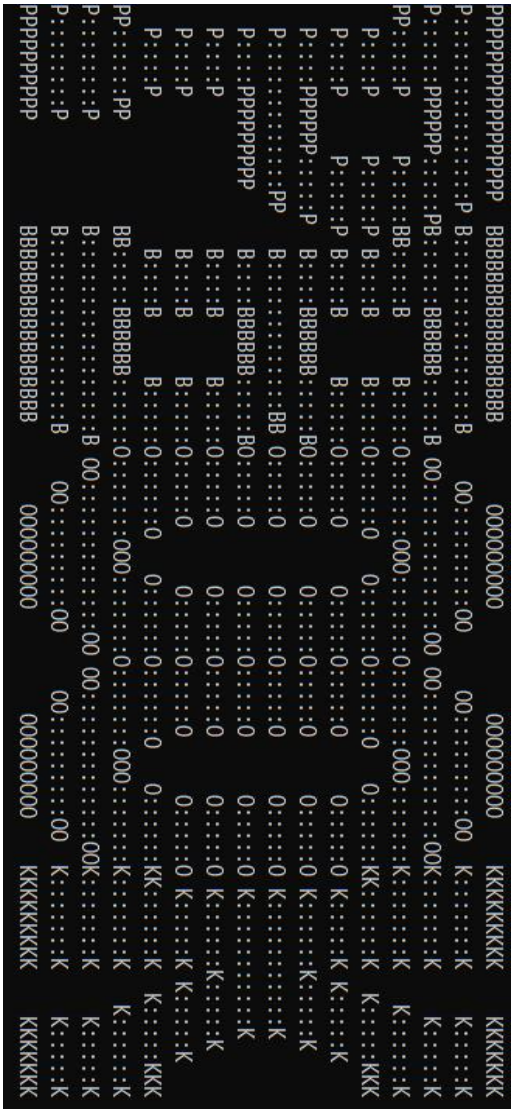
I. General

A global array of structs was used to handle multiple contacts, each contact was in form of a struct, ENTRY, containing multiple data types for each field provided including another struct, birthdate, containing the date of birth of a contact (day, month and year).

A text file is only used when the user loads or saves a text file through the program. Elsewhere, the contacts are stored in and dealt with through the global array of structs.

The “main” function’s role is to just transfer the user to the actual functions in the program by pressing any key.

Two functions were created (milliSecDelay() & secDelay()) to delays between two performances of two visual commands on the screen to make the program look more presentable.

