Database of Hospital

Documentation

Name: Khongmeng Kormoua
Neptun Code: I3MLPQ

Basic of Programming 2

Final project

INTRODUCTION:
A database, mainly store a huge amount of information that to be used later when needed. Database is used in every organizations, to store any kind of information. Since in this era, Data has been a very important aspect for development like AI, marketing, research, etc. This project has demonstrate how a simple database works and this documentation contains an explaination, description in detail and this documentation as a data could be referred and used later by other programmer as an idea and instruction for their further development.

This program can save a list of Doctor, Nurse and Patient, which user can input those information, E.g. name, age, etc. Via a menu user interface and these information will be store in a database. Later when it is needed, we can get these information back again, and when some information is not necessary, it can be deleted from a database.

PROGRAM OVERVIEW:

- Database of hospital is implemented base on Object Oriented Programming.
- In this program, I added several techniques for instance: exception handling, file management, dynamic memory management, inheritance, polymorphism, operator overloading, etc.
- This program consists of several files working together as bellow:
    1. **main.cpp:** which works as switch function, switch from this functionality to others for instance: call function to load database, call function to add a new doctor into a database, save this database in permanent place which in this case is a file.
    2. **menu.h:** a header file which contains all main function declarations for instance: add doctor into a database, remove doctor from a database, show a list of doctor, etc.
    3. **menu.cpp:** a source file which contains an implementation of all functions at menu.h
    4. **person.h:** a header file for class Person which will base class for other class like class Doctor, class Nurse, class Patient. This header file will contain all declaration of this class.
    5. **person.cpp:** a source file which contains all implementation of function, method that is declared in person.h
    6. **doctor.h:** a header file for class Doctor which contains all declaration of function, method and operator overloading of this function.
    7. **doctor.cpp:** a source file which has all implementation of all functions, methods, operator overloading from doctor.h
    8. **nurse.h:** a header file for class Nurse which contains all declaration of function, method and operator overloading of this function.
    9. **nurse.cpp:** a source file which has all implementation of all functions, methods and operator overloading from nurse.h
    10. **patient.h:** a header file for class Patient which has all declaration of function, method and operator overloading of this function.
    11. **patient.cpp:** a source file which has all implementation of all functions, methods and operator overloading from patient.h
    12. **ListOfDoctor.txt:** is used to save and load a list of doctor permanently
    13. **ListOfNurse.txt:** is used to save and load a list of nurse permanently
    14. **ListOfPatient.txt:** is used to save and load a list of patient permanently
    15. **HowToUse.txt:** contain some instruction how to used this program
    16. **Credit.txt:** contain some credits of implementor of this program

IMPLEMENTATION:

- class Person
  - this class has 8 private attributes:
    1. **name** (type string), store a name of this person.
    2. **age** (type int), store an age of this person.
    3. **ID** (type string), store an ID of this person.
    4. **address** (type string), store an address of this person.
    5. **gender** (type string), store a gender of this person.
    6. **contactNumber** (type string), store a phone number of this person.
    7. **dateOfBirth** (type string), store a date of birth of this person.
    8. **dateOfMoveIn** (type string), this attribute will be used in 2 different cases. If a person is Doctor or Nurse, this attribute will indicate a date of starting work. If a person is Patient, this attribute will indicate a date of move in to hospital.
  - This class has a default constructor to set an instance of this class to initial state. For string, set to empty string. For int, set to 0.
  - This class has 8 methods to set a value to each attribute and this class has 8 methods to get a value of every attribute:

    1. **setName**
    2. **setAge**
    3. **setID**
    4. **setAddress**
    5. **setGender**
    6. **setContact**
    7. **setDateOfBirth**
    8. **setDateOfMoveIn**
    9. **getName**
    10. **getAge**
    11. **getID**
    12. **getAddress**
    13. **getGender**
    14. **getContact**
    15. **getDateOfBirth**
    16. **getDateOfMoveIn**

  - this class also contain a virtual function of the derived class (Doctor, Nurse)
  - this class has 2 operator overloading:
    1. **operator<<** which print all information of this person to user.
    2. **operator>>** which let user enter an information into a person.

