КОЛОМИЙСЬКИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ

НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Циклова комісія інженерія програмного забезпечення

**КУРСОВА РОБОТА**

з дисципліни:

*"Об`єктно-орієнтоване програмування"*

на тему: "Автоматизоване робоче місце адміністратора авіакомпанії"

Студента 3 курсу групи П-31

галузі знань 12

спеціальності 121

Стоцького А-В. О.

(прізвище та ініціали)

Керівник: Красничук В.В.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Коломия – 2019 рік

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Найменування етапів  курсової роботи | Термін виконання етапів роботи | Примітки |
| 1. | Постановка задач, аналіз специифікації вимог. | 21.01.2019 |  |
| 2. | Визначення структури даних та методів рішення. | 04.02.2019 |  |
| 3. | Визначення структури програмних модулів. | 18.02.2019 |  |
| 4. | Розробка системи класів. | 27.02.2019 |  |
| 5. | Розробка методів. | 04.03.2019 |  |
| 6. | Інтерфейс програми. | 13.03.2019 |  |
| 7. | Розроблення програми. | 20.03.2019 |  |
| 8. | Розробка документації. | 08.04.2019 |  |
| 9. | Тестування. | 10.04.2019 |  |
| 10. | Оформлення пояснювальної записки. | 24.04.2019 |  |
| 11. | Виправлення недоліків. | 13.04.2019 |  |
| 12. | Захист. | 18.06.2019 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Стоцький Арсен-Віктор Олександрович

(підпис студента) (прізвище, ім‘я, по батькові студента)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Красничук Вікторія Вікторівна

(підпис викладача) (прізвище, ім‘я, по батькові викладача)

**Зміст**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

418.КР 121.01.19 ПЗ

Розроб.

Стоцький А-В.О.

Перевір.

Красничук В.В.

Реценз.

Н. Контр.

Затверд.

Пояснювальна записка.

Автоматизоване робоче місце адміністратора авіакомпанії

Літ.

Аркушів

*32*

КПК НУ «ЛП»

Вступ………………………………………………………………………..……...5

1 Аналіз специфікації вимог технічного завдання………….…………………6

2 Об’єктно-орієнтований аналіз досліджуванної проблеми та методів моделювання й програмування.…………………………………………………8

3 Процес розроблення програми………………………………………………..9

3.1 Розробка системи класів………………………………………..………...9

3.2 Розробка методів…………………………………………………….16

3.3 Опис файлів даних та інтерфейсу програми……………………………17

4 Тестування програми і результати її виконання……………………………21

5 Висновки ……………..……………………………………………………….23

6 Перелік використаних джерел ………………………………………………24

7 Додатки ...………………………………………………………..…………….25

**ВСТУП**

В даний час подорож повітряним транспортом все більше набуває популярності. Через те, що існує велика кількість людей, які бажають пересуватися літаком, необхідне створення програмного продукту, який допоможе адміністратору авіакомпанії услідкувати за усім. А саме, який пасажир користується певним рейсом, який персонал обслуговує цей рейс, яким літаком будуть летіти пасажири.

За допомогою даної програми адміністратор завжди зможе подивитися всіх пасажирів та персонал, всі рейси та літаки, а при необхідності змінити, добавити чи видалити записи.

Важливо розуміти, що програма не є професійною. Це лише приклад можливої ​​конфігурації. В різних авіакомпаніях своя специфіка ведення справ, але програма є динамічною для введення змін, тому при необхідності можна добавити новий функціонал.

**1 АНАЛІЗ СПЕЦИФІКАЦІЇ ВИМОГ ТЕХНІЧНОГО ЗАВДАННЯ**

Даний програмний продукт повинен бути розроблений з використанням основний концепцій ООП. Об'єктно-орієнтоване програмування (ООП) - парадигма програмування, в якій основними концепціями є поняття об'єктів і класів. Дана концепція в даний час є провідною в області програмування.

В процесі аналізу предметної області були сформовані основні вимоги щодо програми, були визначено які пункти інтерфейсу потрібно реалізувати, яка платформа розробки буде використовуватися, які бібліотеки необхідні.

Основним завданням є розробка зручного у використанні продукту, який дасть змогу швидко вирішувати поставлені завдання, а саме, адміністрування даних щодо рейсів. Інтерфейс програми буде легким для розуміння та простим у використанні, всі дані можна буде переглянути, змінити та видалити в разі необхідності.

При запуску програми повинне відкриватися головне меню, яке буде містити 4 кнопки: Меню пасажирів, Меню літаків, Меню персоналу, Меню рейсів. Над кнопками, на панелі, яка займає всю форму, буде назва авіакомпанії. Справа від кнопок знаходитиметься логотип.

