Alexandria University
Faculty of Engineering
Computers and Communication Engineering Program

Company Management System

Project and Report

Course Title: Programming - I

Course Code: CSE 126

Fall 2021

By

Omar Ibrahim Elsayed Maarouf

7442

Mazen Nagy Mansour

7475

Work Done

A management system for companies is built to help the user process the data of the employees in the company easily.

Steps of Work

First, we defined two types of structures (Date and Employee). Date structure contains three variables of data type integers to store three numerical integers corresponding to the day, month and year of the defined date respectively. Employee structure contains five variables of array of characters (strings) that represent the last name with 40 characters, first name with 40 characters, address with 100 characters, phone number with 12 characters and e-mail address with 40 characters of each employee, one integer variable representing id, one floating number variable corresponding to the salary and one date variable representing the birthdate.

We defined an array of employees containing 100 elements, eleven array of characters corresponding to each entry given by the user: Id of 10 characters, Last name and First name each of 40 characters, Salary of 10 characters, Address of 100 characters, Day and Month each of 3 characters, Year of 5 characters, Phone number of 12 characters and E-mail of 40 characters. The previous are defined as global variables in addition to two counters n (corresponding to number of current employees) and c (corresponding to user’s choice) and initializing both with 0. Defining them globally will allow them to be modified directly without the must of sending them to each function.

Second, we built ten validation functions aiming to check the validity of each data of the employee entered by the user. Two functions used for checking that the entry is numerical data only (Integer validation) and that the entry is a decimal data (Float validation). The rest of the functions validate the name of the file containing the list of employees, employee’s id, employee’s last and first names, employee’s salary, employee’s date of birth, employee’s phone number and employee’s e-mail.

- Integer validation: this function returns 1 if all the string’s characters are between ‘0’ and ‘9’. It returns 0 if the previous condition is false.
- Float validation: this function returns 1 if all the string’s characters are between ‘0’ and ‘9’ and contains at most one decimal point ‘.’. It returns 0 if any of the previous conditions is false.
- Filename validation: this function returns 1 if the string containing the filename ends with the substring “.txt”. It returns 0 if the previous condition is false.
- ID validation: this function sends the string containing the id entered by the user to the function validating integer entry and returns the same value returned by the integer validation function.
- First and Last names validation: these two functions check that the strings containing both information consists only of alphabetical letters (whether upper or lower cases) and dash character ‘-‘. It returns 1 if the previous is true and 0 if false.

- Salary validation: this function sends the string containing the salary entered by the user to the float validation function and returns the same value returned by the float validation function.
- Date of Birth validation: this function checks that the entries of Day, Month and Year fields are integer entries where the day must belong to the interval [1,31], the month must be within the interval [1,12] and the year entry must be between [1962, 2003] based on the work laws. It returns 1 if all the previous conditions are true or 0 if any of them is false.
- Phone number validation: this function checks that the string contains exactly eleven digits between '0' and '9' and the first two characters are '0' and '1'. If all these conditions are true, 1 is returned, else, 0 is returned.
- E-mail validation: this function checks that the string holding the e-mail address must contain an at character '@' and at least one character after that character and must ends with a dot '.' and three or two characters after it. It returns 1 if all conditions are true and 0 if not.

Then, we built a group of functions to hold each task called by the user.

- Load: this function gets the name of file and open it in the read mode and reads the record of the employee in the form of "ID,Last Name,First Name,Salary,Date of Birth as (Day-Month-Year),Address,Phone Number,E-mail" and cut it using the built in function (strtok) and storing each data in its place for the employee. This function also increments the counter for the number of employees by one for each rotation. The loop terminates when the internally defined character reads an end of file (EOF) character. If this defined character doesn't find (EOF) the cursor of file is back to its previous position to read the following record correctly. This function returns 1 if the data is successfully located and exits the whole program if the file can't be opened.
- Print: this function prints the data of the employee sent to this function in a well-organized way. This function is called at the end of the Query function, at the end of the Add function, in the middle of the Delete function, in the middle and at the end of the Modify function and in the sort functions.
- Query: this functions gets a last name from the user and check its validation. If it is invalid, the user is asked to re-enter the name again. If the entered name is valid, the function compares the last names of the current employees stored in the array with the entered last name and print every student having similar last name.
- Add: this function gets the information of the new employee from the user and validate each of them. If any entry is invalid, the user will be asked to re-enter this piece of information again. Last, the added employee's information is printed to the user, the counter representing the number of current employees is incremented by 1 and the employee is added at the end of the array. The user is offered that if he entered the first letter of the added employee's first or last name, they are automatically converted to upper cases.