- class Doctor
    - this class is inherited from class Person
    - this class has 4 more private attributes:
        1. **salary** (type string), store an amount of salary of this doctor.
        2. **specialized** (type string), store what specialized is this doctor.
        3. **office** (type string), store an office of this doctor.
        4. **email** (type string), store an email of this doctor.
    - This class has a default constructor to set an instance of this class to initial state. First call a default constructor of base class Person and for other attributes in this class, set them to an empty string.
    - This class has 4 methods to set a value to each attribute and this class has 4 methods to get a value of every attribute:

        1. **setSalary**
        2. **setSpecialized**
        3. **setOffice**
        4. **setEmail**
        5. **getSalary**
        6. **getSpecialized**
        7. **getOffice**
        8. **getEmail**

    - This class has a function **print** which will print a value of those 4 attributes to screen.
    - This class has a function **getInfo** which will get an information from input and set it to all 4 attributes
    - This class has a function **save** which will save all information of this doctor including all attributes from class Person into a file in a correct format.
    - This class has a function **load** which will read information from a file that is formatted correctly and set it as a value of each attributes respectively.
    - This class has an operator overloading for **operator==** which compare this doctor with a given doctor as a parameter and return true or false. Regarding comparison, every attribute of 2 doctors will be compared respectively, if the program found that there are some attributes that differ from each other, then return false. If not, return true.

- class Nurse
    - this class is inherited from class Person
    - this class has 2 more attributes:
        1. **salary** (type string), store an amount of salary of this nurse.
        2. **email** (type string), store an email of this nurse.
    - This class has a default constructor to set an instance of this class to initial state. First call a default constructor of base class Person and for other attributes in this class, set them to an empty string.
    - This class has 2 methods to set a value to each attribute and this class has 2 methods to get a value of every attribute:
        1. **setSalary**
        2. **setEmail**
        3. **getSalary**
        4. **getEmail**

    - This class has a function **print** which will print a value of those 2 attributes to screen.
    - This class has a function **getInfo** which will get an information from input and set it to 2 attributes
    - This class has a function **save** which will save all information of this nurse including all attributes from class Person into a file in a correct format.
    - This class has a function **load** which will read information from a file that is formatted correctly and set it as a value of each attributes respectively.
    - This class has an operator overloading for **operator==** which compare this nurse with a given nurse as a parameter and return true or false. Regarding comparison, every attribute of 2 nurses will be compared respectively, if the program found that there are some attributes that differ from each other, then return false. If not, return true.

- class Patient
    - this class is inherited from class Person
    - this class has 2 more attributes:
        1. **symptoms** (type string), store a symptom of patient.
        2. **room** (type string), store a room of that patient.
    - This class has a default constructor to set an instance of this class to initial state. First call a default constructor of base class Person and for other attributes in this class, set them to an empty string.
    - This class has 2 methods to set a value to each attribute and this class has 2 methods to get a value of every attribute:

        1. **setSymtoms**
        2. **setRoom**
        3. **getSymtoms**
        4. **getRoom**

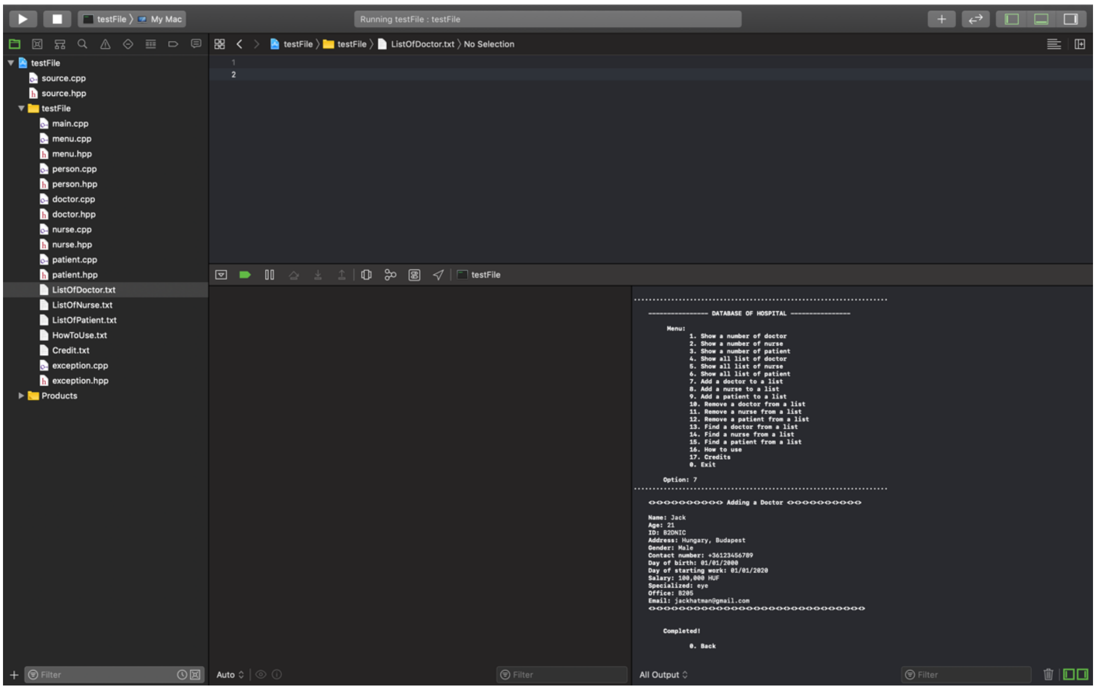
    - This class has a function **print** which will print a value of those 2 attributes to screen.
    - This class has a function **getInfo** which will get an information from input and set it to 2 attributes
    - This class has a function **save** which will save all information of this patient including all attributes from class Person into a file in a correct format.
    - This class has a function **load** which will read information from a file that is formatted correctly and set it as a value of each attributes respectively.
    - This class has an operator overloading for **operator==** which compare this patient with a given patient as a parameter and return true or false. Regarding comparison, every attribute of 2 patients will be compared respectively, if the program found that there are some attributes that differ from each other, then return false. If not, return true.