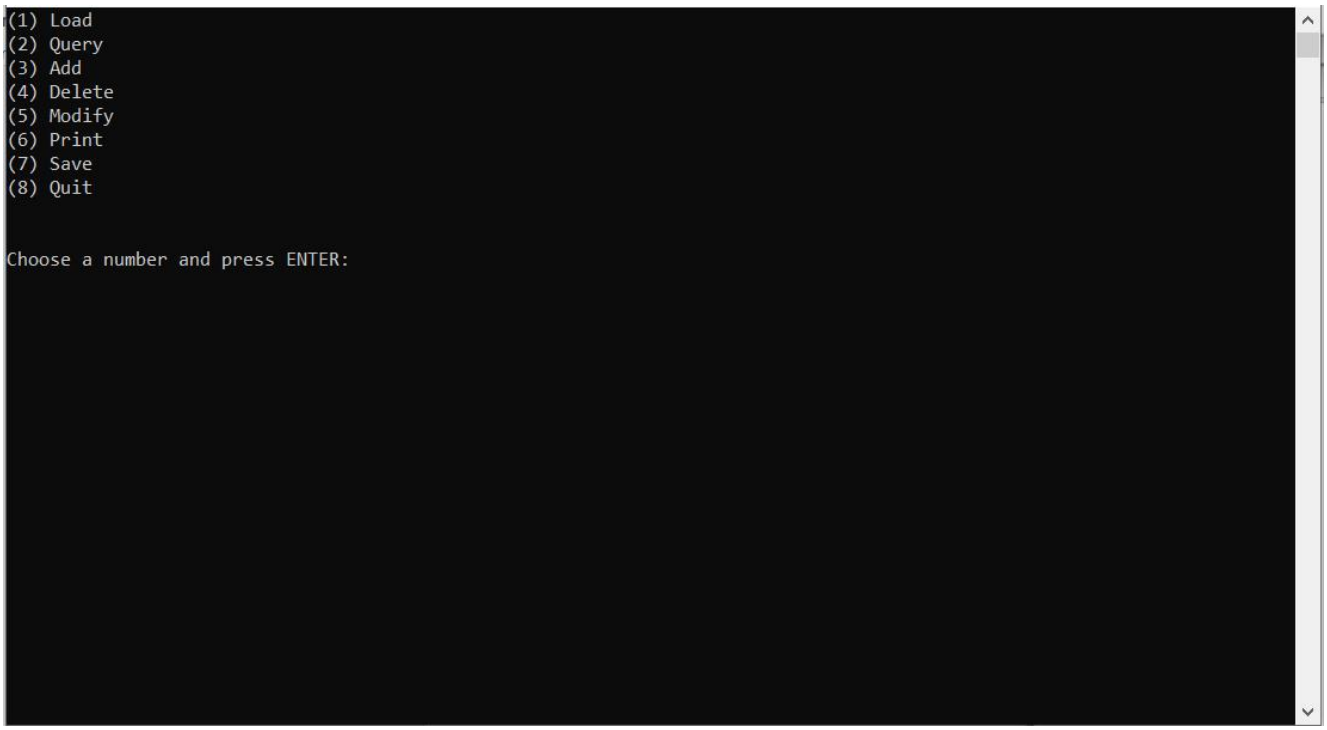


II. Main Menu

Main function, it gets the user access to the functions of their choice by entering the number corresponding to the desired operation. Switch with cases was used as a core idea to the function.

A validation was added using a while loop to detect any entry other than a number from 1 to 8 to force the user to enter a correct function number in order to access an option as well as giving them the option to go back to main menu or exit the program by entering 'y' or 'n' which is not case sensitive.

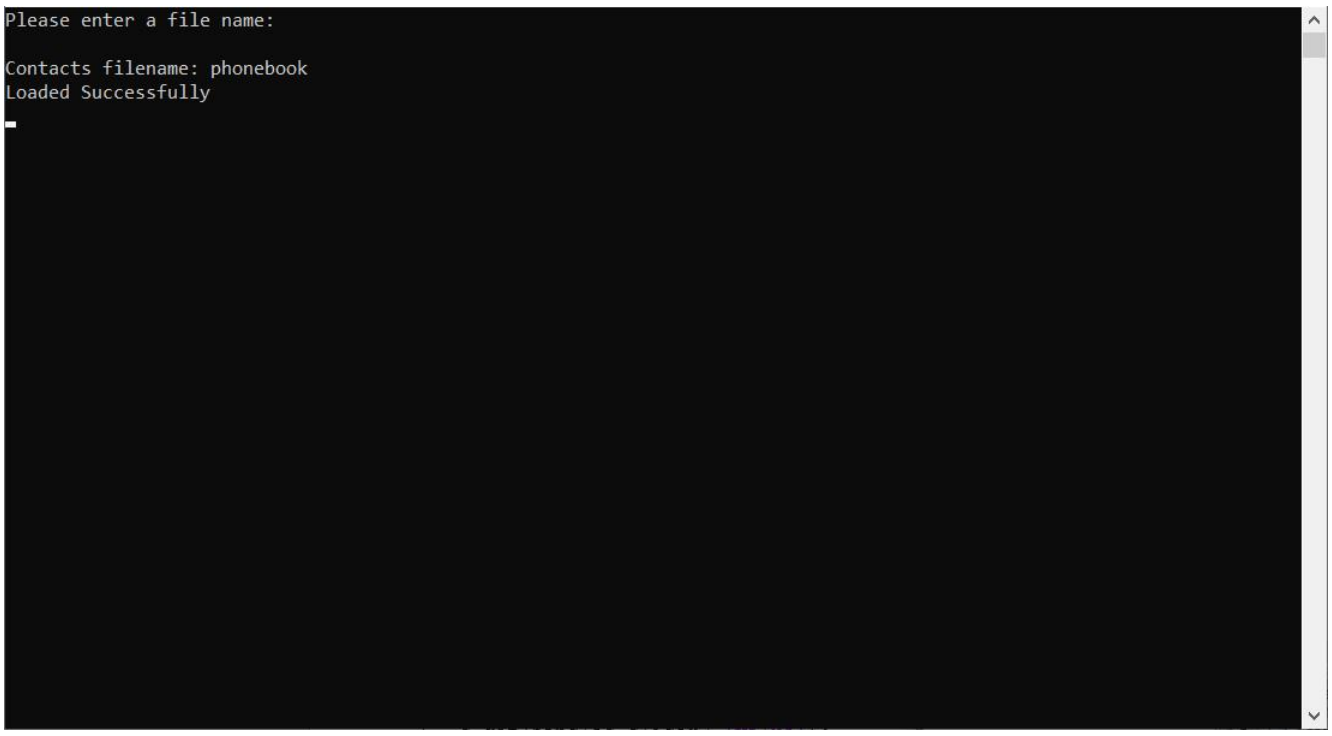
Free() function was used to prevent memory leak resulting from dynamic allocation and prevent duplication of array resulting from multiple usage of the load function.