При натисканні на одну з кнопок буде відкриватися меню, де також зверху буде назва авіакомпанії, а зліва кнопки в кожному з меню:

1. Меню пасажирів
   1. Добавити нового
   2. Переглянути всіх
   3. Видалити (за id та номером паспорта)
   4. Редагувати (всі поля)
   5. Знайти (за id, номером паспорта та іменем)
   6. Зберегти
   7. Назад
2. Меню літаків
   1. Добавити новий
   2. Переглянути всі
   3. Видалити (за id та маркою)
   4. Редагувати (всі поля)
   5. Знайти (за id та маркою)
   6. Зберегти
   7. Назад
3. Меню персоналу
   1. Добавити нового
   2. Переглянути всіх
   3. Видалити (за id та номером паспорта)
   4. Редагувати (всі поля)
   5. Знайти (за id, номером паспорта, посадою, іменем, зміною)
   6. Зберегти
   7. Назад
4. Меню рейсів
   1. Добавити новий
   2. Добавити пасажира
   3. Переглянути всі (також показати кількість куплених місць та виручку)
   4. Видалити/Знайти (видалити за id, пошук за id, назвою рейсу, датою відправлення, пошук пасажирів та персоналу за id рейсу)
   5. Зберегти
   6. Назад

При натисканні на одну з кнопок меню в правій частині буде з’являтися форма з відповідними полями та даними.

# 2 Об’єктно-орієнтований аналіз досліджуваної проблеми та методів моделювання й програмування

Розробка прогарми відбувалася на мові програмування С++ в інтегрованому середовищі розробки Qt Creator з використанням бібліотек Qt.

В даній програмі використовуються такі поняття як клас, графічний інтерфейс користувача, інкапсуляція, наслідування, поліморфізм, сигнально-слотові з’єднання в Qt.

Клас – це спеціальна конструкція, яка використовується для групування пов'язаних змінних та [функцій](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0). При цьому, згідно з термінологією [ООП](https://uk.wikipedia.org/wiki/%D0%9E%D0%9E%D0%9F), глобальні змінні класу (члени-змінні) називаються  полями даних , а члени-функції називають [методами](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) класу. Створений та ініціалізований екземпляр класу називають об'єктом класу. На основі одного класу можна створити безліч об'єктів, що відрізнятимуться один від одного своїм  станом (значеннями полів).

Графічний інтерфейс користувача (GUI, Graphical user interface) — тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

Інкапсуляція – об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів ([методів](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))). Доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи), що дозволяє знизити [зв'язність](https://uk.wikipedia.org/wiki/%D0%97%D0%B2%27%D1%8F%D0%B7%D0%BD%D1%96%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Таким чином контролюються звернення до полів класів та їхня правильна ініціалізація, усуваються можливі помилки пов'язані з неправильним викликом методу.

Наслідування – механізм утворення нових  [класів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))  на основі використання вже існуючих. При цьому властивості та функціональність батьківського класу переходять до класу нащадка (дочірнього).

Поліморфізм – властивість, яка дозволяє одне і те саме ім’я використовувати для вирішення декількох технічно різних задач, Тобто це перевантаження методів, типу коли ми в класі маєм декілька методів з одною назвою, але приймають вони різні параметри.

Сигнально-слотові з’єднання. Будь-яка об'єктно-орієнтована програма складається з об'єктів, які взаємодіють між собою. Кожен з об'єктів володіє станом, який визначає сукупність даних, які зберігає об'єкт у даний момент. У відповідь на взаємодію з об'єктом, його стан може змінитися. Наприклад, об'єкт який реалізує мережеве з'єднання може отримати нові переслані дані, а об'єкт який реалізує кнопку на вікні користувацького інтерфейсу, може бути натиснутий користувачем. Таким чином об'єкт змінив свій стан — і він може повідомити про це інший об'єкт надсилаючи йому повідомлення про зміну

**3 ПРОЦЕС РОЗРОБЛЕННЯ ПРОГРАМИ**

**3.1 Розробка системи класів**

Програма містить 4 основні класи: Passanger, Personal, Airplane, Reis. Класи Passanger та Personal наслідують клас People, в якому організовано роботу з інформаціює про людину, тобто її ідентифікаційний номер, ім’я, прізвище, дата народження. Кожен з інших класів містить поля, які властиві тільки йому, та геттери і сеттери.

Але ці 4 класи містять інформацію тільки про один об’єкт, тому були реалізовані такі класи-меню як MenuPassanger, MenuPersonal, MenuAirplane, MenuReis, де в контейнері класу QList міститься список відповідних об’єктів (список пасажирів, список співробітників, список літаків, список рейсів). В цих класах реалізовані методи пошуку, додавання, видалення, редагування даних, зчитування з файлу та запис в файл. До цих класів також створена форма інтерфейсу. Основний клас MainWindow – це головна форма програми, вона містить кнопку доступу до інших форм.

Класи Personal, Passanger, Airplane є незалежними один від одного. Клас Reis співпрацює з цими трьома класами. Коли заповнюється інформація про новий рейс, то вказується номер літака, звідки автоматично вичисляється кількість місць даного рейсу, також вказується номер зміни, і можна буде переглянути який персонал обслуговує цей рейс. Коли рейс створений, то тепер можна добавляти пасажирів.