- A global function in menu.h
    - There are totally 16 global functions, but mainly we can say that there are only 10 functions since some functions have the save implementation but the different is which class they are working with like (Doctor, Nurse, Patient)
        1. **loadDoctorList**, **loadNurseList**, **loadPatientList**: these functions open a file, read a list of Doctor, Nurse, Patient from a file and save it to a container in a program and close a file.
        2. **option**: this function print a menu on a screen and take a choice from a user and return this choice back.
        3. **numberOfPeople**: this function print number of people in a given list into a screen and wait for user to continue.
        4. **printList**: this function will traverse through the given list and print all information of each person in a list to screen, hold an output and wait for user to continue.
        5. **addDoctorToList**, **addNurseToList**, **addPatientToList**: these function first create a new Doctor, Nurse, Patient and get an information from user to every attribute of a instance that program just created, then compare if this new Person has already existed in a list, if not then add this new Person to a given list.
        6. **removeFromList**: this function first print a list for a user and let user enter a name of person to be removed, then remove a person with a given name from user from a list.
        7. **findFromList**: this function let user to enter a name that user is looking for and search for this name from a given list, after found it print all information of that person for user.
        8. **HowToUse**: this function open a file HowToUse.txt and print all text in that function for user.
        9. **Credits**: this function open a file Credit.txt and print all text in that function to user.
        10. **saveDoctorList**, **saveNurseList**, **savePatientList**: these functions first open a file and write every information of each person in a given list to a file in a correct format, then close a file.

- Exception handling
    - We there is an exception handling list which was designed to handle some potential error:
        1. **CannotOpenDoctorList:** this exception will be throw if a file which store a list of doctor cannot be loaded.
        2. **CannotOpenNurseList:** this exception will be throw if a file which store a list of nurse cannot be loaded.
        3. **CannotOpenPatientList:** this exception will be throw if a file which store a list of patient cannot be loaded.
        4. **CannotOpenHowToUseFile:** this exception will be throw if a file which store an information about how to use this program cannot be loaded.
        5. **CannotOpenCreditFile:** this exception will be throw if a file which store an information about a credits cannot be loaded.
        6. **CannotCloseDoctorList:** this exception will be throw if a file which store a list of doctor cannot be closed
        7. **CannotCloseNurseList:** this exception will be throw if a file which store a list of nurse cannot be closed
        8. **CannotClosePatientList:** this exception will be throw if a file which store a list of patient cannot be closed
        9. **CannotCloseHowToUseFile**: this exception will be throw if a file HowToUse cannot be closed.
        10. **CannotCloseCreditFile**: this exception will be throw if a file Credit cannot be closed.

