КУРСОВА РОБОТА

ІП-220419

Група ІП-22-1

Микитій Юрій

2023

Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти та газу

Інститут Інформаційних технологій

КУРСОВИЙ ПРОЄКТ

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: Розробка прикладної програми « АІС Аеропорту»

студента 1 курсу 1 групи

напряму підготовки Інженерія програмного забезпечення

спеціальності Інженерія програмного забезпечення

Микитій Ю. М.

Керівник

зав. Кафедри ІПЗ, професор, д.т.н. Шекета В. І.

Національна шкала:_____

Кількість балів:_____ Оцінка: ECTS _____

м. Івано-Франківськ —2023 рік

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів курсового проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
Вибір варіанту завдання. Теоретична довідка по предметній області	01-03 березня 2023 р.	Виконано
Побудова рішення для виконання проекту. Обрання методу програмування	05-20 квітня 2023 р.	Виконано
Розробка та документування проекту	20 квітня – 15 травня 2023 р.	Виконано
Оформлення пояснювальної записки та компіляція кінцевого проекту	15 травня – 1 червня 2023 р.	Виконано
Захист курсового проекту	???????	

«___» _____ р.

Зміст

Вим	*Лист*	*№ Докум.*	*Підп.*	*Дата*			
Розроб.		Микитій Ю.			Розробка прикладної програми «АІС аеропорту»	*Літера*	*Лист*	*Листів*
Перев.		Шекета В. І.						
Т. Контр.						ІФНТУНГ, ст. гр. ІП-22-1		
Н. Контр.								
Затв.								

Перелік умовних позначень, символів, одиниць, скорочень і термінів

БД- База даних

ООП- Об’єктно орієнтоване програмування

АІС- Автоматизована інформаційна система

Анотація

Моделювання та аналіз автоматизованої інформаційної системи аеропорту є центром цього курсу. Існує кілька різних категорій працівників аеропорту, включаючи пілотів, диспетчерів, техніків, касирів, офіцерів служби безпеки, співробітників служби підтримки та інших. Кожен має певні риси, що стосуються його сфери знань. Аеропорт ділиться на відділи, кожен з яких очолюється начальником і має набір працівників. Організація, розклад і комплектування рейсів входять до компетенції керівництва аеропорту. Кожен літальний апарат укомплектований екіпажем пілотів, інженерів і технічного персоналу. Літак проходить технічний огляд перед зльотом, а в підготовку до польоту включені технічні та технічні компоненти. Розклад рейсів містить інформацію про тип літака, рейс, дні та години відправлення та прибуття, маршрут і вартість квитків. У касах квитки можна забронювати заздалегідь. На вартість квитка впливає маршрут і час відправлення. Існує багато причин затримки або скасування рейсів. Внутрішні, міжнародні, чартерні, вантажні та спеціальні рейси розділені окремо. Квиток, паспорт і закордонний паспорт необхідні для міжнародних рейсів, а також для проходження митниці. Розповсюдженням квитків на чартерні рейси займається компанія, яка організувала рейс. Для того, щоб ефективно керувати всіма аспектами роботи аеропорту, цей проект створить модель автоматизованої інформаційної системи. За допомогою цієї системи можна керувати технічним станом літаків, вирішувати численні проблеми, пов’язані із затримками або скасуваннями рейсів, відстежувати інформацію про персонал, розклад рейсів, продаж квитків і бронювання. Цей проект має вирішальне значення для підвищення ефективності аеропорту, гарантування зручності та безпеки мандрівників, а також покращення управління ресурсами та скорочення витрат.

ВСТУП

Однією з основних парадигм програмування є об'єктно-орієнтоване програмування (ООП), яке підкреслює важливість об'єктів і класів. ООП засновано на понятті симуляції реального світу з класами, що описують якості та поведінку об’єктів, які замінюють фактичні або абстрактні сутності. Зробити програми більш структурованими, модульними та придатними для повторного використання можна за допомогою ООП.

Використання об’єктно-орієнтованого програмування (ООП) має низку переваг, таких як:

Можливість розділити програму на менші незалежні модулі або класи відома як модульність. Кожен клас має дані та пов’язані методи, що спрощує організацію коду. Завдяки цьому розробка, тестування та обслуговування програмного забезпечення спрощуються. Повторне використання: ООП просуває ідею повторного використання коду. Оскільки класи можна успадковувати, ми можемо створювати нові класи зі старих. У результаті ви можете використовувати функціональні можливості вже існуючих класів і змінювати або вдосконалювати їх за потреби. Це мінімізує дублювання коду та забезпечує ефективне використання ресурсів. Інкапсуляція: ООП дозволяє приховати внутрішню роботу класів і обмежити доступ до них загальнодоступним інтерфейсом. Це підвищує стабільність і безпеку програми, надаючи користувачам контроль над доступом до даних і методів класу. Поліморфізм: ООП дозволяє використовувати поліморфізм, який стосується здатності об’єкта виконувати операції в багатьох контекстах. Для цього використовуються методи перевантаження та успадкування. Використовуючи поліморфізм, ви можете створювати стандартизовані інтерфейси та надавати додаткам додаткову гнучкість і розширюваність.

З прикладу діяльності та функціонування бази даних інтернет магазину, виконано аналіз завдань, розроблено алгоритми взаємодії об’єктів та ієрархію класів, програма розроблена на мові C++, у якій розроблено базу даних та реалізовано консольний інтерфейс. Мета цього проекту —

завершити розробку системи керування базами даних для автоматизації системи аеропорту та навчитися впроваджувати ООП на С++. Метою дослідження є створення системи управління базами даних з консольним інтерфейсом на прикладі інтернет магазину. Основною темою дослідження є програмна реалізація моделі об’єктного класу мовою програмування С++. Парадигма ООП є теоретико-методологічною основою розробки . Робота являє собою рукопис із 84 сторінками машинописного тексту, покажчиком 5 використаних джерел та додатками з кодом програми та результатами виконання контрольного прикладу.

1. АНАЛІЗ ТЕОРІЇ ООП В КОНТЕКСТІ ЗАДАЧІ ПРОЕКТУВАННЯ КОНСОЛЬНОЇ БД ЗАСОБАМИ С++ .

1.1 Класи

У цьому проекті використано 4 класи:

1. “WorkerAIR”- клас для опису працівників аеропорту, він містить вектор зі структурою “Worker” , яка в свою чергу містить поля з відповідними змінними наприклад: ім’я, вік, стать. У цьому класі будуть різні методи для отримання інформації, або про всіх працівників одразу, або ж про працівника за певним запитом наприклад, за вказаним віком, де і буде виводитися список всіх людей за такою характеристикою.

Методи, які містяться у цьому класі:

- void printWorkersWithExperience(int experience);
- void GetOllWorkers();
- void GetOllDepartmentHeads();
- void GetOllEmployeesDepartment(int m_department);
- void GetOllEmployeeSexualCharacteristic(char gender);
- void GetOllEmployeesAge(int age);
- void GetOllEmployeesNumberChildren(int number_children);
- void GetOllEmployeeSalary(int salary);
- void GetOllEmployeeInBrigade(int brigade);
- void GetOllEmployeeInOllBrigade();
- void GetOllEmployeeServingSpecificFlight(string flight);
- void GetOllEmployeeTotalAverageSalaryBrigade(int brigade);
- void GetOllPilotsMedicalExamination();
- void GetOllPilotsNotMedicalExaminationYear(int year);
- void GetOllEmployeeGenderPilots(char gender);
- void GetOllPilotsAge(int age);
- void GetOllPilotsSalary(int salary);

2. “PlaneAIR”- клас для опису літаків аеропорту, він містить вектор зі структурою “Plane” , яка в свою чергу містить поля з відповідними змінними наприклад: тип, рейс, кількість скоєних рейсі. У цьому класі будуть різні методи для отримання інформації, або про всі літаки одразу, або ж про літак за певним запитом наприклад, за вказаним типом, де і буде виводитися список всіх літаків за такою характеристикою.