III. Load & Save

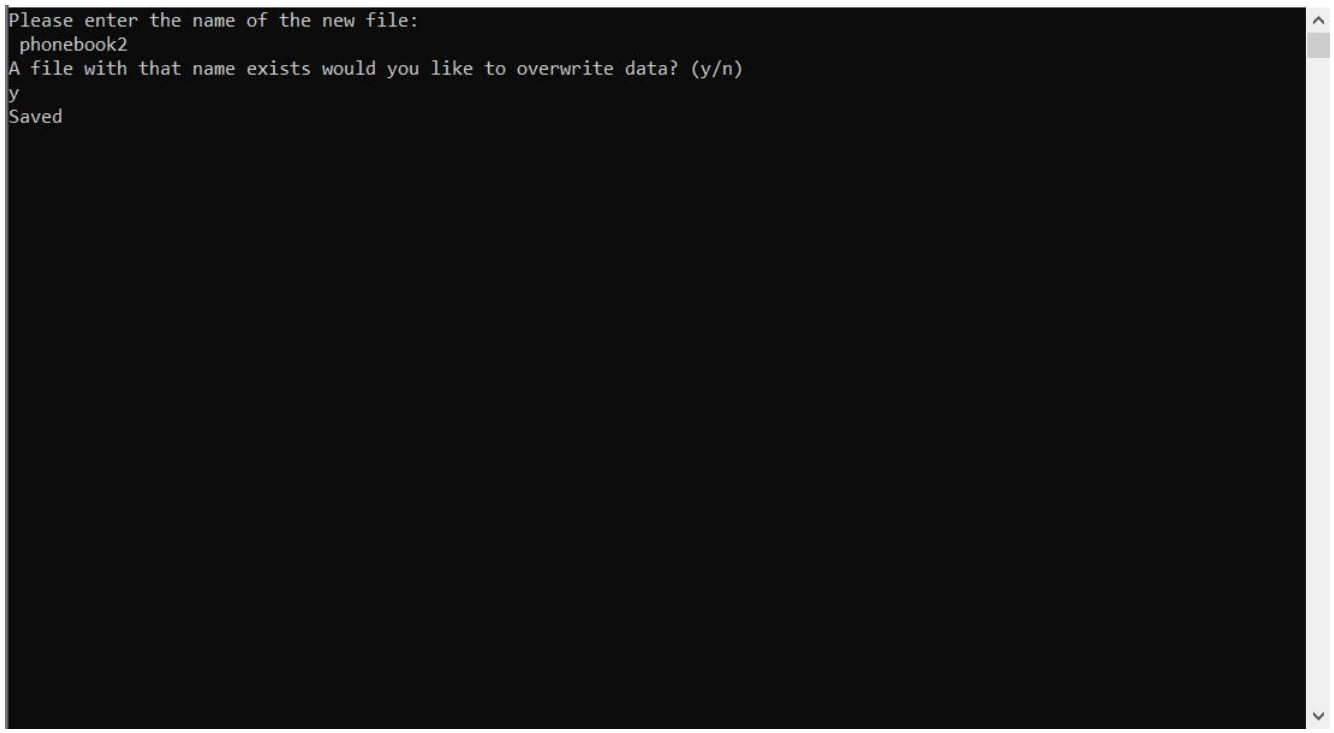
The only two functions assigned to manage external files. The load() function stores the data of a txt file into the global array (contacts) and the save() function stores the data of the array in a new txt file (or overwrite on an already existing txt file).