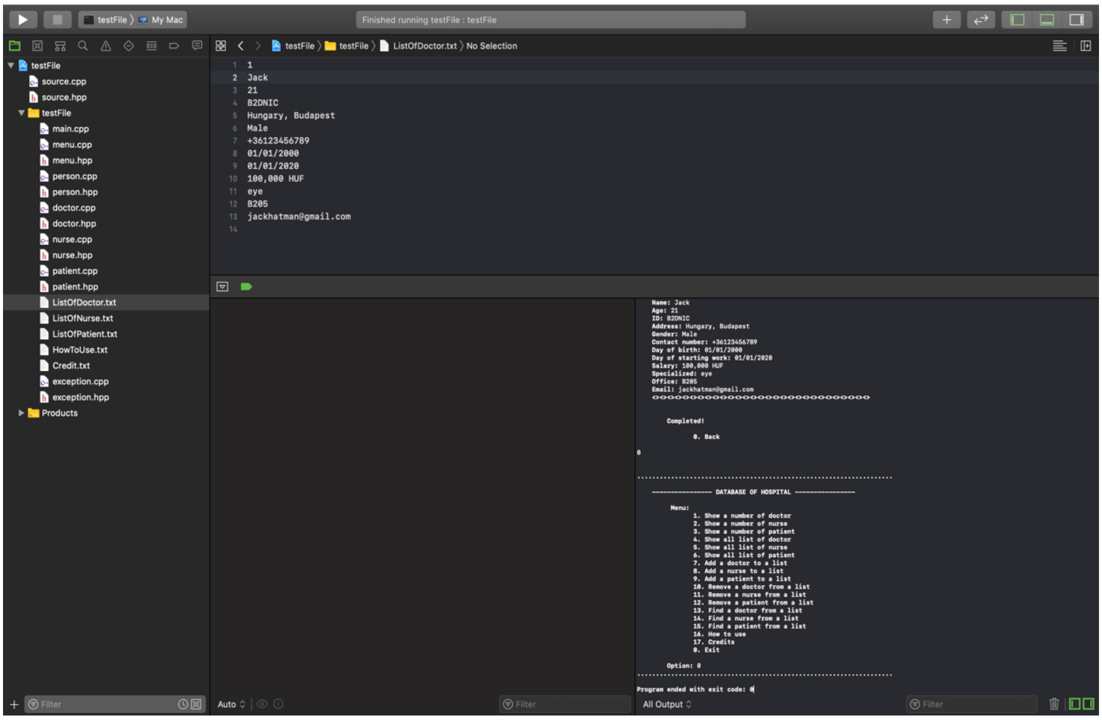
TESTING:

At first test, a list is empty then program read a person's information from user and add these persons into a list and save this list. Then this list is saved into a file as you can see in a figure below.



```
............................................................................

    --------------- DATABASE OF HOSPITAL ---------------

        Menu:
              1. Show a number of doctor
              2. Show a number of nurse
              3. Show a number of patient
              4. Show all list of doctor
              5. Show all list of nurse
              6. Show all list of patient
              7. Add a doctor to a list
              8. Add a nurse to a list
              9. Add a patient to a list
              10. Remove a doctor from a list
              11. Remove a nurse from a list
              12. Remove a patient from a list
              13. Find a doctor from a list
              14. Find a nurse from a list
              15. Find a patient from a list
              16. How to use
              17. Credits
              0. Exit

        Option: 7
............................................................................

    <><><><><><><><><><> Adding a Doctor <><><><><><><><><><>

    Name: Jack
    Age: 21
    ID: B2DNIC
    Address: Hungary, Budapest
    Gender: Male
    Contact number: +36123456789
    Day of birth: 01/01/2000
    Day of starting work: 01/01/2020
    Salary: 100,000 HUF
    Specialized: eye
    Office: B205
    Email: jackhatman@gmail.com
    <><><><><><><><><><><><><><><><><><><><><><><><><><><><>


        Completed!

              0. Back
```

Before save into a file, a file is empty (a upper right box) and an information of a doctor is enter at bottom right box

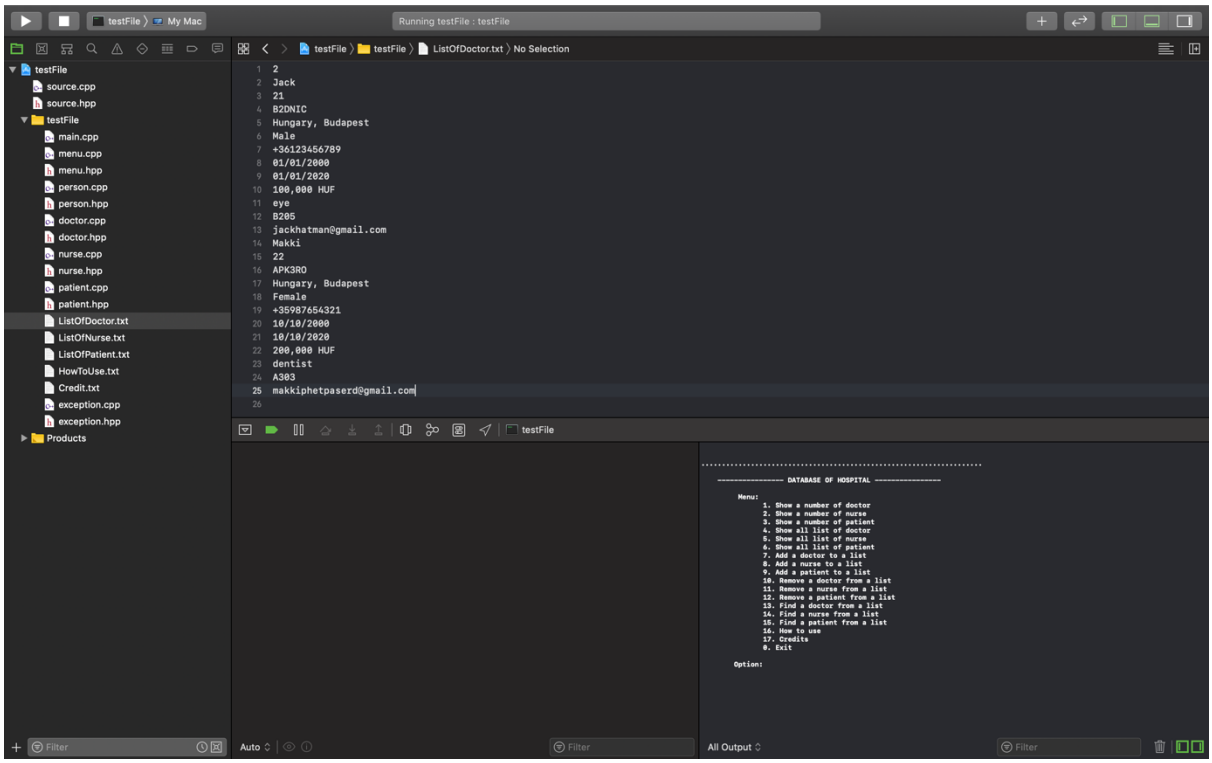


```
1
Jack
21
B2DNIC
Hungary, Budapest
Male
+36123456789
01/01/2000
01/01/2020
100,000 HUF
eye
B205
jackhatman@gmail.com
```