- Delete: this function gets from the user first and last names and validate them. If they are valid, it looks for every employee having similar first and last names to that entered by the user. If employee is found, the user is asked to confirm deleting this employee or to cancel. If he confirmed deleting, the next employees in the array are shifted to the previous index in the list till the last employee is shifted and the counter (n) is decremented by 1.
- Modify: this function gets id from the user and validate. If the entry is valid, the user will see the employee to confirm modifying his information or to skip. For modifying, the user will be asked in each field of data to enter 1 to modify or 0 to skip to the next field. The user is offered that if he entered the first letter of the employee's first or last name in lower case, they are automatically converted to upper cases.
- Sort by Last name: this function sorts the array of employees alphabetically according to their last names. The function compares each two employees, the one with smaller alphabetical indication is brought to the top. At the end the list of employees is printed to the user after sorting.
- Sort by Date of birth: this function sorts the list of employees from elder to younger. The function compares each two employees, the one with smaller value in year field is brought to the top. If two employees have the same year of birth, they are compared according to the month of birth and the one with smaller value in month field is brought to the top. If both have same month of birth, they are sorted that the one with smaller value in day field is brought to the top. At the end, the list of employees is printed to the user after sorting.
- Sort by Salary: this function sorts the list of employees in a descending order with respect to the salary of each employee. Each two employees are compared and the one with higher salary is brought to the top. At the end, the list of employees is printed to the user after sorting.
- Save: this function re-opens the file opened in the load function but this time in the write mode. The array of employees is printed in the file in a similar form to that in the file before editing. At the end of this function, the user can choose to quit directly or return to the main menu.
- Exit: this function asks the user to choose whether to save then quit, quit without saving or return to main menu. The user can make any of the choices by pressing 1 or 0 as illustrated in the user manual.
- Main: in this function the user will enter the name of file to work on. The program will proceed if the filename is valid and if the file is successfully opened and loaded. Then, the user will be asked to enter a choice from 1 to 7 to choose the task to be held according to the order illustrated in the user manual. The value of the choice (stored in c) is compared using the built in comparison statement (switch). The switch statement is repeated using a (do….while) loop statement with a tautology condition "while(1);", as the loop is in continuous execution until the program is exited using the Exit function.

Sample Run

You are firstly greeted with a message asking you to enter an already existing file of text type (.txt) on your device containing the required information to get underway. The program is very user friendly and without being nonchalant to tick all the right boxes.

For example, as shown in the following figures (Fig. 1.1, 1.2, 1.3, 1.4, 1.5) the user is asked to enter a proper data in each field.