1. **load(): the user is asked to enter the desired file name as a string, then strcat() is used to append '.txt' to the name entered by the user in order to properly access the file which is opened in read mode 'r' and assigned to the global pointer (fptr). An if condition is used to check whether the entered file name is correct, and would return the user to the beginning of the function in case wrong entry was found. Finally, a while loop is used to store the contacts info, line by line with a coma delimiting each field, until end of file is reached, then the file is closed and the function concludes.**



III. Load & Save

2- save(): The save() function is used to save changes made from other different functions to a pre-loaded .txt file. It starts with opening the file and saving it to a pointer to file in the mode writing as it overwrites any existing data or creates a new file to save modified data. At first it asks the user if it wants to proceed in case he mistakenly chosen its ID, then if he chooses to proceed it asks him to enter the name of the file in which he wants the data to be saved, taking in consideration that if there exists a .txt file with the same name as the entered name, the user will be warned that the data is going to be overwritten, if he wishes to proceed, then it's going to write the data and closes the file after it finishes.



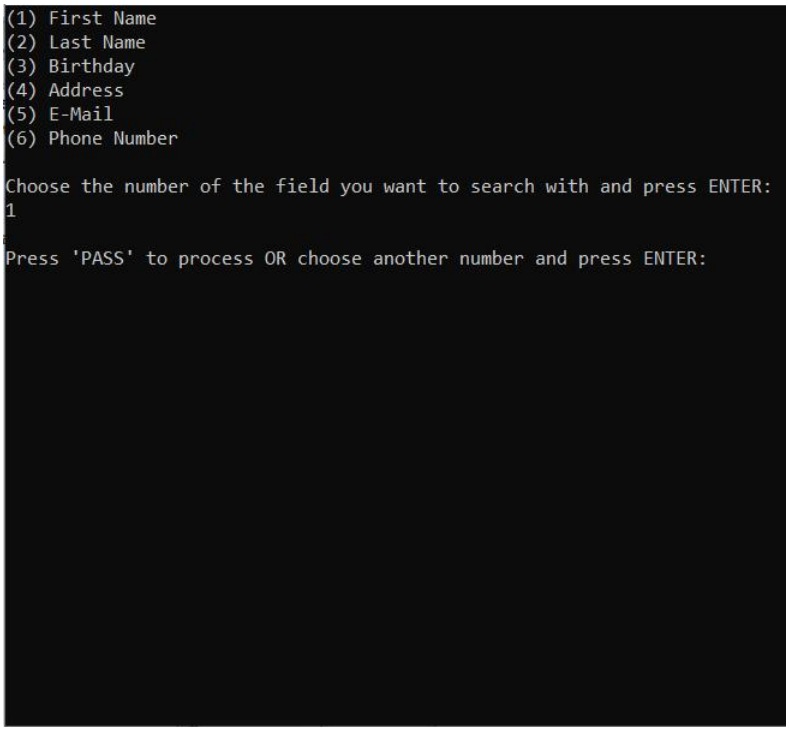
IV. Query

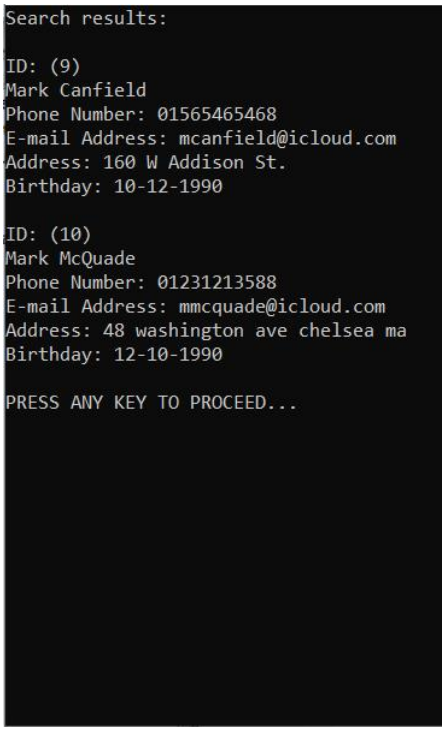
Core function, used to search for contacts using multi-search techniques. The system should process a request by the user to look up information about a specific entry. The user must choose the fields he wants to supply first using a switch that reads a number between 1-6 from the user corresponding to each field and then supply the chosen fields in form of strings.