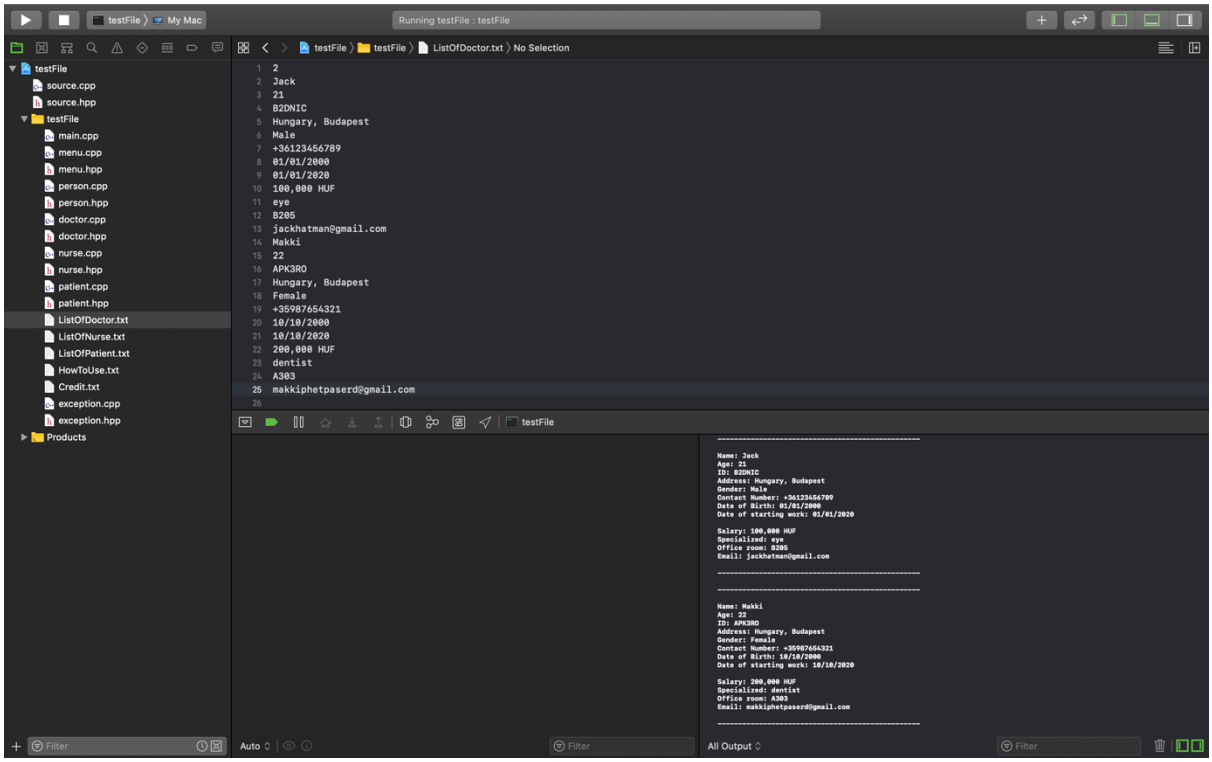
```
    Name: Jack
    Age: 21
    ID: B2DNIC
    Address: Hungary, Budapest
    Gender: Male
    Contact number: +36123456789
    Day of birth: 01/01/2000
    Day of starting work: 01/01/2020
    Salary: 100,000 HUF
    Specialized: eye
    Office: B205
    Email: jackhatman@gmail.com
    <><><><><><><><><><><><><><><><><><><><><><><><><><><><>


        Completed!

              0. Back

0


............................................................................

    --------------- DATABASE OF HOSPITAL ---------------

        Menu:
              1. Show a number of doctor
              2. Show a number of nurse
              3. Show a number of patient
              4. Show all list of doctor
              5. Show all list of nurse
              6. Show all list of patient
              7. Add a doctor to a list
              8. Add a nurse to a list
              9. Add a patient to a list
              10. Remove a doctor from a list
              11. Remove a nurse from a list
              12. Remove a patient from a list
              13. Find a doctor from a list
              14. Find a nurse from a list
              15. Find a patient from a list
              16. How to use
              17. Credits
              0. Exit

        Option: 0
............................................................................
Program ended with exit code: 0
```