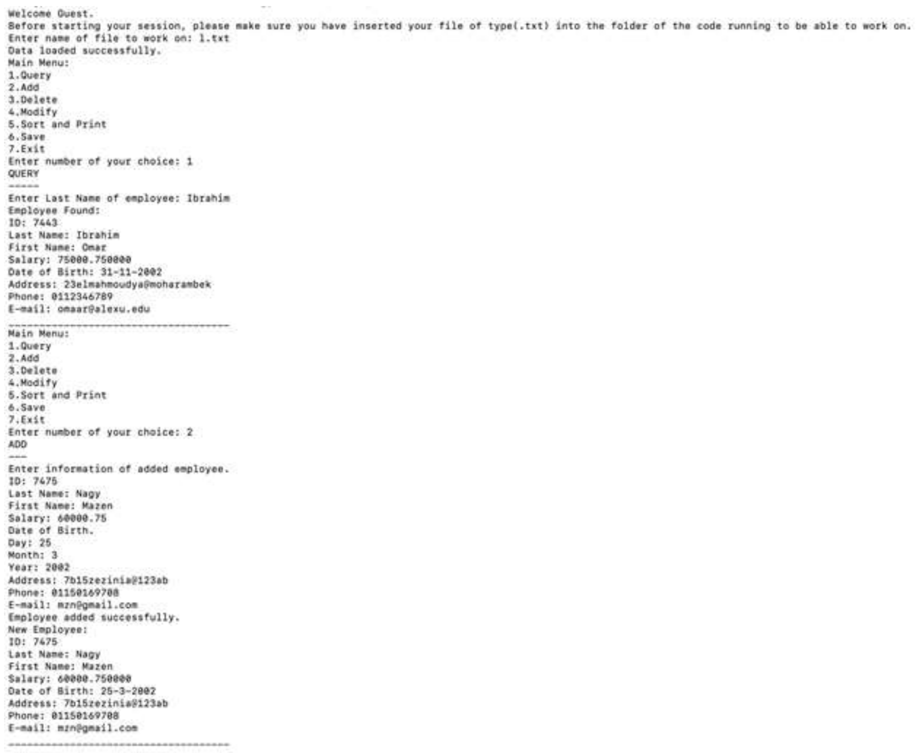
In figure 1.1, the user entered (1) in the main menu to perform a search using a valid last name. Then he entered (2) in the main menu to add an employee by entering valid data.

In figure 1.2, the user entered (3) in the main menu to delete an employee and the system checked for the last and first names of the employee to be deleted and that they are valid entries.

In figure 1.3, the user entered (4) in the main menu to modify the employee having the id entered by the user after validating it. For each field in the employee data the system will skip if the user entered (0) and will acquire a new entry and validate it if the user entered (1).

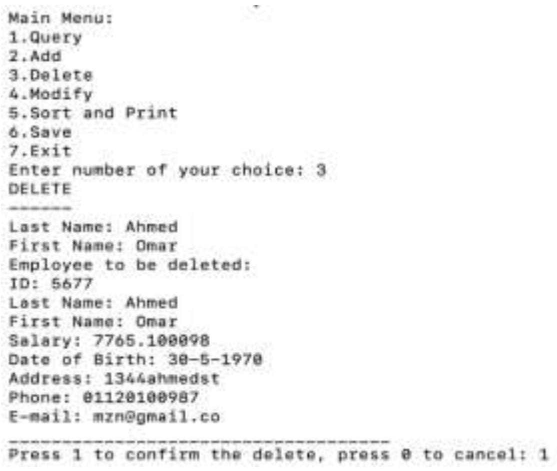
In figure 1.4, the user entered (5) in the main menu to sort the current employees. Then he entered (2) to sort according to the date of birth of each employee. After, he entered (6) in the main menu to save the data. We can save by pressing the save command or if the user tries to exit without saving, the system will ask him if being sure of quitting without saving with offering the option to quit or save and quit or return to main menu.

In figure 1.5, a sample of the data after modification and saving in the original file.



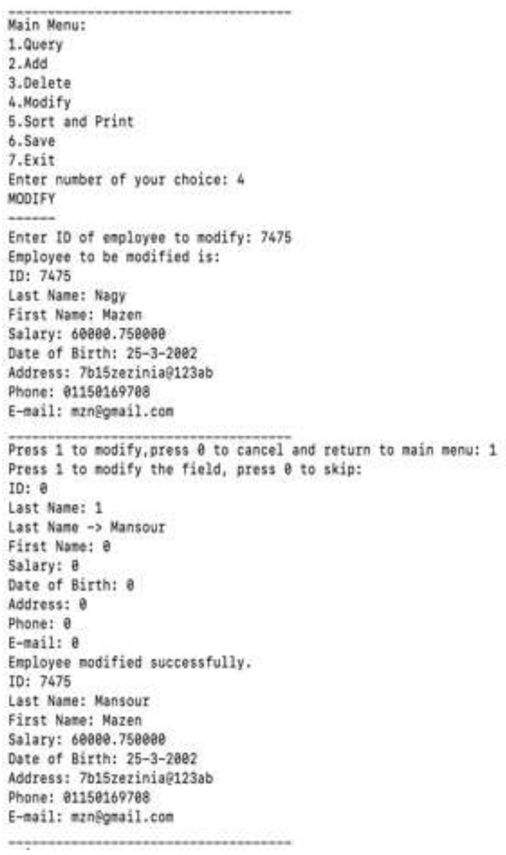
```
Welcome Guest.
Before starting your session, please make sure you have inserted your file of type(.txt) into the folder of the code running to be able to work on.
Enter name of file to work on: 1.txt
Data loaded successfully.
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 1
QUERY
-----
Enter Last Name of employee: Ibrahim
Employee Found:
ID: 7443
Last Name: Ibrahim
First Name: Omar
Salary: 75000.750000
Date of Birth: 31-11-2002
Address: 23elmahmoudya@moharambek
Phone: 0112346789
E-mail: omaar@alexu.edu
------------------------------------
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 2
ADD
---
Enter information of added employee.
ID: 7475
Last Name: Nagy
First Name: Mazen
Salary: 60000.75
Date of Birth.
Day: 25
Month: 3
Year: 2002
Address: 7b15zezinia@123ab
Phone: 01150169708
E-mail: mzn@gmail.com
Employee added successfully.
New Employee:
ID: 7475
Last Name: Nagy
First Name: Mazen
Salary: 60000.750000
Date of Birth: 25-3-2002
Address: 7b15zezinia@123ab
Phone: 01150169708
E-mail: mzn@gmail.com
------------------------------------
```