Діаграми класів:

Клас, який відповідає за зберігання даних про людину:

|  |
| --- |
| **People** |
| int id;  QString name;  QString surname;  QString dateBirth;  QString pasport; |
| +People();  +void setId(int set\_id);  +void setName(QString set\_name);  +void setSurname(QString set\_surname);  +void setDateBirth(QString set\_dateBirth);  +void setPasport(QString set\_pasport);  +int getId();  +QString getName();  +QString getSurname();  +QString getDateBirth();  +QString getPasport(); |
| Клас, що організовує роботу з одним співробітником персоналу:   |  | | --- | | **Personal : public People** | | QString posada;  int zmina;  int zarplata; | | +Personal();  +void setPosada(QString set\_posada);  +void setZmina(int set\_zmina);  +void setZarplata(int set\_zarplata);  +QString getPosada();  +int getZmina();  +int getZarplata(); | |

|  |
| --- |
| Клас, що організовує роботу з одним пасажиром: |
| **Passanger : public People** |
| QString nationality; |
| +Passanger();  +void setNationality(QString set\_nationality);  +QString getNationality(); |

|  |
| --- |
| Клас, що організовує роботу з одним літаком: |
| **Airplane** |
| int id;  QString marka;  int passCapacity;  int vantagCapacity;  int speed; |
| +Airplane();  +void setId(int set\_id);  +void setMarka(QString set\_marka);  +void setPassCapacity(int set\_passCapacity);  +void setVantagCapacity(int set\_vantagCapacity);  +void setSpeed(int set\_speed);  +int getId();  +QString getMarka();  +int getPassCapacity();  +int getVantagCapacity();  +int getSpeed(); |

|  |
| --- |
| Головне меню: |
| **MainWindow** |
| Ui::MainWindow \*ui;  MenuAirplane \*lMenuAirplane;  MenuPassangers \*lMenuPassangers;  MenuPersonal \*lMenuPersonal;  MenuReis \*lMenuReis; |
| +explicit MainWindow(QWidget \*parent = nullptr);  +~MainWindow();  -void slotOpenMenuAirplanes();  -void slotOpenMenuPassangers();  -void slotOpenMenuPersonal();  -void slotOpenMenuReis(); |

|  |
| --- |
| Меню літаків |
| **MenuAirplane** |
| Ui::MenuAirplane \*ui;  QList<Airplane> listAirplane; |
| +explicit MenuAirplane(QWidget \*parent = nullptr);  +~MenuAirplane();  +int getCapacityById(int lId);  +bool checkId(int check\_id);  -void hideAll();  -int maxId();  -void dateInFile();  -void dateFromFile();  -void slotShowAddNew();  -void slotShowShowAll();  -void slotShowDelete();  -void slotShowEdit();  -void slotShowFind();  -void slotGoBack();  -void slotAddNew();  -void slotSave();  -void slotDeleteId();  -void slotDeleteMarka();  -void slotEditMarka();  -void slotEditPassanger();  -void slotEditVantag();  -void slotEditSpeed();  -void slotFindId();  -void slotFindMarka(); |

|  |
| --- |
| Меню пасажирів |
| **MenuPassanger** |
| Ui::MenuPassanger \*ui;  QList<Passanger> listPassanger; |
| +explicit MenuPassangers(QWidget \*parent = nullptr);  +~MenuPassangers();  +bool checkId(int check\_id);  +QString getNameById(int findId);  +QString getSurnameById(int findId);  +QString getDateBirthById(int findId);  +QString getPasportById(int findId);  +QString getNationalityById(int findId);  -void hideAll();  -int maxId();  -void dateInFile();  -void dateFromFile();  -void slotShowAddNew();  -void slotShowShowAll();  -void slotShowDelete();  -void slotShowEdit();  -void slotShowFind();  -void slotGoBack();  -void slotAddNew();  -void slotSave();  -void slotDeleteId();  -void slotDeletePasport();  -void slotEditName();  -void slotEditSurname();  -void slotEditDateBirth();  -void slotEditPasport();  -void slotEditNationality();  -void slotFindId();  -void slotFindPasport();  -void slotFindName(); |

|  |
| --- |
| Меню персоналу |
| **MenuPersonal** |
| Ui::MenuPersonal \*ui;  QList<Personal> listPersonal; |
| +explicit MenuPersonal(QWidget \*parent = nullptr);  +~MenuPersonal();  +bool checkId(int check\_id);  -void hideAll();  -int maxId();  -void dateInFile();  -void dateFromFile();  -void slotShowAddNew();  -void slotShowShowAll();  -void slotShowDelete();  -void slotShowEdit();  -void slotShowFind();  -void slotGoBack();  -void slotAddNew();  -void slotSave();  -void slotDeleteId();  -void slotDeletePasport();  -void slotEditName();  -void slotEditSurname();  -void slotEditDateBirth();  -void slotEditPasport();  -void slotEditPosada();  -void slotEditZmina();  -void slotEditZarplata();  -void slotFindId();  -void slotFindPasport();  -void slotFindZmina();  -void slotFindPosada();  -void slotFindName(); |

|  |
| --- |
| Меню рейсів |
| **MenuReis** |
| Ui::MenuReis \*ui;  QList<Reis> listReis; |
| +explicit MenuReis(QWidget \*parent = nullptr);  +~MenuReis();  -void hideAll();  -int maxId();  -bool checkId(int check\_id);  -void dateFromFile();  -void slotShowAddNew();  -void slotShowAddNewPass();  -void slotShowShowAll();  -void slotShowEdit();  -void slotShowFind();  -void slotGoBack();  -void slotAddNew();  -void slotSave();  -void slotAddNewPass();  -void slotDeleteId();  -void slotFindId();  -void slotFindDepartureDate();  -void slotFindName();  -void slotFindPass();  -void slotFindPersonal();  -void slotEditDepartureDate();  -void slotEditDepartureTime();  -void slotEditArrivalDate();  -void slotEditArrivalTime(); |

**3.2 Розробка методів**

У зв’язку з тим, що робробка програми велася в Qt Creator, то для роботи програми було умісно використовувати бібліотеки Qt.