Методи, які містяться у цьому класі:

- void GetOllPlanes();
- void GetOllPlanesAtSpecifiedTime(string time);
- void GetOllNumberFlightsMade(int number);
- void GetOllPeriodTimeTechnical(string period);
- void GetOllDaySendingRepair(string day_sending_repair);
- void GetOllNumberRepairs(int number_repairs);
- void GetOllNumberFlightsRepaired(int number_flights_repaired);
- void GetOllAgeAircraft(int age_aircraft);

3. “FlightAIR”- клас для опису рейсів в аеропорту, він містить вектор зі структурою “Flight” , яка в свою чергу містить поля з відповідними змінними наприклад: маршрут, ціна квитка, час перельоту скоєних рейсі. У цьому класі будуть різні методи для отримання інформації, або про всі рейси одразу, або ж про рейс за певним запитом наприклад, за вказаним маршрутом, де і буде виводитися список всіх рейсів за такою характеристикою.

Методи, які містяться у цьому класі:

- void GetFlightsSpecifiedRoute(string route);
- void GetFlightsTime(int flight_time);
- void GetFlightsTicketPrice(int ticket_price);
- void GetFlightsAllCriteria(string route, int flight_time, int ticket_price);
- void GetFlightsCanceledFlightsFull();

- void GetFlightsCanceledDirection(string direction);
- void GetFlightsCanceledSpecifiedRoute(string route);
- void GetFlightsCancelUnclaimedSeats(int seats);
- void GetFlightsCancelPercentage(int percentage);
- void GetFlightsAllDelayed();
- void GetFlightsReasonFlightDelay(string reason_flight_delay);
- void GetFlightsDelayedSpecifiedRoute(string route);
- void GetFlightsDelayedTicketsDuringDelay(int number_issued_tickets_during_delay);
- void GetAllFlightsTypeAvarage(string type);
- void GetAllFlightDuration(int flight_time);
- void GetAllFlightDepartureTime(string departure_time);
- void GetAllFlightCategories(string flight_categories);
- void GetAllFlightDirection(string direction);
- void GetAllFlightIsuedTickets(string flight);
- void GetAllFlightIsuedTicketsInDay(string day);
- void GetAllFlightIsuedTicketsDirection(string direction);
- void GetAllFlightIsuedTicketsPrice(int ticket_price);

4. “PassengerAIR”- клас для опису пасажирів, він містить вектор зі структурою “Passenger” , яка в свою чергу містить поля з відповідними змінними наприклад: стать, ознака здачі багажу, забронюване місце. У цьому класі будуть різні методи для отримання інформації, або про всх пасажирів одразу, або ж про пасажирів за певним запитом наприклад, за вказаною статтю, де і буде виводитися список всіх пасажирів за такою характеристикою.

Методи, які містяться у цьому класі:

- void GetAllPassangersDepartedSpecifiedDay(string departure_day);
- void GetAllPassangersDepartedSpecifiedDayFlewAbroad(string departure_day);
- void GetAllPassangersCheckedBaggage(bool checked_luggage);

- void GetAllPassangersGender(char gender);
- void GetAllPassangersAge(int age);
- void GetAllNumbersAge(int age);
- void GetAllNumbersGender(char gender);
- void GetFreeReservedPlacesFlight(string flight);
- void GetFreeReservedPlacesDepartureDay(string departure_day);
- void GetFreeReservedPlacesTicketPrice(int ticket_price);
- void GetFreeReservedPlacesDepartureTime(string departure_time);

1.2 Об'єкти

У цій задачі використано 4 об’єкти класу для того, щоб керувати і виводити методи класу, вони реалізуються в Main.cpp в функції “Menu”. Об’єкти взаємодіють між собою у відповідних пунктах меню(в одному пункті може затребуватися об’єкт класу WorkerAIR і PassengerAIR, або ж інші).

- worker- для представлення класу WorkerAIR;
- plane- для представлення класу PlaneAIR;
- flight- для представлення класу FlightAIR;
- passenger- для представлення класу PassengerAIR;

1.3.Наслідування

При роботі з класами під час аналізу своєї задачі, можливо використати наслідування, адже є клас WorkerAIR де є різні працівники аеропорту, яких можна підрозділити на пілотів, диспетчерів, техніків, касирів, працівників служби безпеки, довідникової служби та інших, які адміністративно відносяться кожен до свого відділу. Кожна з перерахованих категорій працівників мала б унікальні атрибути- характеристики, що визначаються професійною спрямованістю. Є 2 дочірніх класи це: Pilot, Engeener які при потребі можуть розбивати батьківський клас і реалізовуватися самостійно.

1.4. Поліморфізм

При розробці проекту було прийняте рішення не визначати, що деякі методи мають різну реалізацію для різних класів. У завдані немає потреби в різних реалізаціях методів для конкретних класів працівників. Якщо всі працівники аеропорту мають однаковий набір функцій і поведінки, то поліморфізм буде зайвим і його використання може додати зайву складність у систему.

1.5. Інкапсуляція

При розробці не було визначено поля та методи, які будуть приховані або доступні для використання. Всі працівники мають доступ до одних і тих самих даних та методів, отже інкапсуляція може додати непередбачувані витрати. Тому прийняте рішення, що можна використовувати простий підхід, де всі поля і методи класу WorkerAIR є доступними для всіх підкласів без обмежень.

1.6. Абстракція

Оскільки я вирішив використовувати наслідування та не використовувати поліморфізм(наведено в розділі 1.3, 1.4), тоді, юуло прийняте рішення, що немає потреби в абстракції, адже кожен клас працівника має спільні атрибути та методи загального інтерфейсу, які прописані в батьківському класі.

2. АНАЛІЗ БІЗНЕС-ЛОГІКИ ПРЕДМЕТНОЇ ОБЛАСТІ КУРСОВОЇ РОБОТИ ЯКА АВТОМАТИЗУЄТЬСЯ ОБЄКТНО-ОРІЄНТОВАНИМИ ЗАСОБАМИ С++ В ФОРМІ КОНСОЛЬНОЇ БД

Аеропорт працює в офлайн режимі. Автоматизована інформаційна система- це складна система апаратного та програмного забезпечення, яка створена для збору, обробки, зберігання та надання даних з метою автоматизації різних бізнес-операцій і максимального збільшення обсягу роботи в певній предметній області.

У роботі цієї системи можна виділити ряд ситуацій:

1. Організація роботи працівників в аеропорту.

Потрібно вести загальний облік працівників з відповідними даними про них, як фізичних осіб та як працівників. Інформація про працівника включає: ім'я, вік, відділ, ознака завідувача відділу, у якій бригаді працює якщо не є головою відділу, рейс який обслуговує якщо не є головою відділу, стаж роботи, стать, ознака наявності дітей, кількість дітей, зарплата, медогляд(для пілотів), рік проходження медогляду(для пілотів). Під час роботи АІС аеропорту виникає необхідність виводу списку працівників по всім критеріям інформації за винятком ім’я працівника.

2. Введення обліку літаків.