The Query function creates a static array (id) that contains a number of elements equal to the number of elements of the global array (contacts) which all are assigned to the same value (1).

In a for loop for each supplied field, an 'if' condition occurs comparing the data stored in every contact to the data entered by the user, changing the value of the id of the contact if the data doesn't match.

In the end, a while loop is used to print only contacts that have id-s with value of 1.

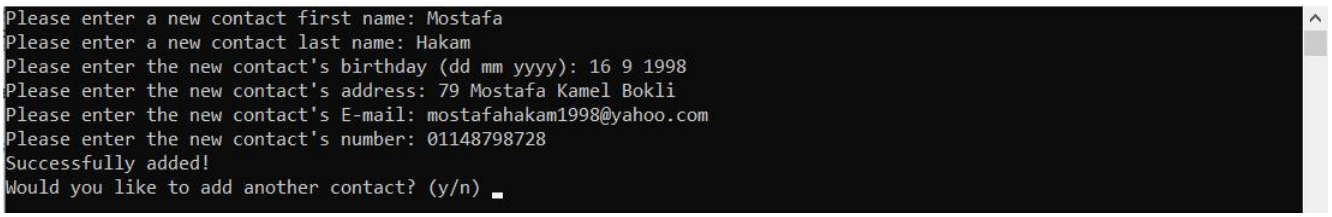




V. Add & Delete

Functions, by their name, to Add and Delete contacts to the array of structs (contacts). Before using any of these functions, the user will be warned that he didn't load a file yet then give him the choice to load a file using the function warning(). Once a file is loaded, he can proceed to the function he originally chose.

1. **add(): The user will be asked field by field for his new contact to be added, starting with the first name then passing by last name, birthday…etc.**
   The user must entered valid inputs to each of the fields as a valid name with no characters, only letters, numbers and spaces.
   For the birthdate the user must add a valid birthdate for a living person who witnessed at least the invention of phones and his birthday must be in a calendar. Ex: 29/2/2010 is invalid as 2010 wasn't a leap year, and so on. Also, must add a valid e-mail, and a valid number, then the contact will be added and the size will increase by 1, then the user will be asked if he wants to add another contact.
2. **del(): uses query() function to find contacts meant to be deleted. Once a desired contact is found, the user is asked to confirm the delete by providing a contact's ID, then a while loop runs so long as j is less than sizeOfarray, where j is the element number of the contact found. In case the user enters a wrong entry when confirming the delete or when entering the name or last name, or when the contact is not found i.e. the counter 'I' of the for loop is equal to the sizeOfarray then an appropriate message will be printed to guide the user.**



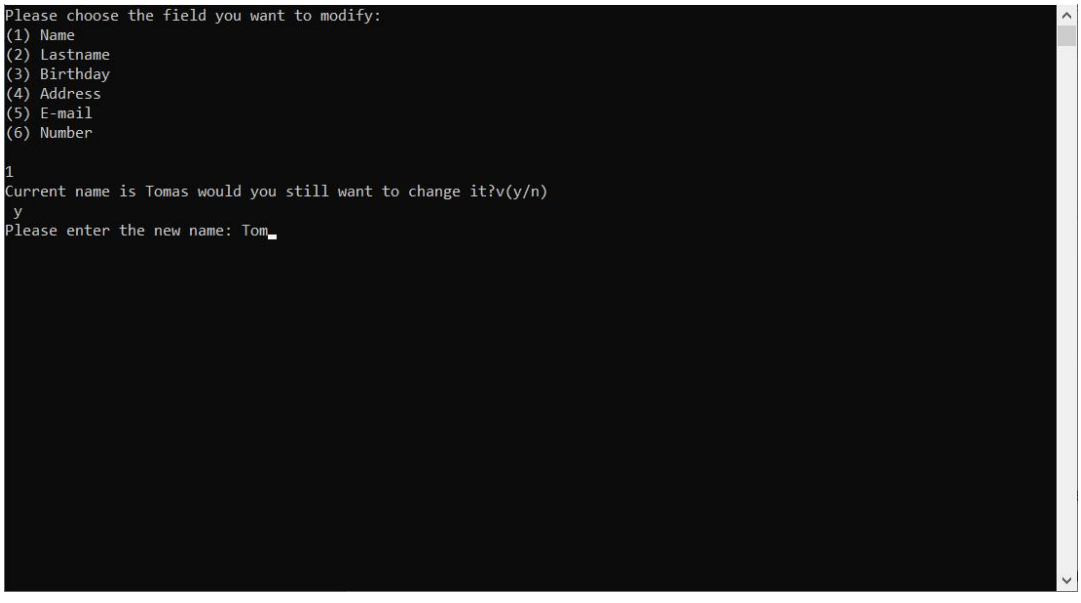
VI. Modify

Used to modify a field chosen by the user. Its actions are taken place within the ram and no changes are done to the file until it is saved.