After save a list, we can see an information of a doctor that we just gave to a program has been save into a file

In second test, a program will read an existed list from a file and print it to user.

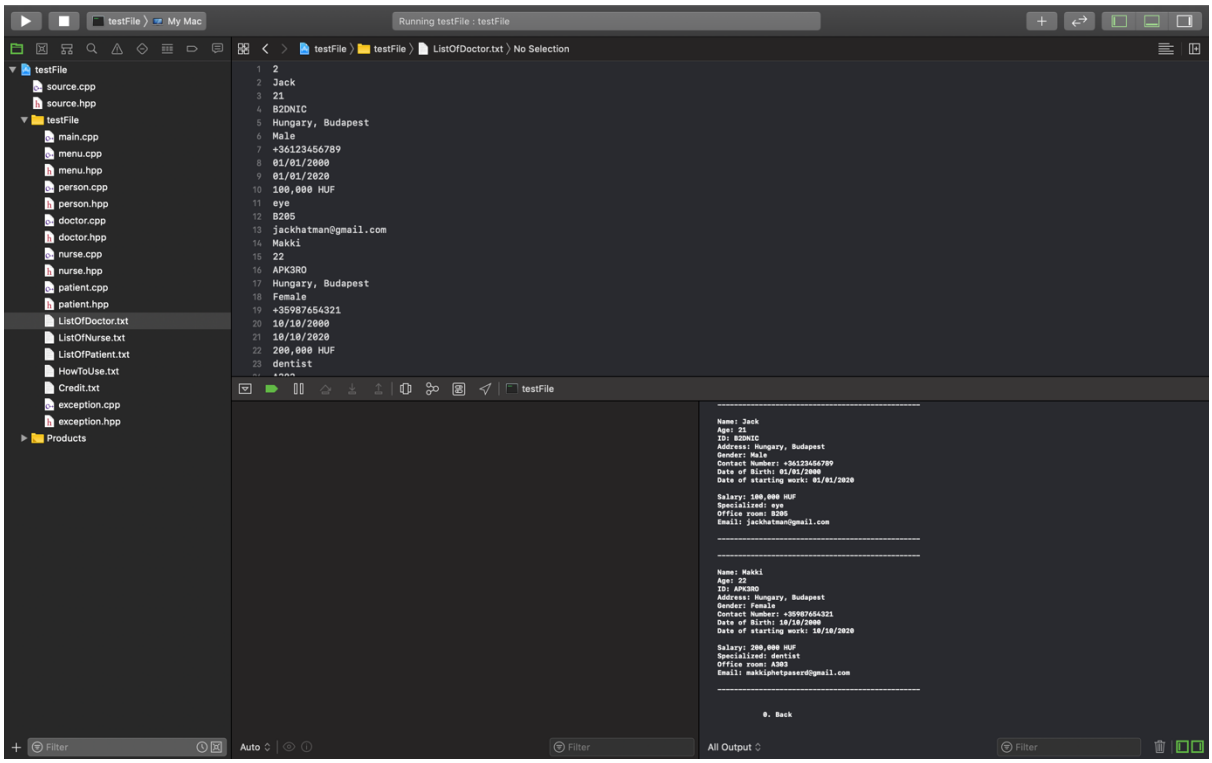


A file has an information of 2 persons, the program will load these information into a container in our program and we can manipulate it. In this case, we will print it.