Потрібно вести облік літаків, які належать аеропорту. Кожен літаків має свою інформацію: тип, вік, рейс яким він літає, кількість місць, час перебування в аеропорту, час надходження в аеропорт, кількість скоєних рейсів, ознака проходження техогляду, період часу проходження техогляду(ранок, полудень, вечір, ніч),ознака відправлення в ремонт, день відправлення в ремонт, загальна кількість ремонтів, кількість скоєних рейсів до ремонту. У ході роботи виникає необхідність виводу списку літаків за всіма критеріями окрім рейсу, кількості місць, час перебування в аеропорту, час надходження в аеропорт.

3. Обробка роботи рейсів аеропорту.

У будівлі аеропорту знаходиться табло на якому пасажири можуть побачити тип літака, рейс, дні вильоту, час вильоту і прильоту, маршрут (початковий і кінцевий пункти призначення, пункт пересадки), вартість квитка. Квитки на авіарейси можна придбати заздалегідь або забронювати в авіакасах.

4. Робота з клієнтами.

В аеропорті при покупці квитка, працівник буде заносити дані про людину(пасажира) у базу пасажирів. Під час цього необхідно зібрати інформацію про фізичну особу(ім’я, вік, стать, рейс на який йде реєстрація, ознака здачі речей в багажне відділення, місце, ціна квитка на який іде бронювання). Після цього у систему також заноситиметься день відльоту(якщо рейс не є скасований), ознака вильоту за кордон(якщо це міжнародний рейс), час вильоту і ознака здачі квитка. Термін повернення грошей досить широкий – від миттєвого повернення до трьох місяців, залежно від банку отримувача. Якщо, авіакомпанія відміняє рейс або міняє розклад (затримка вильоту більш ніж на 3 години), квиток можна поміняти на новий, за умови що це влаштовує власника квитка.

3. ПОСТАНОВКА ЗАДАЧІ

Створити консольну базу даних на C++ по предметній області «МОДЕЛЮВАННЯ ТА АНАЛІЗ АІС АЕРОПОРТУ» .Необхідно реалізувати такі функції:

1. Запити на виведення даних, що відповідають певним умовам.
2. Вивід списку

3.1 Розробка алгоритму вирішення задач, що виникли під час виконання роботи

Запити на виведення даних, що відповідають певним умовам

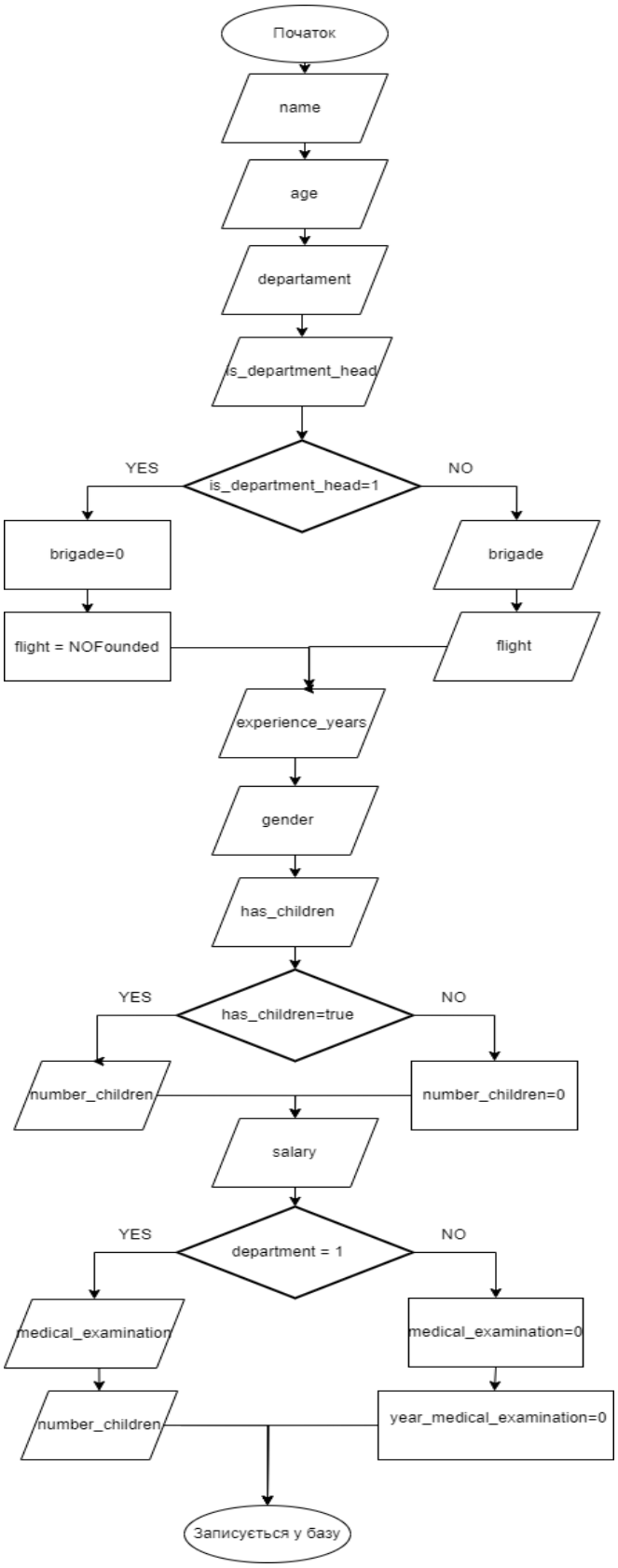
Реалізовано таку кількість запитів для кожної бази даних: Worker- 17 запитів, Plane-8 запитів, Flight-22 запити, Passenger- 12 запитів.

Дані з файлу зчитуються конструктором PlaneAIR() у вектор з типом структури Plane і додаються до основного вектора planes. Розглянемо цей алгоритм на прикладі «Plane.txt»: У ньому є метод GetOllPlanes(): виводить всі літаки. Також є методи, які за певним запитом звіряє змінні з файлом і виводить список літаків. Розглянемо на прикладі методу GetOllNumberFlightsMade(int number) який приймає кількість скоєних перельотів порівнює їх з відповідними даними у базі і виводить список літаків за цією умовою.