At first the user is asked if he wishes to proceed in-case he mistakenly entered, the function will call the function query() to search for the contact he wishes to modify the user will be asked for the fields he wishes to use within his search and then will be given a list of all matching contacts with their respective IDs.

The user will enter the ID of the contact he wishes to edit if the user doesn't want to modify any of the contacts or he didn't find what he was looking for the user should enter 0 as it will give him the choice to re-search again.

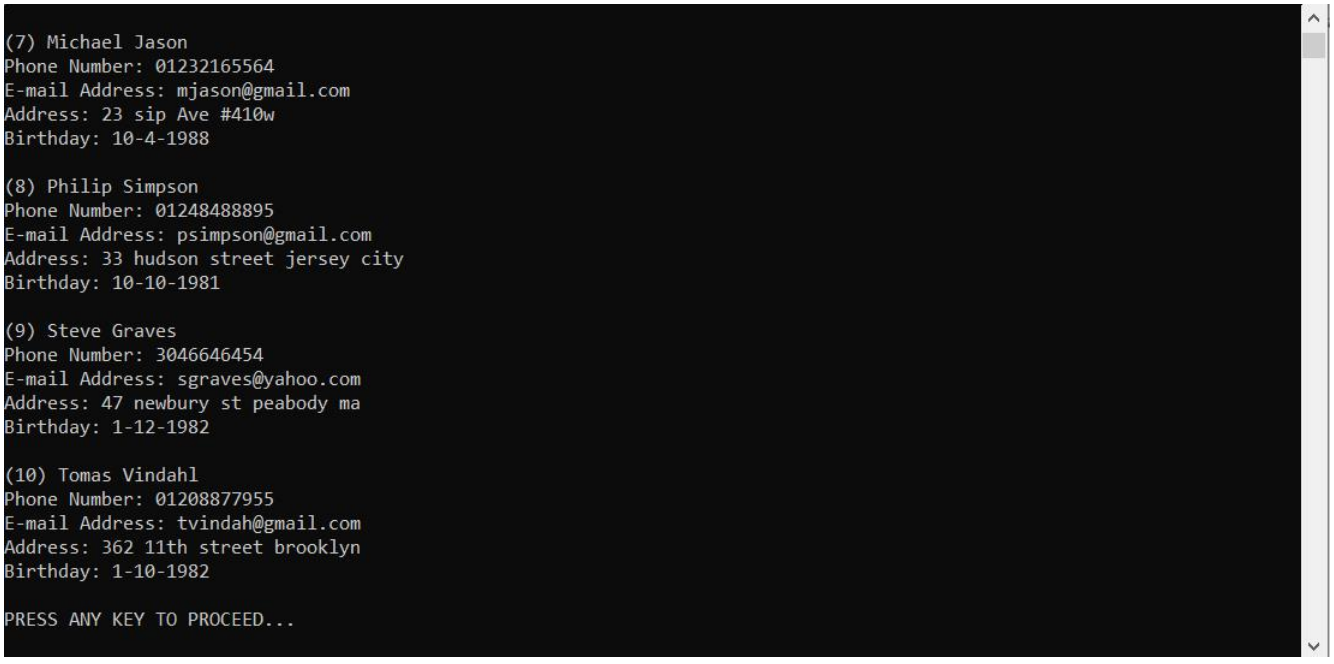
When the user chooses the field, the user will be given the current content of the field as a warning with a message that if he still wishes to change that field, if he chooses yes, they will be asked for the new content and then it will be successfully updated in the loaded contacts. Then the user will be asked if he wishes to modify any other field in the contact chosen, if he chooses yes, he will be shown the menu of fields and then proceed to change the desired field.



