|  |  |
| --- | --- |
| PROJECT  NAME | **ADDRESS BOOK MANAGEMENT SYSTEM DESIGN AND IMPLEMENTATION** |
| CONTENTS | Try designing and implementing an address book management system that can manage the users’ information. You can use C/Java/ C++ programming language to perform this task. Select a kind of appropriate data structure to represent the user information, e.g., if you choose C language as the developing tool, the structure is a wise choice. And for Java, declare a class to stand for the user information. It is also important to think deeply about the collection which is used to store the user information. Array, list, and dynamic array are commonly used in C. And in Java, you can create objects from the Array, List, Set and Map class to store the information items. For permanent storage, you should better write the things in a file. |
| REQUIREMENTS | Basic Requirements:  The system should include the following functionalities:  1. Create a new information card and insert it into the address book.  2. Show details of all information cards stored in the address book.  3. Search an information item by using the specified name.  4. Update an item according to your input.  5. Delete a card item specified by the name. |
| **Advanced Requirements:**  1. Write the data in a file, which means not just in the primary memory.  2. Search for the information item according to different keywords, such as name, gender, etc.  3. Provide a friendly graphic user interface (GUI).  4. Verify the data input. For example, 12345 is not a valid email address.  5. Add features to the system whatever you want. |

|  |  |
| --- | --- |
| **EXPERIMENT REPORT** | ADDRESS BOOK MANAGEMENT SYSTEM DESIGN AND IMPLEMENTATION |
| **1** | **Demand Analysis** |
| * If we design and implement an address book management system that can manage the user’s information, the basic information for a user would be name, phone number, email address, age, and gender. * In order to perform this task, C++ Programming Language is a wise choice for me. Because it also supports Object-Oriented Programming (OOP). * It would be better if a class called “Person” were created and all of a user’s information stored in that class. * Arrays, Vectors (ArrayLists), or Maps can be used to store all person objects, but in this project, Vectors will be ok. * For permanent storage, binary files will be a wise choice. * Multifile Compilation should be used because it helps to keep things organized. What I mean by “Multifile Compilation” is putting my codes in separate files like headers and implementation files. Separating some codes which have the same logic will be very useful for code organizing and code re-use. * As an IDE, “RADStudio-1042 C++ Builder” can be a good choice. It supports cross-platform application development, but in this project compiling the code for “Windows x64” system will be enough. | |
| **2** | **Design Outline** |
| **COMPILATION PROCESS**  **RUNNING PROCESS** | |
| **3** | **Detailed Design** |
| In this section, I’ve talked about how I designed and managed this project in detail.  As an IDE, I have used Embarcadero RADStudio-1042 “C++ Builder Community Edition”.  In the person.h file I put all the declaration of class Person, and declaration of its properties and behaviors.  Definition of class Person and its properties are included in the People.cpp file. If we make the application wider, then it would be better if we create one more class called “Card” to work more specifically. But in this case, it is not so necessary. So, I just created the class “Person” to manage this project. In this class, I declared all the variables as **private**, and created some **setter** and **getter methods** to access and work with the properties of this class. Additionally, most of the validation process has been done inside of class Person. I created some **checker methods** for it. If one of the credentials is incorrect an exception will be thrown, and in the main application, I used try-catch blocks to print the exception if there is any. Moreover, I created the toString() method and overloaded **operator<<** to print this class.  **After reviewing the requirements, I decided to make the project in two versions: Console Application and GUI Application.**  Both the **GUI Application** and **Console Application** included the same class “Person” with the same header file.  There are 6 functionalities in this system. I’ve just created user-defined functions for each of them in the “Console-Application -> AddressBookManagementSystem.console.cpp” and “GUI-Application -> AddressBook.cpp”.  The information of a user is stored in the Person objects. In the memory, all the Person objects are stored in a std::vector called “people”. In the permanent storage, all of the Person objects are stored in the file called “abms.bin”. Moreover, all the information is stored as binary (ios::binary) in the file. For security issues, using binary files is a better choice.  Both the GUI and Console application uses the same file to store information. It has some benefits to keep track of all information while using either of the application.  This was quite an exciting project, if I had more time and knowledge I would add more features to it.  I hope I haven’t missed anything to mention here. In case you can reach out to me if you have any questions about this project. I am so eager to talk about everything in thing project. | |
| **4** | **Test Results** |
| **ADDRESS BOOK MANAGEMENT SYSTEM GUI APPLICATION**  **Testing Basic Requirements**   1. Creating a new information card and inserting it into the address book 2. Showing details of all information cards stored in the address book. 3. Searching an information card by using the specified name. 4. Updating an item by using the specified name. 5. Deleting a card item according to your input.   **Testing Advanced Requirements**   1. **Writing the data in a file, which means not just in the primary memory.**  * Here I’ve closed the application, then re-opened it. All card information is still remaining. It shows some files have been used for permanent storage.      1. **Searching for the information item according to different keywords, such as name, gender, etc.** 2. **Provide a friendly graphic user interface (GUI).** 3. **Verify the data input. For example, 12345 is not a valid email address.** 4. **Add features to the system whatever you want.**  * Additionally, I added showing status notifications, using “Selection List Boxes”, and made the project in two versions (GUI and CONSOLE).     **ADDRESS BOOK MANAGEMENT SYSTEM CONSOLE APPLICATION**  **Testing Basic Requirements**   1. **Creating a new information card and inserting it into the address book.** 2. **Show details of all information cards stored in the address book.**      1. **Search an information item by using the specified name.** 2. **Update an item according to your input.** 3. **Delete a card item specified by name.**     **Testing Advanced Requirements**   1. **Write the data in a file, which means not just in the primary memory.**  * Here I’ve closed the application, then re-opened it. All card information is still remaining. It shows some files have been used for permanent storage.      1. **Search for the information item according to different keywords, such as name, gender, etc.**      1. **Provide a friendly graphic user interface (GUI).**  * I provided a friendly graphic user interface (GUI) as shown above.  1. **Verify the data input. For example, 12345 is not a valid email address.**      1. **Add features to the system whatever you want.** | |
| **5** | **Source Code** |