Fig. 1.1: Sample run for the Query and Add functions



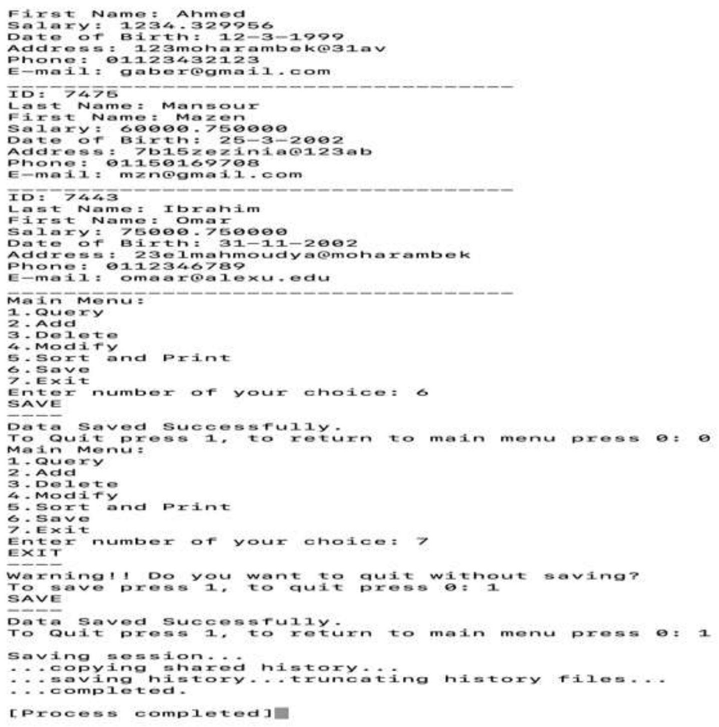
```
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 3
DELETE
------
Last Name: Ahmed
First Name: Omar
Employee to be deleted:
ID: 5677
Last Name: Ahmed
First Name: Omar
Salary: 7765.100098
Date of Birth: 30-5-1970
Address: 1344ahmedst
Phone: 01120100987
E-mail: mzn@gmail.co
------------------------------------
Press 1 to confirm the delete, press 0 to cancel: 1
```

Fig. 1.2: Sample run for the Delete function



```
-------------------------------------
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 4
MODIFY
------
Enter ID of employee to modify: 7475
Employee to be modified is:
ID: 7475
Last Name: Nagy
First Name: Mazen
Salary: 60000.750000
Date of Birth: 25-3-2002
Address: 7b15zezinia@123ab
Phone: 01150169708
E-mail: mzn@gmail.com
-------------------------------------
Press 1 to modify,press 0 to cancel and return to main menu: 1
Press 1 to modify the field, press 0 to skip:
ID: 0
Last Name: 1
Last Name -> Mansour
First Name: 0
Salary: 0
Date of Birth: 0
Address: 0
Phone: 0
E-mail: 0
Employee modified successfully.
ID: 7475
Last Name: Mansour
First Name: Mazen
Salary: 60000.750000
Date of Birth: 25-3-2002
Address: 7b15zezinia@123ab
Phone: 01150169708
E-mail: mzn@gmail.com
-------------------------------------
```