VII. Print and Sort

The print() function prints the entire directory in a sorted order using the getSorted() function.

1. **print(): The original array of structures for contacts is assigned to a temporary array of structures to avoid making changes in the originall array. The getsorted() function is be used on it to sort it and a for loop from 1 untill sizeOfarray is used to print the new sorted array field by field.**
   Finally the temporary array is freed to prevent memory leaking, and the function would terminate when the user presses any key.
2. **get sorted(): A question is first asked if the user would like to sort using default settings or according to birthday, the user is then free to choose.**
   On default settings, using bubble sort, it sorts the entire directory according to last name, first name then phone number. A condition is found in two nested loops to check wether two consecutive contacts are sorted or not, if not, they're swapped with the help of a 'temp' struct.



VIII. Quit

The quit() function is a basic function to allow the user to exit the program after using it. It prints a warning message to the user to remind them to save any changes made to the phonebook, user enters 'y' if they wish to confirm their exit or 'n' if they want to return to main menu.

In the case of any other input, a "invalid option" message would appear and the quit() function process would be repeated through recursion.

To exit, the exit(0) function was used, 'y' and 'n' are not case sensitive.

IX. Extra

1- validation_string(): function to take a pointer to character as an input as it indicates the start of the string we want to check, its return type is integer and it is used in functions: modify() and add() as they are the only functions that make the users make changes on the loaded contacts mechanism: it starts with a for loop that goes by every element of the string and tests if it is a letter, number or a space or it is a different non-wanted character using the ASCII numbers of their respective characters, if it is a non-wanted character it returns 0 instantly, if it's not the function will perform the test on the other elements, element by element until it reaches the size of string and then returns 0 if the loop finished as it cannot be finished if it has an unwanted character.

2- validations_number(): function take to a pointer to character as an input as it indicates the start of the string as we first defined the phone number as a string we want to check, its return type is integer and it is used in functions: modify() and add() as they are the only functions that make the users make changes on the loaded contacts mechanism: it starts with a small check if the number entered already exists in the loaded contacts, if it does it prints a message that the number already exists and return a 0 instantly, if it passes this test, a loop will occur that goes by the string element by element checking the ASCII number of every element if it corresponds a number or not, if not it return 0 instantly if it corresponds a number it proceeds until we reach the end of the string, if all elements passed the test it will return 1 as it indicated that it's a valid number.

IX. Extra

3- validation_email(): f unction to take a pointer to character as an input as it indicates the start of the string we want to check, its return type is integer and it is used in functions: modify() and add() as they are the only functions that make the users make changes on the loaded contacts mechanism: the function contains 3 main tests, the first test is a loop that goes by every element counting the number of @ characters, if the counter isn't equal 1 then that e-mail is not valid and it will return 0 instantly, the second test is a loop that goes by every element checking if there's 2 characters following each other or if there a character before or after the @ sign(We used the dot character as a sample, it applies for all characters depending on the domain used for emails), if it doesn't pass the test the function will return 0 instantly if it passed then the last test will take place as it checks if there is a valid ending like .com, .net…etc if it passed that last test the function will return 1 indicating that the email entered in valid.

4- validation_birthdate(): Function to take a date type which is a structure containing three integer members, day, month and year, its return type is integer it is used in functions: modify() and add() as they are the only functions that make the users make changes on the loaded contacts mechanism the function first makes the test on the year if it is less than 1800 which is nonsense people that lived in 1800 died before the invention of the telephone and if it is greater than 2018 which is also nonsense because a baby that has less than a year wouldn't have a phone number, the second test is a test on a month to check if it's valid like no month greater than 12 and not less than 1 if it passed that test a last test will occur which is a test on day if the day entered is 31 and it's not 31-days month it will return 0 also if the user entered the day 29 and the month is 2 and it's not a leap year the function will return 0, if the date passed it will return 1 as it indicates the validation of the date entered

IX. Extra

5- warning(): an essential function mainly used if the user wants to perform any act without loading a file which is a common mistake. The function will inform him to load his contacts first before trying to perform any act mechanism first it checks on the global variable (fptr) if it is equal null or has a value, if it's equal null then the function will print a warning that the user didn't load any contacts yet, and then asks him if he wishes to load his contacts if he chooses to load his contacts he will be sent directly to function load and then proceeds to the function he entered first, if he chooses not to, the user will be sent to function quit.