Наприклад, для того щоб вивести дані на екран, слугує метод **slotShowShowAll(),** реалізований в кожному класі-меню. Дані виводяться у вигляді таблиці, яка створюється за допомогою бібліотеки QStandardItem та об`єкту інтерфейсу QTableView. Реалізацію методу на прикладі класу MenuReis продемонстровано в додатку А.

За добавлення нового запису відповідає метод **slotAddNew(),** також реалізований в кожному класі-меню. Новий запис – це новий об’єкт. Так, наприклад, щоб добавити новий рейс, потрібно створити об’єкт класу Reis, заповнити всі поля, та закинути цей об’єкт в список класу QList. Реалізацію методу на прикладі класу MenuReis продемонстровано в додатку Б.

В кожному класі-меню також реалізовані методи пошуку, редагування та видалення даних за відповідними значеннями. При пошуку дані виводиться у вигляді таблиці, так як і в методі, який виводить всі дані на екран. Приклад реалізації пошуку продемонстровано в додатку В, редагування – в додатку Г, видалення – в додатку Д.

Зчитування та запис в файл також реалізовано за допомогою однієї з Qt бібліотек – QFile. Приклад реалізації запису в файл в додатку Е, зчитування – в додатку Є.

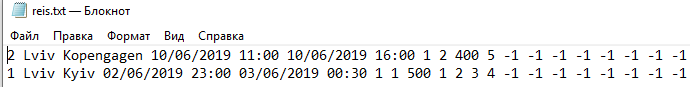
В деяких методах при помилці чи попередженні з’являєтсья вікно класу QMessageBox, який повідомляє що саме було не так.

Код та опис іншиш незначних методів в додатку Ж.

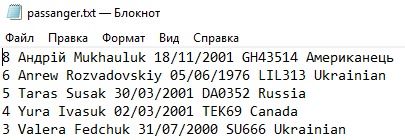
**3.3 Опис файлів даних та інтерфейсу програми**

Інформація щодо рейсів, пасажирів, персоналу, літаків записується в відповідні файли.

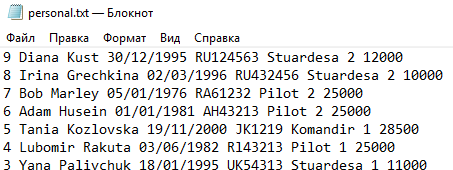
Файл, де зберігається інформація про рейси:



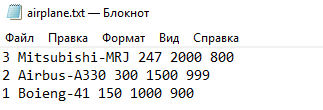
Файл, де зберігається інформація про пасажирів:

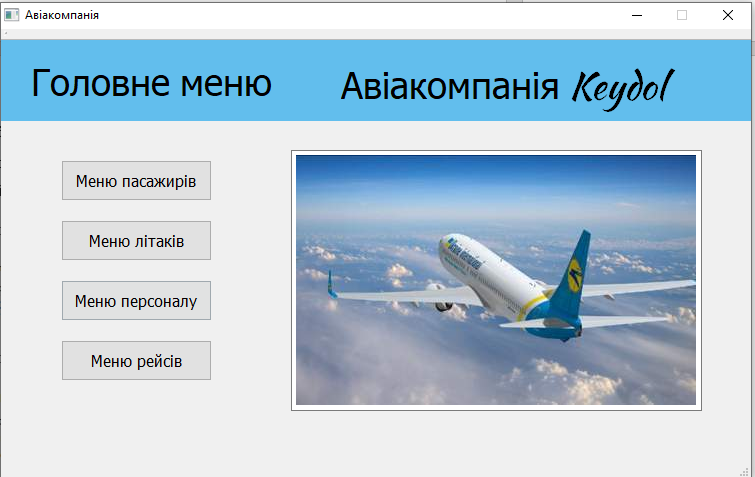


Файл, де зберігається інформація про персонал:

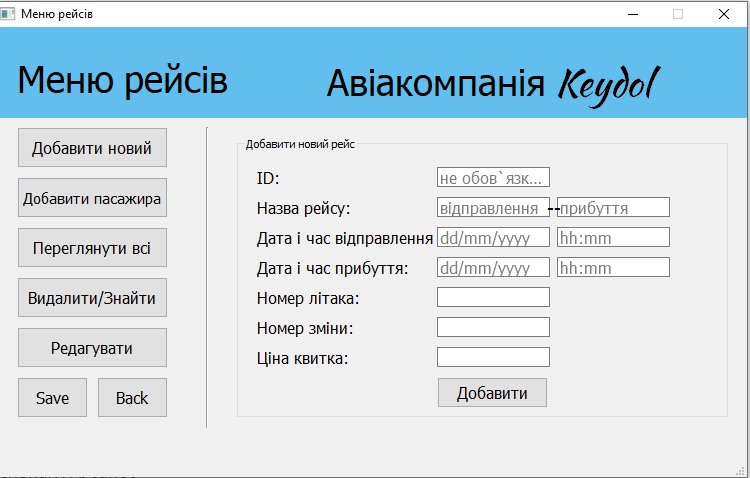


Файл, де зберігається інформація про літаки:

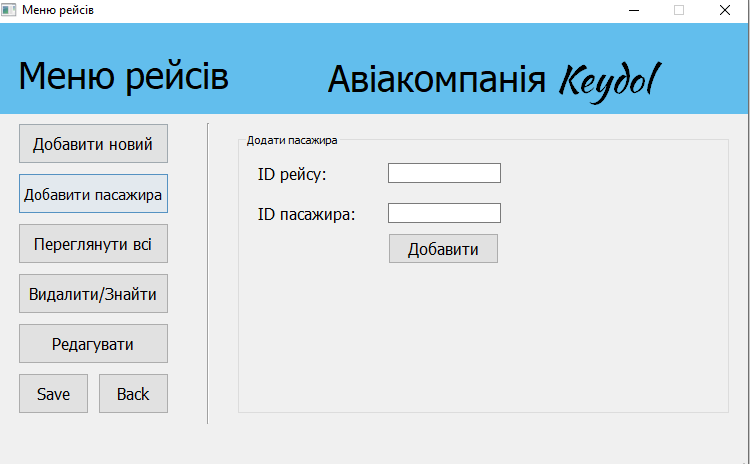


Головна форма

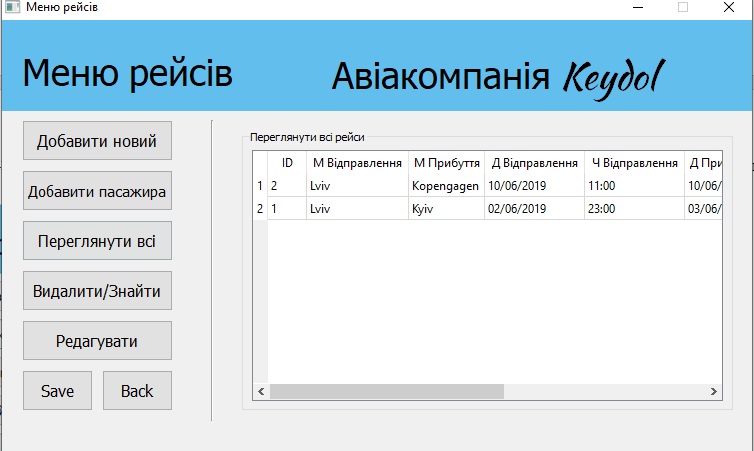
Меню рейсів -> Добавити новий



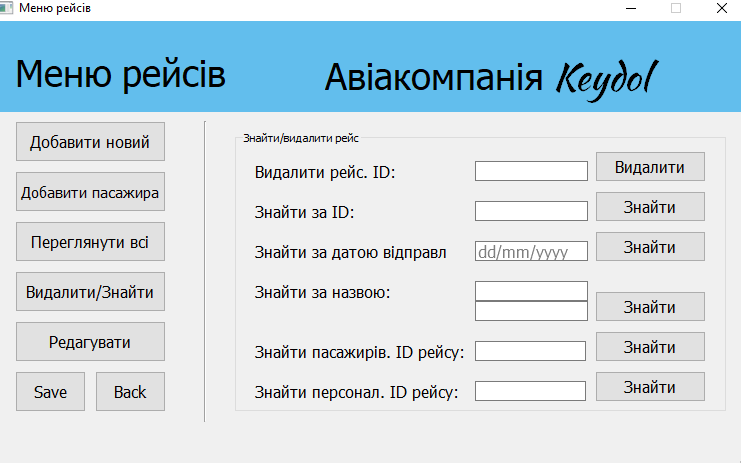
Меню рейсів -> Добавити пасажира



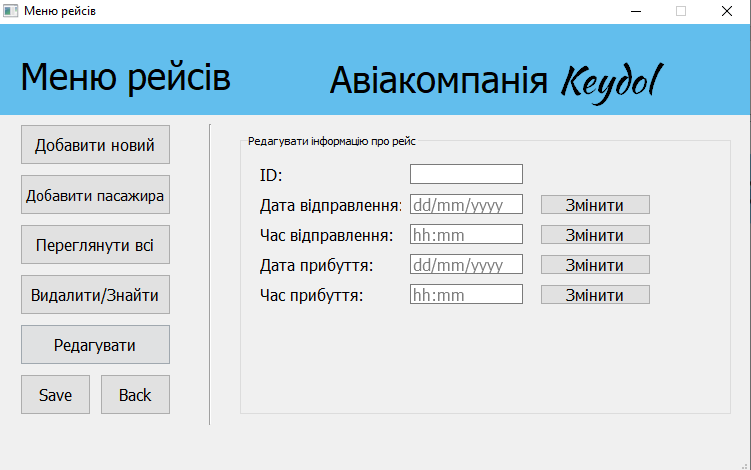
Меню рейсів -> Переглянути всі



Меню рейсів -> Видалити/Знайти



Меню рейсів -> Редагувати

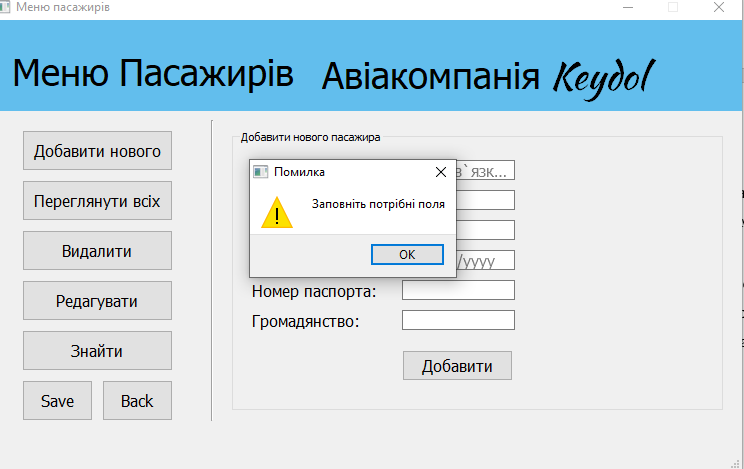


**4 ТЕСТУВАННЯ ПРОГРАМИ**

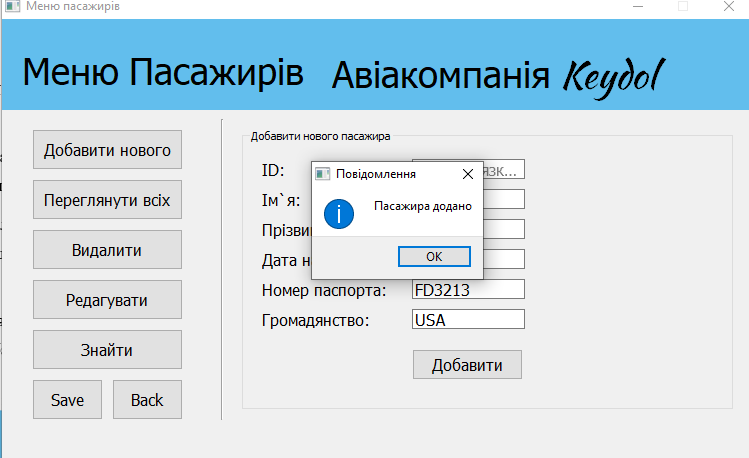
Суть тестування програми полягає в тому, щоб перевірити її на наявні помилки. Якщо такі існують, необхідно їх виправити.

Після кожного етапу розробки проводилося невелике тесутвання, результатом якого було знаходження проблем та подальше їх усунення.

В результаті фінального тестування помилок знайдено не було, все працює правильно, всі кнопки виконують поставлені задачі, всі дані відображаються коректно.

При помилці, яку допускає користувач програми, з'являється відповідне вікно, яке повідомляє йому про це. Наприклад, при добавленні нового пасажира без введення даних, ми бачимо це: ****

А коли все відбулося коректно, програма також повідомляє про це. Такі повідомлення також значно полегшують тестування програми.

****

**ВИСНОВКИ**

Під час виконання роботи були значно вдосконалені навички в об’єктно-орієнтованому прогармуванні. Також навчився працювати з графічним інтерфейсом користувача, створювати, змінювати його. Почав працювати з бібліотеками Qt, які планую використовувати і в подальшому.

Результатом курсової роботи стала готова програма, написана на мові С++ в середовищі Qt Creator, яка дозволяє автоматизувати робоу адміністратора авіакомпанії. Програма дозволяє маніпулювати даними чотирьох сутностей: паспжири, літаки, персонал, рейси. Адміністратор може добавляти, змінювати, видаляти, шукати їх. Хоча програма і не є професійною, але є динамічною та підлягає вдосконаленню

**ЛІТЕРАТУРА:**

1. Бьярне Страуструп: «Программирование принципы и практика использования C++».

2. Герберт Шилдт: «С++ базовый курс».

3. «Программирование на языке С++ в среде Qt Creator» - Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк, О. В. Чеснокова, А. С. Чмыхало.