| **Person.h**  #ifndef PERSON\_H  #define PERSON\_H  #include <iostream>  #include <cstring>  #include <string>  #include <regex>  #include <exception>  #include <sstream>  class Person{  char name[50];  char tel[30];  char email[100];  int age;  char gender[6];  friend std::ostream& operator<<(std::ostream&, Person&);  public:  //Constructors  Person();  ~Person();  //Checkers  static bool isNumber(const std::string&);  static bool isValidPhoneNumber(const std::string&);  static bool isValidEmail(const std::string&);  static bool isEmpty(const std::string&);  //Setters  void setName(const std::string&);  void setTel(const std::string&);  void setEmail(const std::string&);  void setAge(const std::string&);  void setGender(const std::string&);  //Getters  std::string getName();  std::string getTel();  std::string getEmail();  int getAge();  std::string getGender();  std::string toString();  };  #endif  **Person.cpp**  #ifndef PERSON  #define PERSON  #include "Person.h"  //Constructors  Person::Person(){};  Person::~Person(){};  //Checkers  bool Person::isNumber(const std::string &s){  std::string expr = "[0-9]+";  const std::regex regexRule(expr);  return regex\_match(s, regexRule);  }  bool Person::isValidPhoneNumber(const std::string &s){  std::string expr = "^(\\+\\d{1,3}( )?)?((\\(\\d{1,3}\\))|”\\d{1,3})[- .]?\\d{3,4}[- .]?\\d{4}$";  const std::regex regexRule(expr);  return regex\_match(s, regexRule);  }  bool Person::isValidEmail(const std::string &s){  std::string expr = "^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";  const std::regex regexRule(expr);  return regex\_match(s, regexRule);  }  bool Person::isEmpty(const std::string &s){  if(s == "")  return true;  for(int i=0; i<s.length(); ++i){  if(s[i] != ' ')  return false;  }  return true;  }  //Setters  void Person::setName(const std::string &name){  if(isEmpty(name))  throw std::exception("Please enter a name first!");  if(name.length() > 100)  throw std::exception("Length of name should be less than 100!");  std::strcpy(this->name, name.c\_str());  }  void Person::setTel(const std::string &tel){  if(!isValidPhoneNumber(tel) || tel.length() > 30)  throw std::exception("Please enter a valid phone number!");    std::strcpy(this->tel, tel.c\_str());  }  void Person::setEmail(const std::string &email){  if(!isValidEmail(email) || email.length() > 100)  throw std::exception("Please enter a valid email!");    std::strcpy(this->email, email.c\_str());  }  void Person::setAge(const std::string &s\_age){  if(!isNumber(s\_age))  throw std::exception("This field requires number. Please try again!");    int age = std::stoi(s\_age);  if(age < 1)  throw std::exception("Age should be greater than 0. Please try again!");    this->age = age;  }  void Person::setGender(const std::string &gender){  if(gender != "male" && gender != "female")  throw std::exception("Enter 'male' or 'female' only!");  std::strcpy(this->gender, gender.c\_str());  }  //Getters  std::string Person::getName(){  return this->name;  }  std::string Person::getTel(){  return this->tel;  }  std::string Person::getEmail(){  return this->email;  }  int Person::getAge(){  return this->age;  }  std::string Person::getGender(){  return this->gender;  }  std::string Person::toString(){  std::stringstream s;  s << "Name: " << name << "\nTel: " << tel << "\nEmail: "  << email << "\nAge: " << age << "\nGender: " << gender;  return s.str();  }  //Overload  std::ostream& operator<<(std::ostream &output, Person &person){  output << "Name: " << person.name << "\nTel: " << person.tel << "\nEmail: "  << person.email << "\nAge: " << person.age << "\nGender: " << person.gender;  return output;  }  #endif  **AddressBookManagementSystem.Console.cpp**  #include "Person.h"  #include "Person.cpp"  #include <fstream>  #include <map>  void printAtCenter(std::string str, char ch, int size) {  size = size - str.length();  for(int i=0; i<size; ++i) {  if(i == size/2)  std::cout<<str;  std::cout<<ch;  }  std::cout<<"\n";  }  void showMenu(){  std::cout<<"\n";  printAtCenter("\*", '\*', 50);  printAtCenter("\*", '\*', 50);  std::cout<<"Address Book Management System"<<std::endl;  printAtCenter("=", '=', 50);  std::cout<<"\n1 Insert New Card Item\n"  <<"2 Show All Card Items\n"  <<"3 Search A Card Item\n"  <<"4 Modify A Card Item\n"  <<"5 Delete A Card Item\n";  printAtCenter("-", '-', 40);  std::cout<<"\n6 Exit"<<std::endl;  printAtCenter("=", '=', 50);  }  int getIndexOf(const std::string &name, std::vector<Person> &people){  for(int i=0; i<people.size(); ++i){  if(people[i].getName() == name)  return i;  }  return -1;  }  void insertTo(std::vector<Person> &people, std::string &filename){  std::cout<<"\n";  printAtCenter("INSERTION", '-', 41);  std::cout<<"Please provide these informations "  <<"to insert a new Card!"