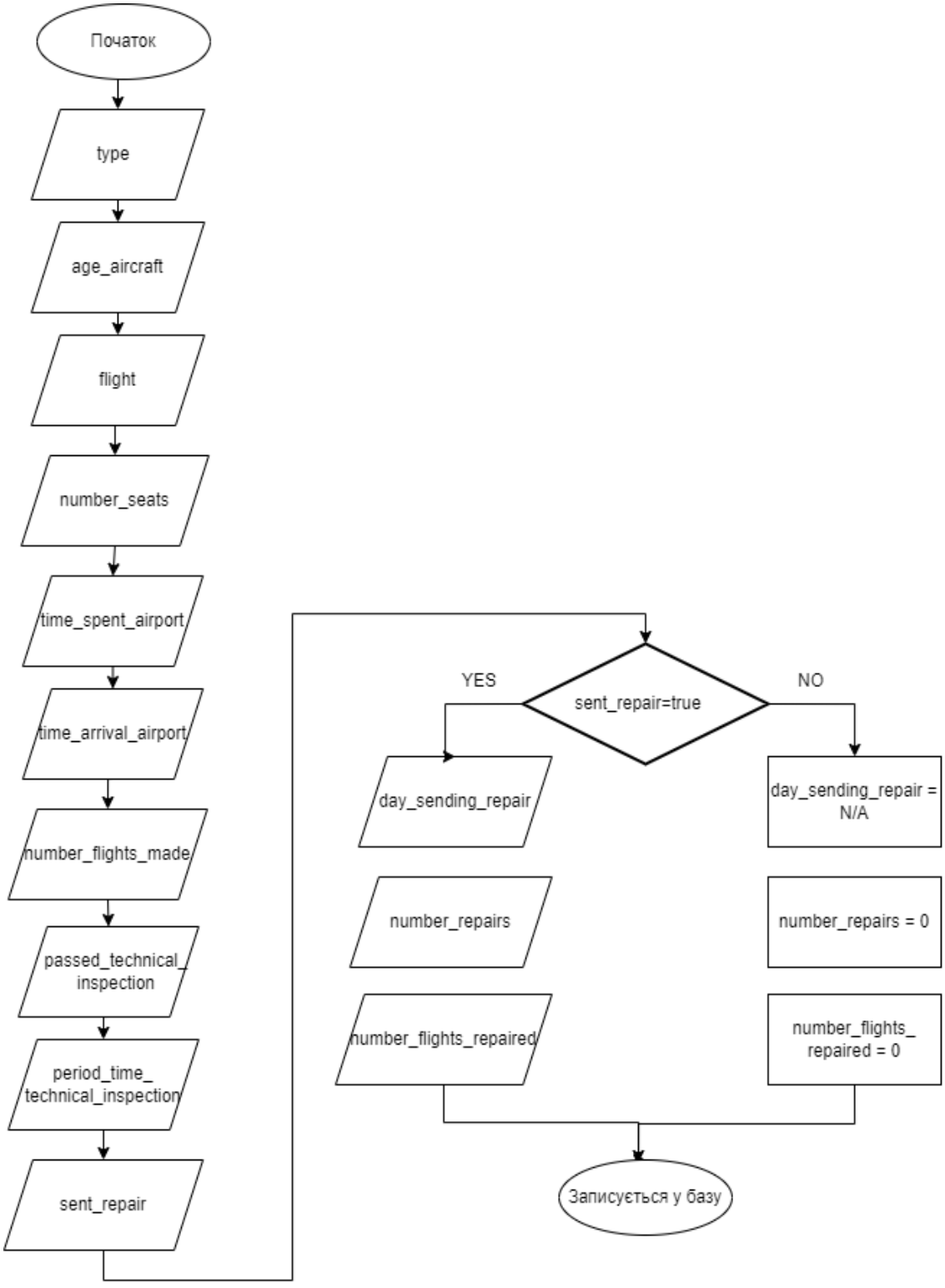
4. ВІЗУАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ ПРЕДМЕТНОЇ ОБЛАСТІ

4.1. Блок-схема рішення (Flowchart diagram)

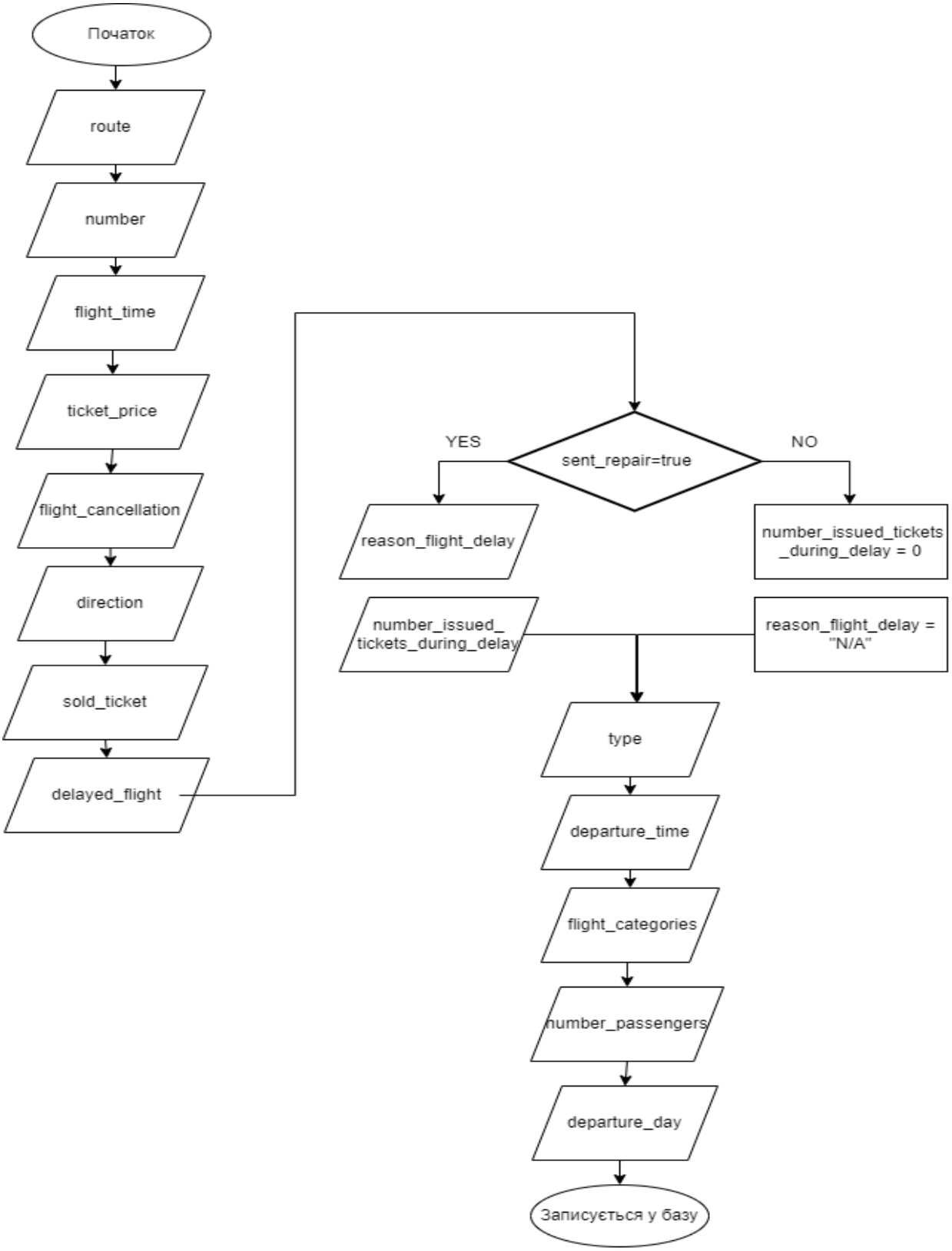
4.1.1 Блок-схема рішення: «Працівник» (Worker)