**ІНФОРМАЦІЙНІ ДЖЕРЕЛА:**

1. http://doc.crossplatform.ru/qt/

2. [www.cppstudio.com](http://www.cppstudio.com)

3. <https://code-live.ru>

**ДОДАТКИ**

**Додаток А**

void MenuReis::slotShowShowAll() {

hideAll();

ui->groupShowAll->show();

QStandardItemModel \*model = new QStandardItemModel;

QStandardItem \*item;

QStringList horizontalHeader;

horizontalHeader.append("ID");

horizontalHeader.append("М Відправлення");

horizontalHeader.append("М Прибуття");

horizontalHeader.append("Д Відправлення");

horizontalHeader.append("Ч Відправлення");

horizontalHeader.append("Д Прибуття");

horizontalHeader.append("Ч Прибуття");

horizontalHeader.append("Ціна квитка");

horizontalHeader.append("Літак");

horizontalHeader.append("Зміна");

horizontalHeader.append("Місця");

horizontalHeader.append("Куплені місця");

horizontalHeader.append("Прибуток");

model->setHorizontalHeaderLabels(horizontalHeader);

int i = 0;

for(QList<Reis>::iterator it = listReis.begin(); it != listReis.end(); \*it++, i++) {

item = new QStandardItem(QString::number(it->getId()));

model->setItem(i, 0, item);

item = new QStandardItem(it->getDepartureCity());

model->setItem(i, 1, item);

item = new QStandardItem(it->getArrivalCity());

model->setItem(i, 2, item);

item = new QStandardItem(it->getDepartureDate());

model->setItem(i, 3, item);

item = new QStandardItem(it->getDepartureTime());

model->setItem(i, 4, item);

item = new QStandardItem(it->getArrivalDate());

model->setItem(i, 5, item);

item = new QStandardItem(it->getArrivalTime());

model->setItem(i, 6, item);

item = new QStandardItem(QString::number(it->getTicketPrice()));

model->setItem(i, 7, item);

item = new QStandardItem(QString::number(it->getAirplaneId()));

model->setItem(i, 8, item);

item = new QStandardItem(QString::number(it->getZmina()));

model->setItem(i, 9, item);

item = new QStandardItem(QString::number(it->getPassCount()));

model->setItem(i, 10, item);

item = new QStandardItem(QString::number(it->getPassCountBuy()));

model->setItem(i, 11, item);

item = new QStandardItem(QString::number(it->getPassCountBuy() \* it->getTicketPrice()));

model->setItem(i, 12, item);