<<std::endl;  try{  Person person;  std::string input;  std::cout<<"Name: ";  std::cin>>input;  if(getIndexOf(input, people) > -1)  throw std::exception("This user already exists!");  person.setName(input);  std::cout<<"Tel: ";  std::cin>>input;  person.setTel(input);  std::cout<<"Email: ";  std::cin>>input;  person.setEmail(input);  std::cout<<"Age: ";  std::cin>>input;  person.setAge(input);  std::cout<<"Gender: ";  std::cin>>input;  person.setGender(input);  //Adding new person to vector in the memory  people.push\_back(person);  //Adding new person to file in the storage  ofstream file;  file.open(filename, ios::binary | ios::out | ios::app);  if (!file) {  std::cout<<"Error in opening file! "  <<"All data will be only stored in the memory.";  }else{  file.write((char\*)&person, sizeof(person));  file.close();  }  std::cout<<"Insertion Successful!"<<std::endl;  }catch(std::exception &e){  std::cout<<e.what()<<std::endl;  }  }  void update(std::vector<Person> &people, std::string &filename){  std::cout<<"\n";  printAtCenter("MODIFICATION", '-', 38);    std::string name;  std::cout<<"Enter the name of card to modify: ";  std::cin>>name;  int index = getIndexOf(name, people);  if(index == -1){  std::cout<<"There isn't any card named "  <<name<<std::endl;  return;  }  auto& person = people[index];  std::cout<<"Enter " << person.getName()  << "'s new informations!"<<std::endl;  try{  std::string input;  std::cout<<"Tel: ";  std::cin>>input;  person.setTel(input);  std::cout<<"Email: ";  std::cin>>input;  person.setEmail(input);  std::cout<<"Age: ";  std::cin>>input;  person.setAge(input);  std::cout<<"Gender: ";  std::cin>>input;  person.setGender(input);  //Updating in storage file  fstream original, temp;  original.open(filename, ios::in | ios::binary);  if(!original){  std::cout<<"\nCouldn't open file. It hasn't been deleted from file!"<<std::endl;  }else{  temp.open("temp.bin", ios::out | ios::app | ios::binary);  if(!temp){  std::cout<<"\nCouldn't create new file. It hasn't been deleted from file!"<<std::endl;  }else{  bool isFound = false;  Person p;  while (original.read((char\*)&p, sizeof(p))) {  if(p.getName() == person.getName()){  temp.write((char\*)&person, sizeof(person));  isFound = true;  }else{  temp.write((char\*)&p, sizeof(p));  }  }  if(isFound == false){  temp.write((char\*)&person, sizeof(person));  }  original.close();  temp.close();  std::remove(filename.c\_str());  std::rename("temp.bin", filename.c\_str());  }  }  std::cout<<"Modification successful!"<<std::endl;  }catch(std::exception &e){  std::cout<<e.what()<<std::endl;  }  }  void deleteFrom(std::vector<Person> &people, std::string &filename){  std::cout<<"\n";  printAtCenter("DELETION", '-', 42);    std::string name;  std::cout<<"Enter the name of card to delete: ";  std::cin>>name;  auto iterator = people.end();  for (auto i = people.begin(); i != people.end(); ++i){  if((\*i).getName() == name){  iterator = i;  }  }  if(iterator == people.end()){  std::cout<<"There isn't any card named "  <<name<<std::endl;  return;  }  Person person = \*iterator;  people.erase(iterator);  //Deleting from file  fstream original, temp;  original.open(filename, ios::in | ios::binary);  if(!original){  std::cout<<"\nCouldn't open file. It hasn't been deleted from file!"<<std::endl;  }else{  temp.open("temp.bin", ios::out | ios::app | ios::binary);  if(!temp){  std::cout<<"\nCouldn't create new file. It hasn't been deleted from file!"<<std::endl;  }else{  Person p;  while (original.read((char\*)&p, sizeof(p))) {  if(p.getName() != person.getName())  temp.write((char\*)&p, sizeof(p));  }  original.close();  temp.close();  std::remove(filename.c\_str());  std::rename("temp.bin", filename.c\_str());  }  }  std::cout<<name<<"'s deleted successfully!"<<std::endl;  }  //As long as, I store objects in both memory and storage,  //there is no need to pass filename and read from the file  //for this method  void showAll(std::vector<Person> &people){  std::cout<<"\n";  printAtCenter("ALL CARD ITEMS", '-', 40);  std::cout<<"\nThere are " << people.size()  << " cards in total.