Fig. 1.3: Sample run for the Modify function

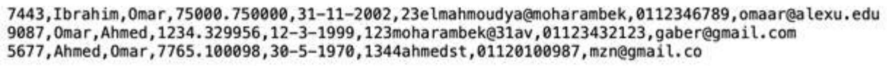


```
First Name: Ahmed
Salary: 1234.329956
Date of Birth: 12-3-1999
Address: 123moharambek@31av
Phone: 01123432123
E-mail: gaber@gmail.com
_____________________________________
ID: 7475
Last Name: Mansour
First Name: Mazen
Salary: 60000.750000
Date of Birth: 25-3-2002
Address: 7b15zezinia@123ab
Phone: 01150169708
E-mail: mzn@gmail.com
_____________________________________
ID: 7443
Last Name: Ibrahim
First Name: Omar
Salary: 75000.750000
Date of Birth: 31-11-2002
Address: 23elmahmoudya@moharambek
Phone: 0112346789
E-mail: omaar@alexu.edu
_____________________________________
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 6
SAVE
----
Data Saved Successfully.
To Quit press 1, to return to main menu press 0: 0
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 7
EXIT
----
Warning!! Do you want to quit without saving?
To save press 1, to quit press 0: 1
SAVE
----
Data Saved Successfully.
To Quit press 1, to return to main menu press 0: 1

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

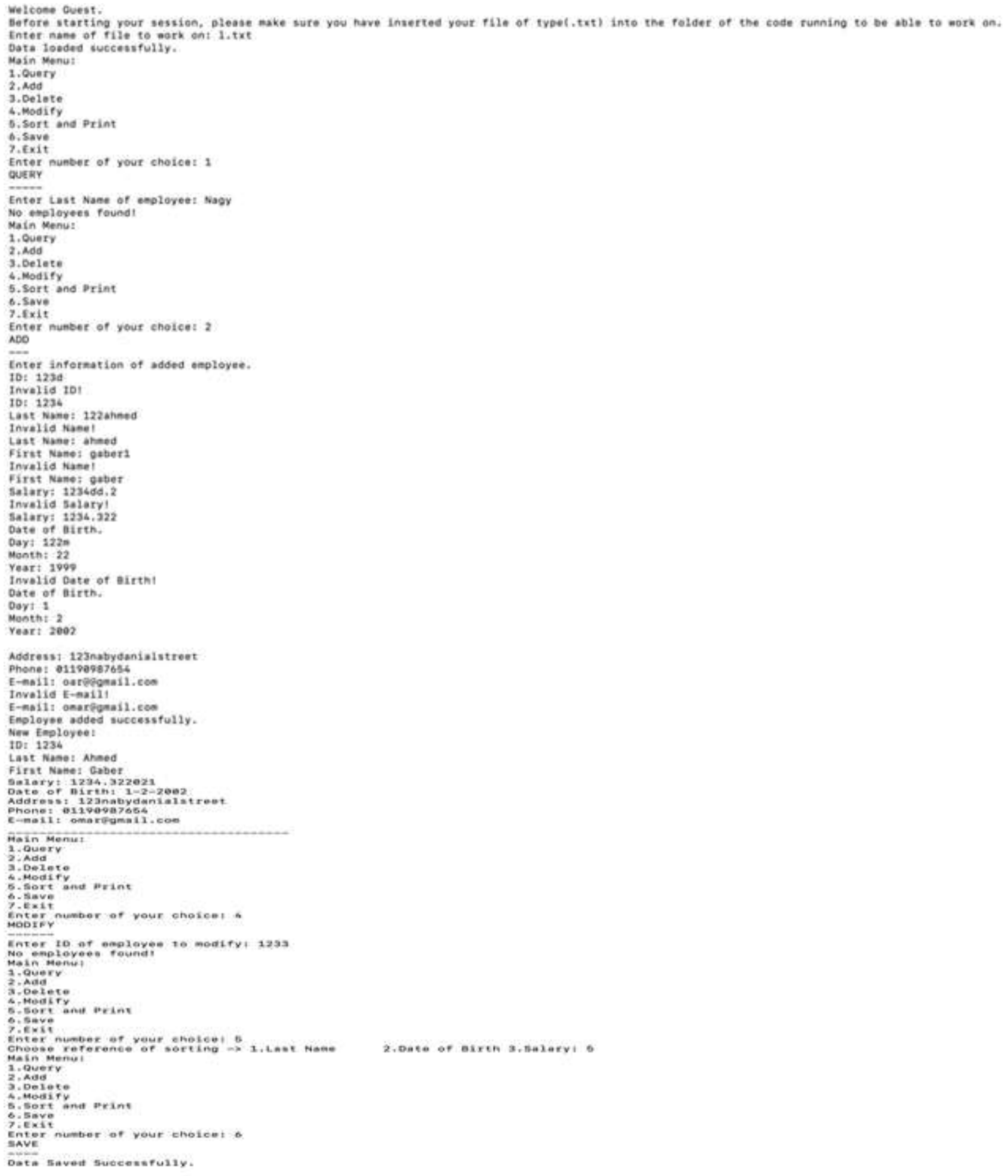
Fig. 1.4: Sample run for the Sort by Date of birth and Save functions