}

ui->tableViewShowAll->setModel(model);

ui->tableViewShowAll->resizeRowsToContents();

ui->tableViewShowAll->resizeColumnsToContents();

}

**Додаток Б**

void MenuReis::slotAddNew() {

Reis reis;

MenuAirplane lMenuAirplane;

if(ui->lineEditAddNewDepartureCity->displayText().isEmpty() ||

ui->lineEditAddNewArrivalCity->displayText().isEmpty() ||

ui->lineEditAddNewDepartureDate->displayText().isEmpty() ||

ui->lineEditAddNewDepartureTime->displayText().isEmpty() ||

ui->lineEditAddNewArrivalDate->displayText().isEmpty() ||

ui->lineEditAddNewArrivalTime->displayText().isEmpty() ||

ui->lineEditAddNewAirplaneId->displayText().isEmpty() ||

ui->lineEditAddNewZmina->displayText().isEmpty() ||

ui->lineEditAddNewTicket->displayText().isEmpty()) {

QMessageBox::warning(this, "Помилка", "Заповніть потрібні поля");

return;

}

if(ui->lineEditAddNewId->displayText().isEmpty()) {

reis.setId(maxId()+1);

} else if(checkId(ui->lineEditAddNewId->displayText().toInt())) {

QMessageBox::warning(this, "Помилка", "Такий id уже використовується");

return;

} else {

reis.setId(ui->lineEditAddNewId->displayText().toInt());

}

if(!lMenuAirplane.checkId(ui->lineEditAddNewAirplaneId->displayText().toInt())) {

QMessageBox::warning(this, "Помилка", "Літак не знайдено");

return;

}

reis.setDepartureCity(ui->lineEditAddNewDepartureCity->displayText());

reis.setArrivalCity(ui->lineEditAddNewArrivalCity->displayText());

reis.setDepartureDate(ui->lineEditAddNewDepartureDate->displayText());

reis.setDepartureTime(ui->lineEditAddNewDepartureTime->displayText());

reis.setArrivalDate(ui->lineEditAddNewArrivalDate->displayText());

reis.setArrivalTime(ui->lineEditAddNewArrivalTime->displayText());

reis.setAirplaneId(ui->lineEditAddNewAirplaneId->displayText().toInt());

reis.setZmina(ui->lineEditAddNewZmina->displayText().toInt());

reis.setTicketPrice(ui->lineEditAddNewTicket->displayText().toInt());

listReis.prepend(reis);

QMessageBox::information(this, "Повідомлення", "Рейс додано");

ui->lineEditAddNewId->setText("");

ui->lineEditAddNewDepartureCity->setText("");

ui->lineEditAddNewArrivalCity->setText("");

ui->lineEditAddNewDepartureDate->setText("");

ui->lineEditAddNewDepartureTime->setText("");

ui->lineEditAddNewArrivalDate->setText("");

ui->lineEditAddNewArrivalTime->setText("");

ui->lineEditAddNewAirplaneId->setText("");

ui->lineEditAddNewZmina->setText("");

ui->lineEditAddNewTicket->setText("");

}

**Додаток В**

void MenuReis::slotFindId() {

ui->groupFind->hide();

ui->groupFinder->show();

QStandardItemModel \*model = new QStandardItemModel;

QStandardItem \*item;

QStringList horizontalHeader;

horizontalHeader.append("ID");

horizontalHeader.append("М Відправлення");

horizontalHeader.append("М Прибуття");

horizontalHeader.append("Д Відправлення");

horizontalHeader.append("Ч Відправлення");

horizontalHeader.append("Д Прибуття");

horizontalHeader.append("Ч Прибуття");

horizontalHeader.append("Ціна квитка");

horizontalHeader.append("Літак");

horizontalHeader.append("Зміна");

horizontalHeader.append("Місця");

horizontalHeader.append("Куплені місця");

horizontalHeader.append("Прибуток");

model->setHorizontalHeaderLabels(horizontalHeader);

int findId = ui->lineEditFindId->displayText().toInt();

for(QList<Reis>::iterator it = listReis.begin(); it != listReis.end(); \*it++) {

if(findId == it->getId()) {

item = new QStandardItem(QString::number(it->getId()));

model->setItem(0, 0, item);

item = new QStandardItem(it->getDepartureCity());

model->setItem(0, 1, item);

item = new QStandardItem(it->getArrivalCity());

model->setItem(0, 2, item);

item = new QStandardItem(it->getDepartureDate());

model->setItem(0, 3, item);

item = new QStandardItem(it->getDepartureTime());

model->setItem(0, 4, item);