\n"<<std::endl;  for(int i=0; i<people.size(); ++i)  std::cout<<"#CARD"<<i+1<<"\n"<<people[i]<<"\n\n";  }  //As long as, I store objects in both memory and storage,  //there is no need to pass filename and read from the file  //for this method  void search(std::vector<Person> &people){  std::cout<<"\n";  printAtCenter("SEARCH", '-', 46);  std::string search;  std::cout<<"Enter any information about cards for searching: ";  std::cin>>search;  int numberOfMatches = 0;  std::map<int, Person> matches;  for(int i=0; i<people.size(); ++i){  if(people[i].getName() == search || people[i].getTel() == search ||  people[i].getEmail() == search || std::to\_string(people[i].getAge()) == search ||  people[i].getGender() == search){  matches.insert({i+1, people[i]});  numberOfMatches++;  }  }  std::cout<<"\nThere are " << numberOfMatches  << " cards match this information.\n\n";  for(auto& [index, person] : matches)  std::cout<<"#CARD"<<index<<"\n"<<person<<"\n\n";  }  int main(){  std::cout<<"Welcome to ADDRESS BOOK MANAGEMENT SYSTEM!"<<std::endl;  bool firstTime = true;  //Actually, if I use a file to store the informations,  //there is no need for storing all person objects in the memory  //but, as it is asked in the requirements of the experiment, so,  //I stored them in the BOTH MEMORY(via vectors) and STORAGE(via file)  std::vector<Person> people;  std::string filename = "abms.bin";  //If not exist create the file  ofstream f;  f.open(filename, ios::binary | ios::out | ios::app);  f.close();  //Reading all informations from storage (from <filename> file)  ifstream file;  file.open(filename, ios::binary | ios::out | ios::in);  if (!file) {  std::cout<<"Error in opening file! "  <<"All data will be only stored in the memory.";  }else{  if(file){  Person person;  while(file.read((char\*)&person, sizeof(person))) {  people.push\_back(person);  }  }  }  //SYSTEM DISPLAY  while(true) {  std::string input;  if(firstTime == false){  std::cout<<"\nDo you want to continue? (y/n): ";  std::cin>>input;  if(input == "n" || input == "N") {  break;  }else if(input != "y" && input != "Y"){  std::cout<<"Only 'y' or 'n' is allowed,"  <<" please try again!"<<std::endl;  continue;  }  }    firstTime = false;  showMenu();  std::cout<<"Please select the operation serial number: ";  std::cin>>input;  if(!Person::isNumber(input)) {  std::cout<<"\nOnly numbers are allowed,"  <<" please try again!"<<std::endl;  continue;  }    int option = std::stoi(input);  if(option == 1) {  insertTo(people, filename);  }else if(option == 2) {  showAll(people);  }else if(option == 3) {  search(people);  }else if(option == 4) {  update(people, filename);  }else if(option == 5){  deleteFrom(people, filename);  }else if(option == 6) {  std::cout<<"Exit"<<std::endl;  break;  }else {  std::cout<<"The number you entered is not correct,"  <<" please try again!"<<std::endl;  continue;  }  }  std::cout<<"\nThanks for using. See you next time :)"  <<std::endl;  return 0;  }  **These are the source code files of the GUI Application:**  **Note: GUI Application also uses “People.h” and “People.cpp”.**  **AddressBookManagementSystem.cpp**  //---------------------------------------------------------------------------  #include <fmx.h>  #ifdef \_WIN32  #include <tchar.h>  #endif  #pragma hdrstop  #include <System.StartUpCopy.hpp>  //---------------------------------------------------------------------------  USEFORM("AddressBook.cpp", AddressBookForm);  //---------------------------------------------------------------------------  extern "C" int FMXmain()  {  try  {  Application->Initialize();  Application->CreateForm(\_\_classid(TAddressBookForm), &AddressBookForm);  Application->Run();  }  catch (Exception &exception)  {  Application->ShowException(&exception);  }  catch (...)  {  try  {  throw Exception("");  }  catch (Exception &exception)  {  Application->ShowException(&exception);  }  }  return 0;  }  //---------------------------------------------------------------------------  **AddressBook.h**  //---------------------------------------------------------------------------  #ifndef AddressBookH  #define AddressBookH  //---------------------------------------------------------------------------  #include <System.Classes.hpp>  #include <FMX.Controls.hpp>  #include <FMX.Forms.hpp>  #include <FMX.Gestures.hpp>  #include <FMX.StdCtrls.hpp>  #include <FMX.TabControl.hpp>  #include <FMX.Types.hpp>  #include <FMX.Controls.Presentation.hpp>  #include <FMX.Colors.hpp>  #include <FMX.Edit.hpp>  #include <FMX.ListBox.hpp>  #include <FMX.ComboEdit.hpp>  #include <FMX.Header.hpp>  #include "Person.h"  #include "Person.cpp"  #include <FMX.Layouts.hpp>  //---------------------------------------------------------------------------  class TAddressBookForm : public TForm  {  \_\_published: // IDE-managed Components  TTabControl \*Z;  TTabItem \*InsertTab;  TTabItem \*ShowAllTab;  TTabItem \*SearchTab;  TTabItem \*ModifyTab;  TTabItem \*DeleteTab;  TComboBox \*GenderListBox;  TEdit \*NameField;  TListBoxItem \*male;  TListBoxItem \*female;  TButton \*InsertButton;  TLabel \*Header;  TEdit \*PhoneNumberField;  TEdit \*EmailField;  TComboBox \*AgeListBox;  TListBoxItem \*SelectAge;  TLabel \*NameLabel;  TLabel \*PhoneNumberLabel;  TLabel \*EmailLabel;  TLabel \*AgeLabel;  TLabel \*GenderLabel;  TLabel \*StatusLabel;  TListBoxItem \*SelectGender;  TLabel \*CardsLabel;  TLabel \*ShowAllHeader;  TListBox \*Cards;  TEdit \*SearchField;  TLabel \*Label1;  TLabel \*Label3;  TListBox \*MatchingCards;  TLabel \*MatchingCardsLabel;  TButton \*SearchButton;  TGroupBox \*ModificationField;  TLabel \*Label2;  TComboBox \*ModificationCardListBox;  TListBoxItem \*ListBoxItem1;  TEdit \*PhoneNumberModificationField;  TEdit \*EmailModificationField;  TLabel \*Label4;  TLabel \*Label5;  TLabel \*Label6;  TComboBox \*AgeModificationList;  TButton \*ModifyButton;  TLabel \*Label7;  TComboBox \*GenderModificationList;  TListBoxItem \*Agelist1;  TListBoxItem \*ListBoxItem2;  TListBoxItem \*ListBoxItem3;  TListBoxItem \*ListBoxItem4;  TLabel \*ModificationStatusLabel;  TLabel \*Label8;  TComboBox \*DeletionCardListBox;  TListBoxItem \*DeletionCardListItem1;  TButton \*DeleteButton;  TLabel \*DeletionLabel;  void \_\_fastcall FormCreate(TObject \*Sender);  void \_\_fastcall InsertButtonClick(TObject \*Sender);  void \_\_fastcall ShowAllTabClick(TObject \*Sender);  void \_\_fastcall SearchButtonClick(TObject \*Sender);  void \_\_fastcall ModifyTabClick(TObject \*Sender);  void \_\_fastcall DeleteTabClick(TObject \*Sender);  void \_\_fastcall DeleteButtonClick(TObject \*Sender);  void \_\_fastcall ModifyButtonClick(TObject \*Sender);  private: // User declarations  public: // User declarations  \_\_fastcall TAddressBookForm(TComponent\* Owner);  };  //---------------------------------------------------------------------------  extern PACKAGE TAddressBookForm \*AddressBookForm;  //---------------------------------------------------------------------------  #endif  **AddressBook.cpp**  //---------------------------------------------------------------------------  #include <fmx.h>  #include <System.UITypes.hpp>  #pragma hdrstop  #include "AddressBook.h"  #include <fstream>  #include <map>  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.fmx"  TAddressBookForm \*AddressBookForm;  std::vector<Person> people;  std::string filename = "abms.bin";  int getIndexOf(const std::string &name, std::vector<Person> &people){  for(int i=0; i<people.size(); ++i){  if(people[i].getName() == name)  return i;  }  return -1;  }  //---------------------------------------------------------------------------  \_\_fastcall TAddressBookForm::TAddressBookForm(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::FormCreate(TObject \*Sender)  {  // This defines the default active tab at runtime  //TabControl1->ActiveTab = TabItem1;  for(int i=1; i<=150; ++i){  AgeListBox->Items->Add(i);  AgeModificationList->Items->Add(i);  }  //If file not exists, then create the file  ofstream f;  f.open(filename, ios::binary | ios::out | ios::app);  f.close();  //Reading all informations from storage (from <filename> file)  ifstream file;  file.open(filename, ios::binary | ios::in);  if (file) {  Person person;  while(file.read((char\*)&person, sizeof(person))) {  people.push\_back(person);  }  file.close();  }  std::string str = "There are " + std::to\_string(people.size()) + " cards in total.