```
7443,Ibrahim,Omar,75000.750000,31-11-2002,23elmahmoudya@moharambek,0112346789,omaar@alexu.edu
9087,Omar,Ahmed,1234.329956,12-3-1999,123moharambek@31av,01123432123,gaber@gmail.com
5677,Ahmed,Omar,7765.100098,30-5-1970,1344ahmedst,01120100987,mzn@gmail.co
```

Fig. 1.5: Data after saving

For every wrong entry, the system will automatically compromise and ask for the correct as shown in the following figure (Fig. 1.6):



```
Welcome Guest.
Before starting your session, please make sure you have inserted your file of type(.txt) into the folder of the code running to be able to work on.
Enter name of file to work on: 1.txt
Data loaded successfully.
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 1
QUERY
-----
Enter Last Name of employee: Nagy
No employees found!
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 2
ADD
---
Enter information of added employee.
ID: 123d
Invalid ID!
ID: 1234
Last Name: 122ahmed
Invalid Name!
Last Name: ahmed
First Name: gaber1
Invalid Name!
First Name: gaber
Salary: 1234dd.2
Invalid Salary!
Salary: 1234.322
Date of Birth.
Day: 122m
Month: 22
Year: 1999
Invalid Date of Birth!
Date of Birth.
Day: 1
Month: 2
Year: 2002