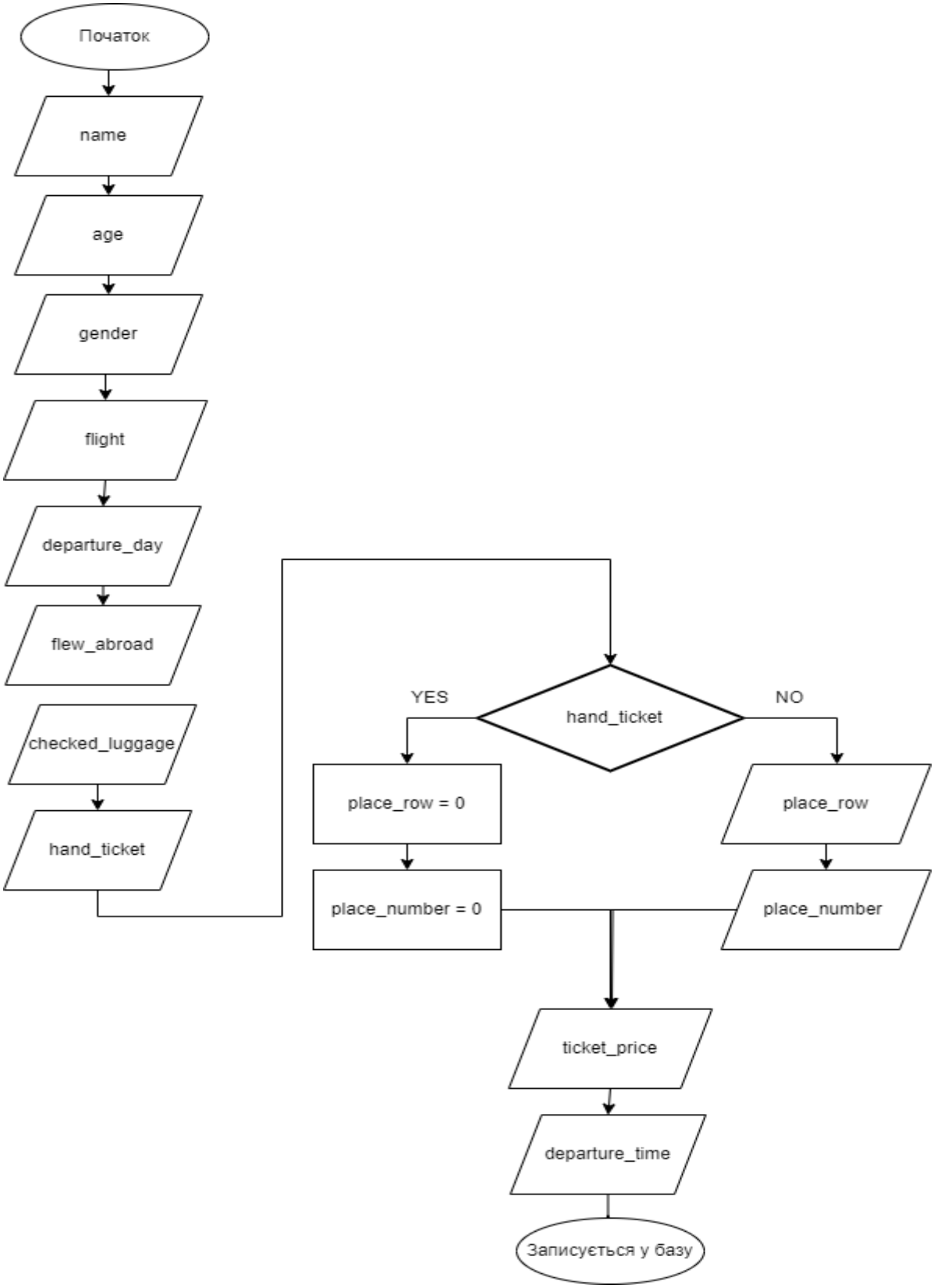
4.1.2 Блок-схема рішення: «Літак» (Plane)



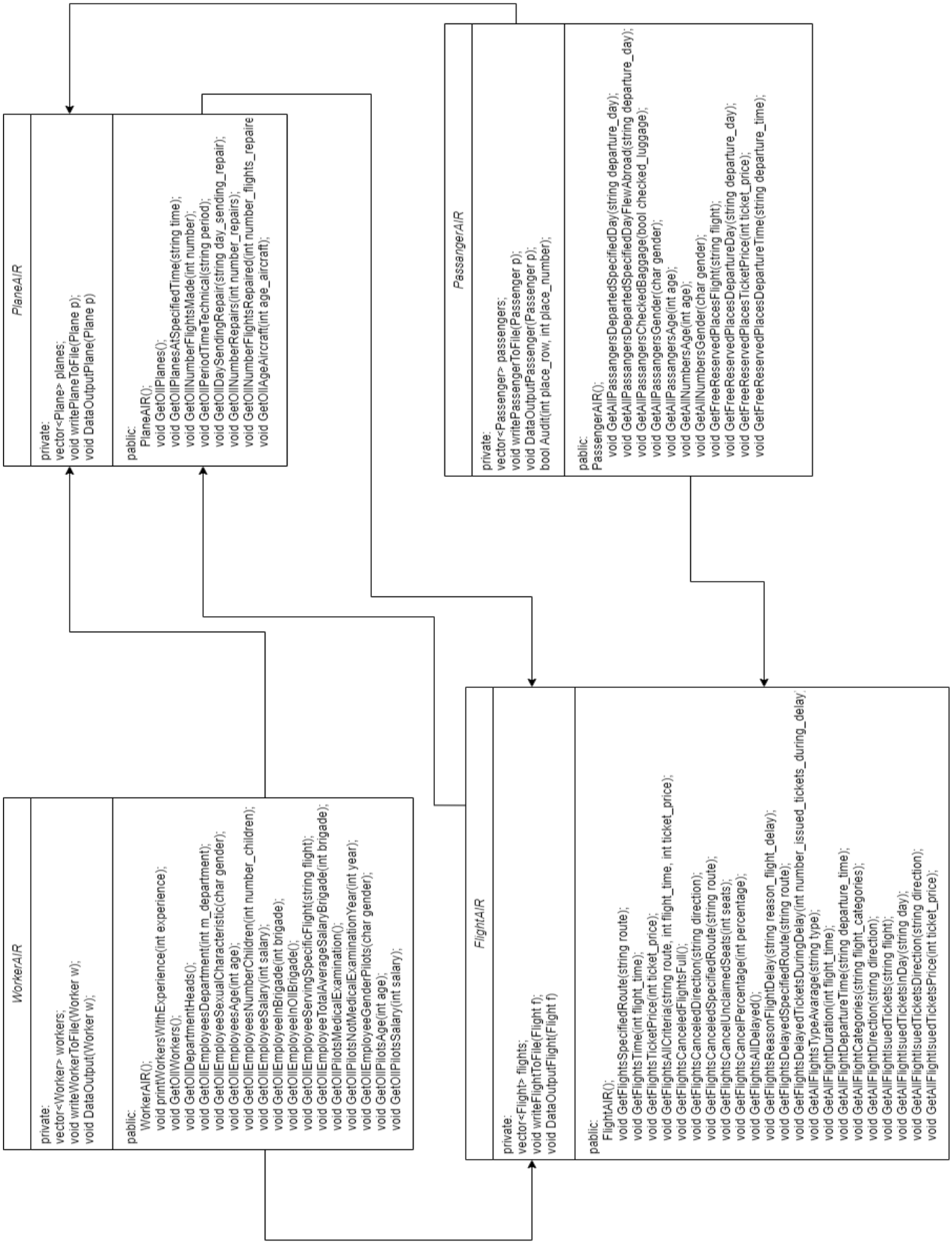
4.1.3 Блок-схема рішення: «Рейс» (Flight)