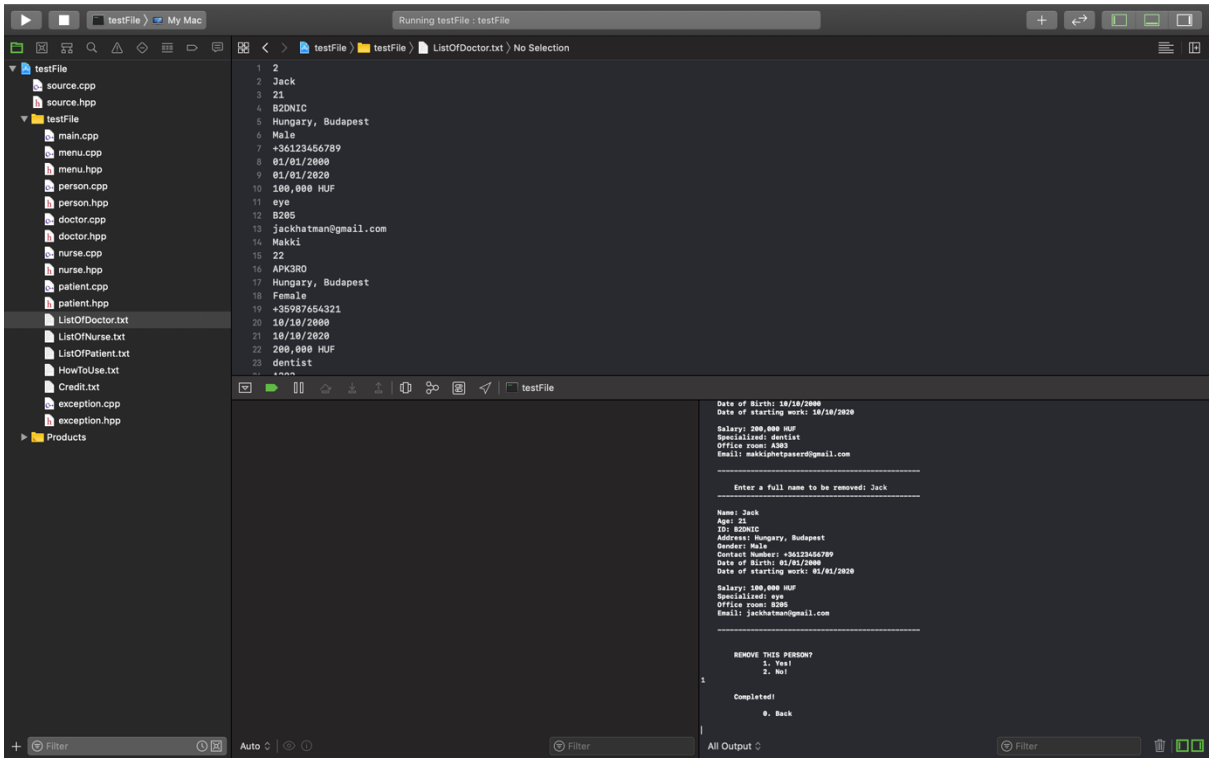


After we selected a menu to print a list we can see a result on the bottom right.

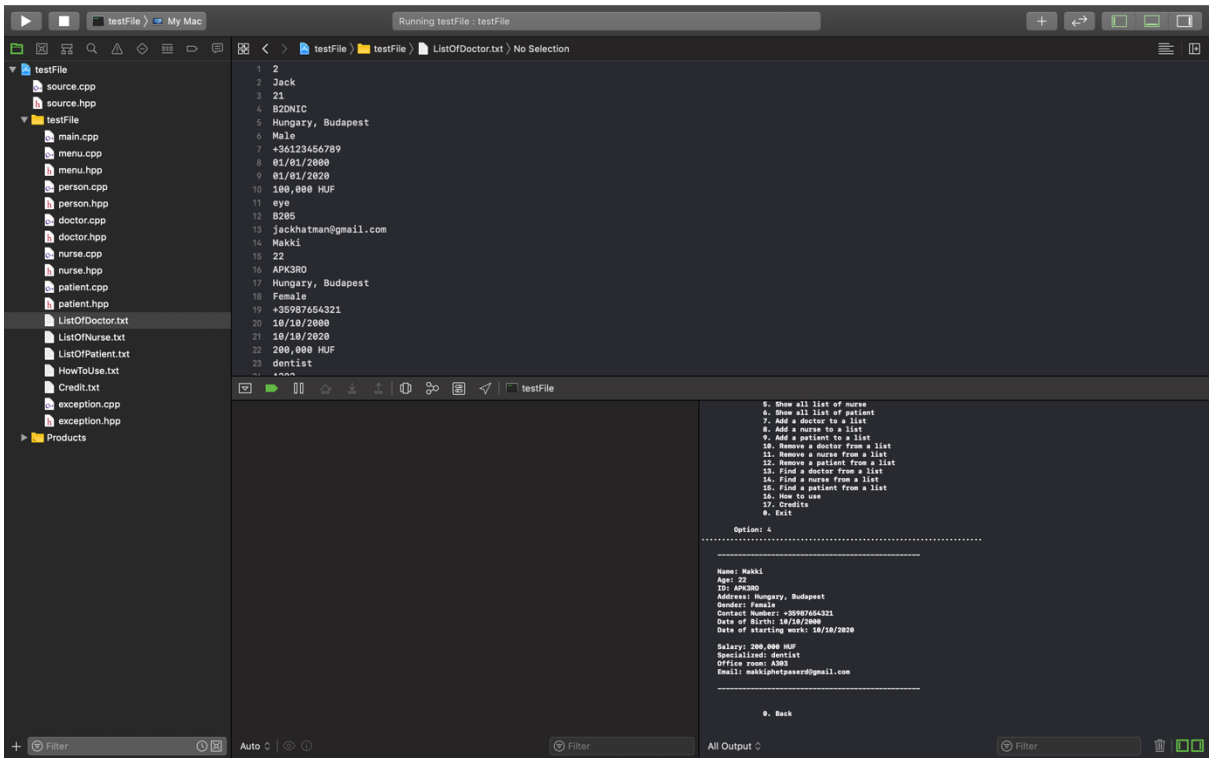
In third test, our program will remove one person from a list and save this list instead of the old list.