Address: 123nabydanialstreet
Phone: 01190987654
E-mail: oar@@gmail.com
Invalid E-mail!
E-mail: omar@gmail.com
Employee added successfully.
New Employee:
ID: 1234
Last Name: Ahmed
First Name: Gaber
Salary: 1234.322021
Date of Birth: 1-2-2002
Address: 123nabydanialstreet
Phone: 01190987654
E-mail: omar@gmail.com
------------------------------------
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 4
MODIFY
------
Enter ID of employee to modify: 1233
No employees found!
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 5
Choose reference of sorting -> 1.Last Name       2.Date of Birth 3.Salary: 6
Main Menu:
1.Query
2.Add
3.Delete
4.Modify
5.Sort and Print
6.Save
7.Exit
Enter number of your choice: 6
SAVE
----
Data Saved Successfully.
```

Fig. 1.6: Sample run using invalid entries

User Manual

Steps

1. Enter the name of file containing the data correctly to be loaded and you can process them successfully. Your filename must end with ".txt". If the messaged (Error occurred while loading file "filename"!) this will automatically exit your program to give you the opportunity to correct any mistake done in installing the file before running the code. If the message (Data loaded successfully.) appeared, then you are now able to edit the records of the employees of the company.
2. A list of commands will appear numbered from 1 to 7 for Query, Add, Delete, Modify, Sort, Save and Exit respectively.
3. Insert the number corresponding to the choice you wanted to proceed from the main menu.
   - 3.1. Query: You can search for an employee by entering his last name. If there exist one or more employees having a last name similar to that you entered previously, the found employee(s) will be printed with complete information. If there aren't any employees, this message will appear (No employees found!).
   - 3.2. Add: You can insert a new employee in the list but you must make sure manually that this employee doesn't present in the previous list before adding. In case you made that error you will be able to correct it using the Delete function. If the process completed correctly, you will read the message (Employee added successfully.) and the data of the added employee will be printed in front of you.
   - 3.3. Delete: You can remove any employee by entering the employee's first and last names and the system will print the employee(s) which have similar first and last name. You will be asked to enter 1 to confirm the deleting process or press 0 to cancel the process and return back to the main menu. If there aren't any employees having similar data, you will read a message (No employees found!).
   - 3.4. Modify: You can edit the data of any employee by entering the id corresponding to that employee. If an employee is found, you will be asked to press 1 to modify or 0 to skip, for each field of information of the employee as illustrated previously. If no employee was found having the entered id, you will get a message (No employees found!).
   - 3.5. Sort: You can arrange the current list of employees according to one of these fields: 1. Last Name 2. Date of Birth 3. Salary. You can choose one of the previous options by entering the number corresponding to that selection. For arrangement according to last name, the employees are sorted in an alphabetical order of their last name. In case of sorting by date of birth, the employees are sorted descendingly with respect to their age. In case of salary sorting, the employees with larger salary are inserted at the top.
   - 3.6. Save: You can re-write your file with the modified list of employees by choosing this option. You will get a message (Data saved successfully.) if the file is edited correctly. You will have the option to return to the main menu by pressing 0 or exiting directly by pressing 1.

3.7. Exit: You can terminate your session through this function. You will be asked to press 1 to save and exit directly or press 0 for other choices which are to press 1 to exit without saving or to press 0 to return back to main menu.