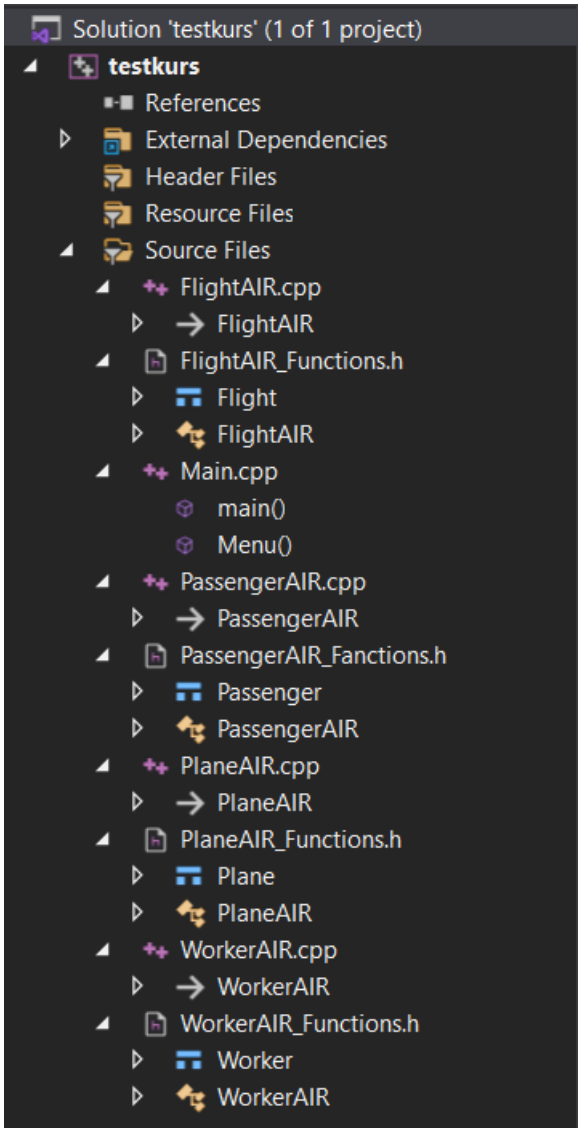
4.1.4 Блок-схема рішення: «Пасажир» (Passenger)



4.2. Діаграма класів (Class UML diagram)



4.3. Дерево проекту (Project Tree)



5. ОПИС КЛАСІВ ТА ЇХ МЕТОДІВ

5.1 Підключені бібліотеки

- <iostream> використовується для організації введення-виведення в мові програмування C++ .
- <fstream> для роботи з файлами, у якому підключено такі заголовкові файли як <ifstream> — бібліотека для файлового введення, і <ofstream> — бібліотека для файлового виведення.
- <vector>додає стандартний шаблон узагальненого програмування мови C++, що реалізує динамічний масив.
- <string> представляє текстовий рядок .
- <Windows.h> надає інтерфейс доступу до Windows API

5.2 Опис коду – файл main.cpp

Усі файли заголовків "WorkerAIR_Functions.h", "PlaneAIR_Functions.h", "FlightAIR_Functions.h" і "PassengerAIR_Functions.h" включені в цей код.

Ми генеруємо об’єкти з різних класів у функції Menu(), зокрема WorkerAIR, PlaneAIR, FlightAIR і PassengerAIR. Далі ми просимо користувача вибрати один із варіантів, після чого виконуємо відповідні дії.

Основна ідея коду полягає у використанні різних функцій із цих класів для отримання різних списків, фільтрації даних і показу результатів на екрані. Вибравши варіант 1, наприклад, користувач може отримати список співробітників залежно від різних факторів, включаючи досвід роботи, стать, вік і зарплату. Після вибору опції користувач введе необхідну інформацію, а результат буде показано на екрані.

Програмне забезпечення контролює вибір опції та викликає відповідні функції за допомогою перемикача. Користувач може вибрати варіант, надати дані та отримати результат. Після операції програма знову пропонує вам вибрати іншу альтернативу або вийти.

Для отримання та обробки різних типів даних цей код містить ряд меню з численними параметрами та підменю. Кожен вибір викликає відповідний метод із відповідного класу, який потім застосовує відповідну логіку.

Метод system("cls") у коді також очищає екран перед показом нових даних.

ВИСНОВКИ

Метою курсової роботи було вивчення теоретичного та практичного змісту з галузі «Об’єктно-орієнтоване програмування» та розвиток практичних навичок програмування на C++.

Виконано роботу по темі курсової “Моделювання та аналіз АІС Аеропорту”. У своїй роботі працював з файлами та функціями, розподілом пам’яті та технічним пошуком певних значень, та ін.

Виконання зазначеного завдання ,створення програми відбувалося в середовищі Visual Studio 2019. Перевірка результатів програми довела, що робота була виконана правильно.

Створена система управління базами даних дозволяє використовувати власні додаткові можливості, крім можливості повної реалізації баз даних на основі розроблених функцій, які відповідають передумовам для роботи з базами даних.

Основні бібліотеки, мова C++ і методи їх реалізації, а також глибоке розуміння роботи консолі було отримано під час розробки курсової роботи.

Було досліджено та проаналізовано мову програмування C++, і було визначено, що це гнучкий інструмент для роботи з базами даних і як компонент у проектах, виконаних іншими мовами програмування.

Таким чином, можна сказати, що C++ є мовою програмування високого рівня, яка підходить для реалізації баз даних на додаток до всіх інших робіт.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Юрчишин В.М. Програмування: навч. посіб. / В.М.Юрчишин, Б.В.Клим, В.Б.Кропивницька. - Івано-Франківськ: ІФНТУНГ, 2012. - 188 с

2. Глинський Я.М. C++ і C++ Builder: навч. посіб. / Я.М Глинський, В.Є Анохін, В.А Ряжська.- Львів: СПД, 2006.-192с.

3. Інтернет-джерело «acode.com.ua».

4. Інтернет-джерело «Wikipedia.org».