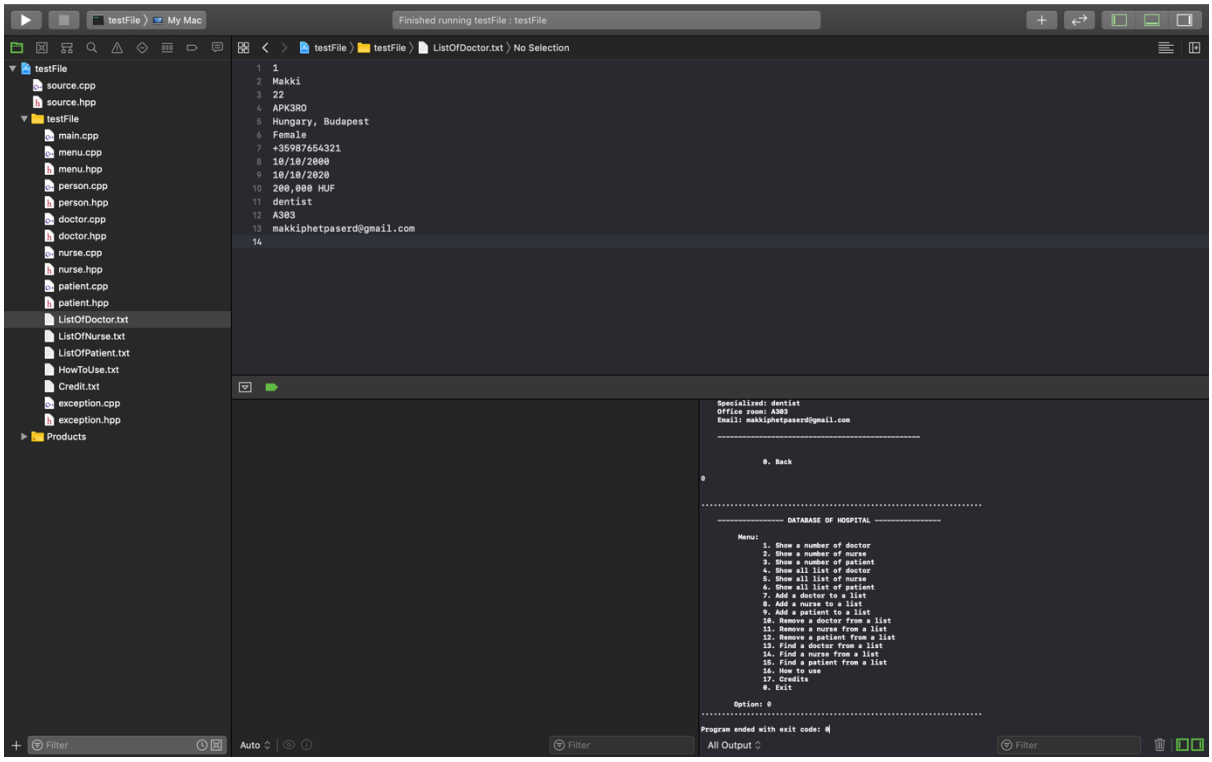
A list before we remove a person, we can see both 2 person on a file (upper right box) and in the output (bottom right box)



Then we will remove 1 person name Jack from a list as in the bottom right box



Before close a program and save a file we can check in a program that there is only 1 person left who is not Jack



Then after we save a list into a file, we can see that in a file (upper right box) has only 1 person left who is Makki

SUMMARY:
Base on a lot of case tests that have been done on background, all features of a program work fine without compiler error and runtime error. File handling works as expected. The program works as specified.