Instructions

Before running the code of the system make sure that you have inserted a file containing the records of the employees in this form “ID,Last Name,First Name,Salary,Date of Birth as (Day-Month-Year),Address,Phone Number,E-mail” of type text (.txt) in the folder containing the C code where the code will run using the Code Blocks application. Also, make sure that the file is available for edit - not read only file -.

While entering the data of the employees while adding or modifying the employee’s record take care of the following:

- Filename must end with (.txt) extension.
- ID must be numerical data only.
- Last and First Names must contain alphabetical characters only as well as the dash character ‘-‘.
- Salary must be numerical data with at most one decimal point with maximum total number of 9 characters.
- Date of Birth is numerical entry only whose years must belong to the interval [1963, 2003].
- Phone Number must consist of 11 digits where the first 2 digits must be ‘0’ and ‘1’.
- E-mail address must contain 1 at character ‘@’ not more or less. It must also contain a dot character ‘.’ at the forth or third character from the end.

At any point a choice is made between two choices, entering any value other than ‘0’ will considered to be true and will perform the task assigned to the ‘1’ selection.

Algorithms

Searching Algorithm

Sequential Search

This method (algorithm) of searching can be called as Linear Search. In this method, every element is checked until the element meeting the search target is found. In this way, we can find the elements in one comparison in best case scenarios or in n comparisons in worst case scenarios; where n is the number of elements in array or list. The following figure (Fig.) shows an example for the flowchart of a linear search:

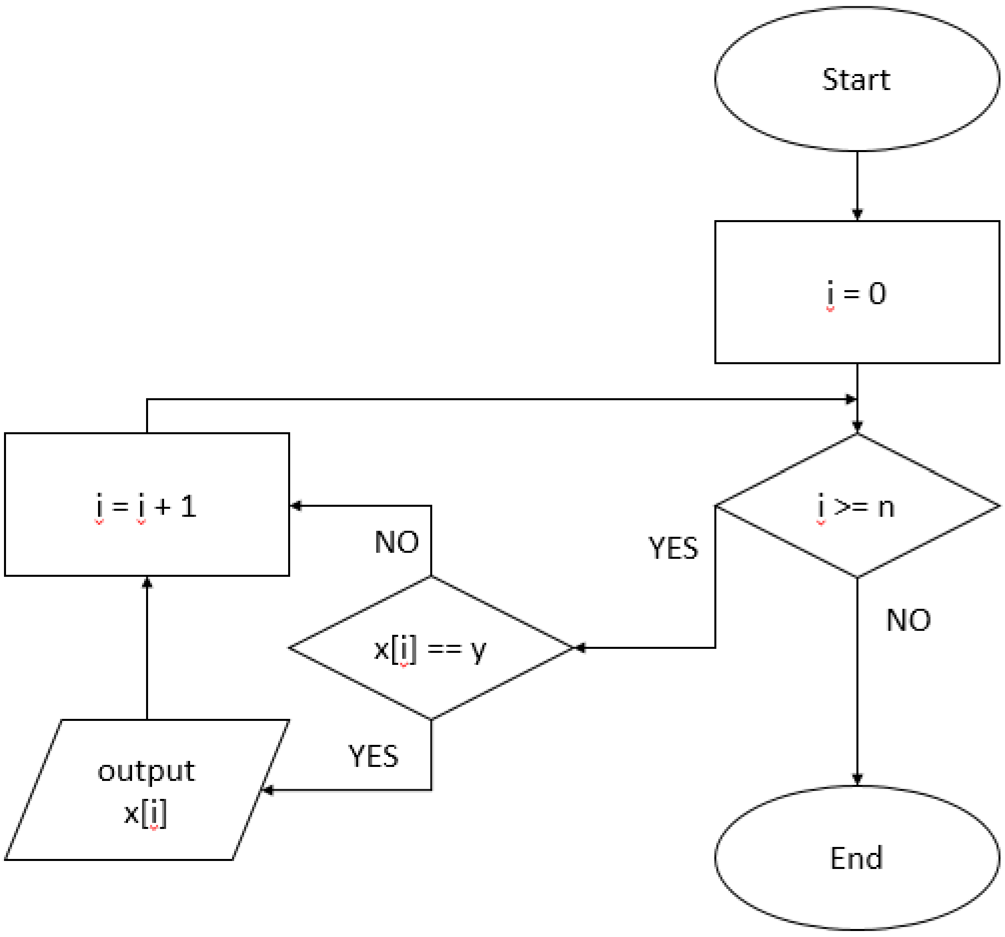


Fig. 2: Example of linear search for finding element = y in list x of n elements

Sorting Algorithm

Bubble Sort

The distance and direction in which the items should move when sorting determines the performance of bubble sorting because the items move in different directions at different speeds. An item that needs to move towards the end of the list can move quickly because it can participate in subsequent exchanges. For example, the biggest item on the list will win each trade, so it will move to its sorted position on the first pass, even if it starts near the start. On the other hand, an item that needs to move to the top of the list cannot move faster than a step by step, so the items move to the start very slowly. If the smallest item is at the end of the list, it will take n-1,steps to move it up.

Bubble sort is the simplest sorting algorithm that swaps adjacent elements multiple times if out of order. Example:

First pass:

(5 1 4 2 8) -> (1 5 4 2 8). Here the algorithm compares and swaps the first two elements because 5> 1.

(1 5 4 2 8) -> (1 4 5 2 8), Swap start 5> 4

(1 4 5 2 8) -> (1 4 2 5 8), start swap 5> 2

(1 4 2 5 8) -> (1 4 2 5 8), now that these are already sorted (8 > 5), the algorithm does do. Do not exchange.

Second pass:

(1 4 2 5 8) -> (1 4 2 5 8)

(1 4 2 5 8) -> (1 2 4 5 8), 4> 2

(1 2 4 5 8) -> (1 2 4 5 8)

(1 2 4 5 8) -> (1 2 4 5 8)

Now the array is already sorted, but the algorithm doesn't know if it's done. It takes one full pass with no substitutions to know that the algorithm is sorted.

Third pass:

(1 2 4 5 8) -> (1 2 4 5 8)

(1 2 4 5 8) -> (1 2 4 5 8)

(1 2 4 5 8) -> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

So on sorting employees based on their age the older employee (smaller number) sits the top while younger is down the list (bigger number) which is ascending order same with salary the bigger the salary the higher the spot on the list which is descending order and this algorithm deals with comparing the names as if the letters have values where the smallest is (a or A) as we used built in function (strcasecmp) and the greatest is (z or Z) which is ascending alphabetical order.