5. https://msdn.microsoft.com/uk-ua/library/420970az.aspx

ДОДАТКИ

Додаток А

Код програми мовою С++

Main.cpp

```
#include "WorkerAIR_Functions.h"

#include "PlaneAIR_Functions.h"

#include "FlightAIR_Functions.h"

#include "PassengerAIR_Fanctions.h"

#include <iostream>

using namespace std;



void Menu()

{

   system("cls");

   WorkerAIR worker;

   PlaneAIR plane;

   FlightAIR flight;

   PassengerAIR passenger;

   int option;

   do {

      cout << "Select an option:" << endl;

cout << "Select an option:" << endl;
cout << "1. List of employees" << endl;
cout << "2. Employees (serve a certain flight in divisions, age, average salary)" << endl;
cout << "3. Pilots (passed medical examination: filter by years, sex, age, salary)" <<
endl;
cout << "4. Planes (by time, arrival at the airport, number of flights)" << endl;
cout << "5. State of the aircraft (service period, age of the aircraft, number of repairs,
number of flights before repair)" << endl;
cout << "6. Flights (according to the specified route, flight duration, ticket price, all
these criteria)" << endl;
cout << "7. Canceled flights (in the specified direction, on the specified route, by the
number unclaimed seats, the percentage unclaimed seats)" << endl;
```

```
cout << "8. Delayed flights (indicating the reason for the specified route, the number of
issued tickets for the time of delay)" << endl;
cout << "9. List, number of flights (aircraft of this type, average number of tickets sold,
flight duration, ticket price, departure time)" << endl;
cout << "10. List, number of flights (of the specified category, on a certain direction,
with the specified type of aircraft)" << endl;
cout << "11. Passengers (of a certain flight, that departed on the specified day, flew
abroad on this day, by baggage registration, gender, age)" << endl;
cout << "12. List, number of free and reserved seats(the specified flight, specified day,
according to the specified route, price, at the time of departure)" << endl;
cout << "13. Number of tickets (for a specific flight, on a specified day, on a specific
route, by ticket price, by age, gender)" << endl;
cout << "14. Exit" << endl;

cin >> option;

        cin.ignore(); // Ignore newline character

        switch (option) {

        case 21:

          system("cls");

          worker.addWorker();

          break;

        case 31:

          system("cls");

          plane.addPlane();

          break;

        case 41:

          system("cls");

          flight.addFlight();

          break;

        case 51:

          system("cls");

          passenger.addPassenger();

          break;

        case 1:7
```

```
system("cls");
cout << "Select a request:" << endl;
cout << "1. Get a list of all employees and their number" << endl;
cout << "2. Get a list of department heads" << endl;
cout << "3. Get a list of employees of the department" << endl;
cout << "4. Get a list of employees with experience" << endl;
cout << "5. Get a list of employees with sexual characteristic" << endl;
cout << "6. Get a list of employees with age" << endl;
cout << "7. Get the number of children of an employee" << endl;
cout << "8. Get the amount of the employee's salary" << endl;
cout << "9. Exit" << endl;
int choice_1;
cin >> choice_1;
switch (choice_1)
{
case 1:
   system("cls");
   worker.GetOllWorkers();
   system("pause");
   break;
case 2:
   system("cls");
   worker.GetOllDepartmentHeads();
   system("pause");
   break;
```

```
            case 3:
                system("cls");
                cout << "1.Pilots \n2.Dispatchers \n3.Technigues \n4.Cashieres \n5.Security \n6.Directory service \nEnter department (1-6): ";
                int department;
                cin >> department;
                worker.GetOllEmployeesDepartment(department);
                system("pause");
                break;
            case 4:
                system("cls");
                cout << "Enter experience:";
                int experience;
                cin >> experience;
                worker.printWorkersWithExperience(experience);
                system("pause");
                break;
            case 5:
                system("cls");
                cout << "Enter sexual characteristic(M or F):";
                char gender;
                cin >> gender;
                worker.GetOllEmployeeSexualCharacteristic(gender);
                system("pause");
                break;
            case 6:
```

```
            system("cls");
            cout << "Enter age:";
            int age;
            cin >> age;
            worker.GetOllEmployeesAge(age);
            system("pause");
            break;
        case 7:
            system("cls");
            cout << "Enter number children:";
            int number_children;
            cin >> number_children;
            worker.GetOllEmployeesNumberChildren(number_children);
            system("pause");
            break;
        case 8:
            system("cls");
            cout << "Enter the salary amount:";
            int salary;
            cin >> salary;
            worker.GetOllEmployeeSalary(salary);
            system("pause");
            break;
        }
        system("cls");
```

```
        break;
    case 2:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get workers in the brigade" << endl;
        cout << "2.Get workers in oll department" << endl;
        cout << "3.Get workers in the department" << endl;
        cout << "4.Get a list of employees by serviced flight" << endl;
        cout << "5.Get a list of employees with age"<< endl;
        cout << "6.Get a list of employees total average salary in brigade" << endl;
        cout << "7. Exit" << endl;
        int choice_2;
        cin >> choice_2;
        switch (choice_2)
        {
        case 1:
            system("cls");
            cout << "Enter brigade(1-3): ";
            int brigade;
            cin >> brigade;
            worker.GetOllEmployeeInBrigade(brigade);
            system("pause");
            break;
        case 2:
            system("cls");
```

```
                worker.GetOllEmployeeInOllBrigade();
                system("pause");
                break;
            case 3:
                system("cls");
                cout << "1.Pilots \n2.Dispatchers \n3.Technigues \n4.Cashieres \n5.Security \n6.Directory
service \nEnter department (1-6): ";
                int department;
                cin >> department;
                worker.GetOllEmployeesDepartment(department);
                system("pause");
                break;
            case 4:
                system("cls");
                {string flight;
                cin >> flight;
                worker.GetOllEmployeeServingSpecificFlight(flight); }
                system("pause");
                break;
            case 5:
                system("cls");
                cout << "Enter age:";
                int age;
                cin >> age;
                worker.GetOllEmployeesAge(age);
                system("pause");
```

```
            break;
        case 6:
            system("cls");
            cout << "Enter brigade(1-3): ";
            int m_brigade;
            cin >> m_brigade;
            worker.GetOllEmployeeTotalAverageSalaryBrigade(m_brigade);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 3:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get a list of pilots who have passed a medical examination" << endl;
        cout << "2.Get a list of pilots who did not pass it in the specified year" << endl;
        cout << "3.Get a list of pilots with sexual characteristic" << endl;
        cout << "4.Get a list of pilots with age" << endl;
        cout << "5.Get the amount of pilots salary" << endl;
        cout << "6. Exit" << endl;
        int choice_3;
        cin >> choice_3;
        switch (choice_3)
        {
```

```
case 1:
    system("cls");
    worker.GetOllPilotsMedicalExamination();
    system("pause");
    break;
case 2:
    system("cls");
    cout << "Enter year: ";
    int year;
    cin >> year;
    worker.GetOllPilotsNotMedicalExaminationYear(year);
    system("pause");
    break;
case 3:
    system("cls");
    cout << "Enter sexual characteristic(M or F):";
    char gender;
    cin >> gender;
    worker.GetOllEmployeeGenderPilots(gender);
    system("pause");
    break;
case 4:
    system("cls");
    cout << "Enter age:";
    int age;
```

```
            cin >> age;
            worker.GetOllPilotsAge(age);
            system("pause");
            break;
        case 5:
            system("cls");
            cout << "Enter the salary amount:";
            int salary;
            cin >> salary;
            worker.GetOllPilotsSalary(salary);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 4:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get a list of all plane and their number" << endl;
        cout << "2.Get a list of all the planes that are at the airport at the specified time" << endl;
        cout << "3.Get a list and the total number of planes by arrival time at the airport" << endl;
        cout << "4.Get a list and the total number of planes by the number of flights made." << endl;
        cout << "5. Exit" << endl;
        int choice_4;
        cin >> choice_4;
```

```
switch (choice_4)
{
case 1:
   system("cls");
   plane.GetOllPlanes();
   system("pause");
   break;
case 2:
   system("cls");
   {cout << "Enter time: ";
   string time;
   cin >> time;
   plane.GetOllPlanesAtSpecifiedTime(time); }//non
   system("pause");
   break;
case 3:
   system("cls");
   {cout << "Enter time: ";
   string time;
   cin >> time;
   plane.GetOllPlanesAtSpecifiedTime(time); }
   system("pause");
   break;
case 4:
   system("cls");
```

```
            cout << "Enter number flights made: ";
            int number;
            cin >> number;
            plane.GetOllNumberFlightsMade(number);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 5:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get a list and the total number of aircraft that have passed technical inspection for a certain period of time" << endl;
        cout << "2.Get a list and the total number of aircraft sent for repair at the specified time" << endl;
        cout << "3.Get a list and the total number of aircraft repaired a given number of times" << endl;