\n\n";  CardsLabel->Text = str.c\_str();  for(int i=0; i<people.size(); ++i){  std::string s = "#CARD" + std::to\_string(i);  s += "\n" + people[i].toString() + "\n\n";  Cards->Items->Add(s.c\_str());  ModificationCardListBox->Items->Add(people[i].getName().c\_str());  DeletionCardListBox->Items->Add(people[i].getName().c\_str());  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::InsertButtonClick(TObject \*Sender){  try{  Person person;  std::string input = "";  AnsiString text;  text = NameField->Text;  input = text.c\_str();  if(getIndexOf(input, people) > -1)  throw std::exception("This user already exists!");  person.setName(input);  text = PhoneNumberField->Text;  input = text.c\_str();  person.setTel(input);  text = EmailField->Text;  input = text.c\_str();  person.setEmail(input);  text = AgeListBox->Selected->Text;  input = text.c\_str();  if(input == "Age")  throw std::exception("Please select age!");  person.setAge(input);  text = GenderListBox->Selected->Text;  input = text.c\_str();  if(input == "Gender")  throw std::exception("Please select gender!");  person.setGender(input);  //Adding new person to vector in the memory  people.push\_back(person);  //Adding new person to file in the storage  ofstream file;  file.open(filename, ios::binary | ios::out | ios::app);  if (!file) {  StatusLabel->Text = "Error in opening file! All data will be only stored in the memory.";  }else{  file.write((char\*)&person, sizeof(person));  file.close();  }  StatusLabel->Text = "Insertion Successful!";  }catch(std::exception &e){  StatusLabel->Text = e.what();  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::ShowAllTabClick(TObject \*Sender)  {  std::string str = "There are " + std::to\_string(people.size()) + " cards in total.\n\n";  CardsLabel->Text = str.c\_str();  Cards->Items->Clear();  for(int i=0; i<people.size(); ++i){  std::string s = "#CARD" + std::to\_string(i);  s += "\n" + people[i].toString() + "\n\n";  Cards->Items->Add(s.c\_str());  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::SearchButtonClick(TObject \*Sender)  {  AnsiString text = SearchField->Text;  std::string search = text.c\_str();  int numberOfMatches = 0;  std::map<int, Person> matches;  for(int i=0; i<people.size(); ++i){  if(people[i].getName().c\_str() == search || people[i].getTel().c\_str() == search ||  people[i].getEmail().c\_str() == search || std::to\_string(people[i].getAge()).c\_str() == search ||  people[i].getGender().c\_str() == search){  matches.insert({i+1, people[i]});  numberOfMatches++;  }  }  search = "There are " + std::to\_string(numberOfMatches) + " cards match this information.";  MatchingCardsLabel->Text = search.c\_str();  MatchingCards->Items->Clear();  for(auto& [index, person] : matches){  std::string s = "#CARD" + std::to\_string(index);  s += "\n" + person.toString() + "\n\n";  MatchingCards->Items->Add(s.c\_str());  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::ModifyTabClick(TObject \*Sender)  {  ModificationCardListBox->Items->Clear();  ModificationCardListBox->Items->Add(" Select a card to modify");  ModificationCardListBox->ItemIndex = 0;  for(int i=0; i<people.size(); ++i)  ModificationCardListBox->Items->Add(people[i].getName().c\_str());  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::DeleteTabClick(TObject \*Sender)  {  DeletionCardListBox->Items->Clear();  DeletionCardListBox->Items->Add(" Select a card to delete");  DeletionCardListBox->ItemIndex = 0;  for(int i=0; i<people.size(); ++i)  DeletionCardListBox->Items->Add(people[i].getName().c\_str());  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::DeleteButtonClick(TObject \*Sender)  {  AnsiString text = DeletionCardListBox->Selected->Text;  std::string name = text.c\_str();  if(name == " Select a card to delete"){  DeletionLabel->Text = "Please select a card!";  return;  }  auto iterator = people.end();  for (auto i = people.begin(); i != people.end(); ++i){  if((\*i).getName() == name){  iterator = i;  }  }  if(iterator == people.end())  return;  Person person = \*iterator;  people.erase(iterator);  //Deleting from file  fstream original, temp;  original.open(filename, ios::in | ios::binary);  if(!original){  DeletionLabel->Text = "\nCouldn't open file. It hasn't been deleted from file!";  }else{  temp.open("temp.bin", ios::out | ios::app | ios::binary);  if(!temp){  DeletionLabel->Text = "\nCouldn't create new file. It hasn't been deleted from file!";  }else{  Person p;  while (original.read((char\*)&p, sizeof(p))) {  if(p.getName() != person.getName())  temp.write((char\*)&p, sizeof(p));  }  original.close();  temp.close();  std::remove(filename.c\_str());  std::rename("temp.bin", filename.c\_str());  }  }  std::string s = name + "'s deleted successfully!";  DeletionLabel->Text = s.c\_str();  DeletionCardListBox->Items->Clear();  DeletionCardListBox->Items->Add(" Select a card to delete");  DeletionCardListBox->ItemIndex = 0;  for(int i=0; i<people.size(); ++i)  DeletionCardListBox->Items->Add(people[i].getName().c\_str());  }  //---------------------------------------------------------------------------  void \_\_fastcall TAddressBookForm::ModifyButtonClick(TObject \*Sender)  {  try{  std::string input = "";  AnsiString text;  text = ModificationCardListBox->Selected->Text;  text = text.c\_str();  if(text == " Select a card to modify"){  throw std::exception("Please select a card!");  }  auto iterator = people.end();  for (auto i = people.begin(); i != people.end(); ++i){  if((\*i).getName() == text.c\_str()){  iterator = i;  }  }  if(iterator == people.end())  throw std::exception("Couldn't find this card!");  Person person = \*iterator;  text = PhoneNumberModificationField->Text;  input = text.c\_str();  person.setTel(input);  text = EmailModificationField->Text;  input = text.c\_str();  person.setEmail(input);  text = AgeModificationList->Selected->Text;  input = text.c\_str();  if(input == "Age")  throw std::exception("Please select age!");  person.setAge(input);  text = GenderModificationList->Selected->Text;  input = text.c\_str();  if(input == "Gender")  throw std::exception("Please select gender!");  person.setGender(input);  //Updating in storage file  fstream original, temp;  original.open(filename, ios::in | ios::binary);  if(!original){  ModificationStatusLabel->Text = "\nCouldn't open file. It hasn't been deleted from file!";  }else{  temp.open("temp.bin", ios::out | ios::app | ios::binary);  if(!temp){  ModificationStatusLabel->Text = "\nCouldn't create new file. It hasn't been deleted from file!";  }else{  bool isFound = false;  Person p;  while (original.read((char\*)&p, sizeof(p))) {  if(p.getName() == person.getName()){  temp.write((char\*)&person, sizeof(person));  isFound = true;  }else{  temp.write((char\*)&p, sizeof(p));  }  }  if(isFound == false){  temp.write((char\*)&person, sizeof(person));  }  original.close();  temp.close();  std::remove(filename.c\_str());  std::rename("temp.bin", filename.c\_str());  }  }  ModificationStatusLabel->Text = "Modification Successful!";  }catch(std::exception &e){  ModificationStatusLabel->Text = e.what();  }  }  //---------------------------------------------------------------------------  **Inside the Project Folder “Address Book Management System”**  **Inside the release folder** | |
| **6** | **Summary** |
| In this experiment, I learned how to make a real-world project called “Address Book Management System”. This was a great experience to practice my skills in “C++ Programming Language”, OOP, DSA, and GUI Application Development. Some people say you don’t learn much in the class, you learn things practicing in the real life. But I don’t agree with this quote. Because in the class I learned these things in theory and I practiced them while I was managing this project. I’ve got a better understanding of those things now. So, if you study well in the class and do the experiment part, then you will be much better in your gained skills.  In this process, my teacher was also very helpful. I learned a lot in the lab class. Now, I’ve leveled up my skills. Thanks to my teacher.  I am so happy to have managed this project. | |

## Student’s Signature: 阿哥战 Date: May 21, 2022