item = new QStandardItem(it->getArrivalDate());

model->setItem(0, 5, item);

item = new QStandardItem(it->getArrivalTime());

model->setItem(0, 6, item);

item = new QStandardItem(QString::number(it->getTicketPrice()));

model->setItem(0, 7, item);

item = new QStandardItem(QString::number(it->getAirplaneId()));

model->setItem(0, 8, item);

item = new QStandardItem(QString::number(it->getZmina()));

model->setItem(0, 9, item);

item = new QStandardItem(QString::number(it->getPassCount()));

model->setItem(0, 10, item);

item = new QStandardItem(QString::number(it->getPassCountBuy()));

model->setItem(0, 11, item);

item = new QStandardItem(QString::number(it->getPassCountBuy() \* it->getTicketPrice()));

model->setItem(0, 12, item);

break;

}

}

ui->tableViewFind->setModel(model);

ui->tableViewFind->resizeRowsToContents();

ui->tableViewFind->resizeColumnsToContents();

}

**Додаток Г**

void MenuReis::slotEditArrivalDate() {

if(ui->lineEditEditArrivalDate->displayText().isEmpty()) {

QMessageBox::warning(this, "Помилка", "Заповніть поле");

return;

}

QString newArrivalDate = ui->lineEditEditArrivalDate->displayText();

int editId = ui->lineEditEditId->displayText().toInt();

for(int i = 0; i < listReis.size(); i++) {

if(editId == listReis[i].getId()) {

listReis[i].setArrivalDate(newArrivalDate);

QMessageBox::information(this, "Повідомлення", "Успішно змінено");

return;

}

}

QMessageBox::warning(this, "Помилка", "Id не знайдено");

}

**Додаток Д**

void MenuReis::slotDeleteId() {

int deleteId = ui->lineEditDeleteId->displayText().toInt();

for(int i = 0; i < listReis.size(); i++) {

if(deleteId == listReis[i].getId()) {

listReis.removeAt(i);

QMessageBox::information(this, "Повідомлення", QString("Рейс з id %1 успішно видалено").arg(deleteId));

return;

}

}

QMessageBox::warning(this, "Помилка", "Такий id не знайдено");

}

**Додаток Е**

void MenuReis::**slotSave**() {

QFile reisFile("reis.txt");

if(reisFile.*open*(QIODevice::WriteOnly | QIODevice::Text | QIODevice::Truncate)) {

for(QList<Reis>::iterator it = listReis.begin(); it != listReis.end(); \*it++) {

reisFile.write(qPrintable(QString::number(it->getId())));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(it->getDepartureCity()));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(it->getArrivalCity()));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(it->getDepartureDate()));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(it->getDepartureTime()));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(it->getArrivalDate()));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(it->getArrivalTime()));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(QString::number(it->getAirplaneId())));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(QString::number(it->getZmina())));

reisFile.write(qPrintable(" "));

reisFile.write(qPrintable(QString::number(it->getTicketPrice())));

reisFile.write(qPrintable(" "));

for(int i = 0; i < it->getPassCount(); i++) {

reisFile.write(qPrintable(QString::number(it->getPassanger(i))));

reisFile.write(qPrintable(" "));

}

reisFile.write(qPrintable("\n"));

}

reisFile.*close*();

QMessageBox::information(this, "Повідомлення", "Дані збережено");

}

else {

QMessageBox::warning(this, "Помилка", "Не вдалося відкрити reis.txt для запису даних");

}

}

**Додаток Є**

void MenuReis::dateFromFile() {

QFile reisFile("reis.txt");

QString lStr;

QStringList lStrList;

Reis reis;

int idPass;

if(reisFile.open((QIODevice::ReadOnly | QIODevice::Text))) {

while(!reisFile.atEnd()) {

lStr = reisFile.readLine();

lStrList = lStr.split(' ', QString::SkipEmptyParts);

reis.setId(lStrList.at(0).toInt());

reis.setDepartureCity(lStrList.at(1));

reis.setArrivalCity(lStrList.at(2));

reis.setDepartureDate(lStrList.at(3));

reis.setDepartureTime(lStrList.at(4));

reis.setArrivalDate(lStrList.at(5));

reis.setArrivalTime(lStrList.at(6));

reis.setAirplaneId(lStrList.at(7).toInt());

reis.setZmina(lStrList.at(8).toInt());

reis.setTicketPrice(lStrList.at(9).toInt());

for(int i = 0; i < reis.getPassCount(); i++) {

idPass = lStrList.at(i + 10).toInt();

if(idPass > -1) {

reis.addPassanger(idPass);

}

}

listReis.append(reis);

}

reisFile.close();

}

else {

QMessageBox::warning(this, "Помилка", "Не вдалося відкрити reis.txt для зчитування даних");

}

}

**Додаток Ж**

*Метод перевідки id на унікальність*

bool MenuReis::checkId(int check\_id) {

for(QList<Reis>::iterator it = listReis.begin(); it != listReis.end(); \*it++) {

if (it->getId() == check\_id) {

return true;

}

}

return false;

}

*Метод знаходження максимального вільного id*

int MenuReis::maxId() {

int max\_id = 0;

for(QList<Reis>::iterator it = listReis.begin(); it != listReis.end(); \*it++) {

if (it->getId() > max\_id) {

max\_id = it->getId();

}

}

return max\_id;

}