        cout << "4.Get a list and the total number of aircraft according to the number of flights before repair" << endl;
        cout << "5.Get a list and the total number of aircraft by age of the aircraft." << endl;
        cout << "6. Exit" << endl;
        int choice_5;
        cin >> choice_5;
        switch (choice_5)
        {
        case 1:
```

```
            system("cls");
            {cout << "Enter a time period(morning, noon, evening, night): ";
            string period;
            cin >> period;
            plane.GetOllPeriodTimeTechnical(period); }
            system("pause");
            break;
        case 2:
            system("cls");
            {cout << "Enter day sending repair: ";
            string day_sending_repair;
            cin >> day_sending_repair;
            plane.GetOllDaySendingRepair(day_sending_repair); }
            system("pause");
            break;
        case 3:
            system("cls");
            cout << "Enter number repairs: ";
            int number_repairs;
            cin >> number_repairs;
            plane.GetOllNumberRepairs(number_repairs);
            system("pause");
            break;
        case 4:
            system("cls");
```

```
            cout << "Enter number flights repaired: ";
            int number_flights_repaired;
            cin >> number_flights_repaired;
            plane.GetOllNumberFlightsRepaired(number_flights_repaired);
            system("pause");
            break;
        case 5:
            system("cls");
            cout << "Enter age aircraft: ";
            int age_aircraft;
            cin >> age_aircraft;
            plane.GetOllAgeAircraft(age_aircraft);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 6:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get a list of flights on the specified route" << endl;
        cout << "2.Get a list of flights by flight duration" << endl;
        cout << "3.Get a list of flights by ticket price" << endl;
        cout << "4.Get a list of flights by all criteria at once" << endl;
        cout << "5. Exit" << endl;
```

```
int choice_6;
cin >> choice_6;
switch (choice_6)
{
case 1:
   system("cls");
   cout << "Enter route: ";
   {string route;
   cin >> route;
   flight.GetFlightsSpecifiedRoute(route); }
   system("pause");
   break;
case 2:
   system("cls");
   cout << "Enter flight time: ";
   int flight_time;
   cin >> flight_time;
   flight.GetFlightsTime(flight_time);
   system("pause");
   break;
case 3:
   system("cls");
   cout << "Enter ticket price: ";
   int ticket_price;
   cin >> ticket_price;
```

```
            flight.GetFlightsTicketPrice(ticket_price);
            system("pause");
            break;
        case 4:
            system("cls");
            {cout << "Enter route: ";
            string route;
            cin >> route;
            cout << "Enter flight time: ";
            int flight_time;
            cin >> flight_time;
            cout << "Enter ticket price: ";
            int ticket_price;
            cin >> ticket_price;
            flight.GetFlightsAllCriteria(route, flight_time, ticket_price); }
            system("pause");
            break;
        }
        system("cls");
        break;
    case 7:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get the list and the total number of canceled flights in full" << endl;
        cout << "2.Get a list and the total number of canceled flights at the indicated direction" << endl;
```

```
            cout << "3.Get the list and total number of canceled flights on the specified route" << endl;
            cout << "4.Get a list and the total number of canceled flights by the number of unclaimed seats" << endl;
            cout << "5.Get a list and the total number of canceled flights in full by the percentage of unclaimed seats" << endl;
            cout << "6. Exit" << endl;
            int choice_7;
            cin >> choice_7;
            switch (choice_7)
            {
            case 1:
               system("cls");
               flight.GetFlightsCanceledFlightsFull();
               system("pause");
               break;
            case 2:
               system("cls");
               cout << "Enter direction: ";
               {string direction;
               cin >> direction;
               flight.GetFlightsCanceledDirection(direction); }
               system("pause");
               break;
            case 3:
               system("cls");
               cout << "Enter route: ";
```

```
            {string route;
            cin >> route;
            flight.GetFlightsCanceledSpecifiedRoute(route); }
            system("pause");
            break;
        case 4:
            system("cls");
            cout << "Enter seats: ";
            int seats;
            cin >> seats;
            flight.GetFlightsCancelUnclaimedSeats(seats);
            system("pause");
            break;
        case 5:
            system("cls");
            cout << "Enter a number as a percentage: ";
            int percentage;
            cin >> percentage;
            flight.GetFlightsCancelPercentage(percentage);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 8:
```

```
            system("cls");
            cout << "Select a request:" << endl;
            cout << "1.Get the full list and total number of delayed flights" << endl;
            cout << "2.Get the list and total number of delayed flights fromthe specified reason" << endl;
            cout << "3.Get a list and the total number of delayed flights on the specified route" << endl;
            cout << "4.Get a list and the total number of delayed flights for the number of issued tickets during the delay." << endl;
            cout << "5. Exit" << endl;
            int choice_8;
            cin >> choice_8;
            switch (choice_8)
            {
            case 1:
               system("cls");
               flight.GetFlightsAllDelayed();
               system("pause");
               break;
            case 2:
               system("cls");
               cout << "Enter reason: ";
               {
                  string reason_flight_delay;
                  getline(cin, reason_flight_delay);
                  flight.GetFlightsReasonFlightDelay(reason_flight_delay); }
               system("pause");
               break;
```

```
        case 3:
            system("cls");
            cout << "Enter route: ";
            {string route;
            cin >> route;
            flight.GetFlightsDelayedSpecifiedRoute(route); }
            system("pause");
            break;
        case 4:
            system("cls");
            cout << "Enter the number of tickets issued during the delay. ";
            int number_issued_tickets_during_delay;
            cin >> number_issued_tickets_during_delay;
            flight.GetFlightsDelayedTicketsDuringDelay(number_issued_tickets_during_delay);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 9:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get a list and the total number of flights operated by aircraft of a given type" << endl;
        cout << "2.Get a list and the total number of flights by flight duration" << endl;
        cout << "3.Get a list of the total number of flights by ticket price" << endl;
```

```
cout << "4.Get the list and total number of flights by departure time" << endl;
cout << "5. Exit" << endl;
int choice_9;
cin >> choice_9;
switch (choice_9)
{
case 1:
   system("cls");
   {cout << "Enter type: ";
   string type;
   cin >> type;
   flight.GetAllFlightsTypeAvarage(type);}
   system("pause");
   break;
case 2:
   system("cls");
   cout << "Enter flight time(in hourse): ";
   int flight_time;
   cin >> flight_time;
   flight.GetAllFlightDuration(flight_time);
   system("pause");
   break;
case 3:
   system("cls");
   cout << "Enter ticket price: ";
```

```
            int ticket_price;
            cin >> ticket_price;
            flight.GetFlightsTicketPrice(ticket_price);
            system("pause");
            break;
        case 4:
            system("cls");
            {cout << "Enter departure time: ";
            string departure_time;
            cin >> departure_time;
            flight.GetAllFlightDepartureTime(departure_time); }
            system("pause");
            break;
        }
        system("cls");
        break;
    case 10:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get the list and total number of flights of the specified category" << endl;
        cout << "2.Get a list and the total number of flights in a certain direction" << endl;
        cout << "3. Exit" << endl;
        int choice_10;
        cin >> choice_10;
        switch (choice_10)
```

```
            {
            case 1:
               system("cls");
               {cout << "Enter flight categories: ";
               string flight_categories;
               cin >> flight_categories;
               flight.GetAllFlightCategories(flight_categories);}
               system("pause");
               break;
            case 2:
               system("cls");
               cout << "Enter direction: ";
               string direction;
               cin >> direction;
               flight.GetAllFlightDirection(direction);
               system("pause");
               break;
            }
         case 11:
            system("cls");
            cout << "Select a request:" << endl;
            cout << "1.Get a list and the total number of passengers on this flight who departed on the specified day" << endl;
            cout << "2.Get a list and the total number of passengers on this flight who flew abroad on the specified day" << endl;
```

```
cout << "3.Get a list and the total number of passengers on this flight, according to the indication of the baggage claim" << endl;
cout << "4.Get the list and total number of passengers on this flight, by gender" << endl;
cout << "5.Get the list and total number of passengers on this flight, by age." << endl;
cout << "6. Exit" << endl;
int choice_11;
cin >> choice_11;
switch (choice_11)
{
case 1:
   system("cls");
   cout << "Enter day: ";
   {string departure_day;
   cin >> departure_day;
   passenger.GetAllPassangersDepartedSpecifiedDay(departure_day);
   system("pause"); }
   break;
case 2:
   system("cls");
   {string departure_day;
   cin >> departure_day;
   passenger.GetAllPassangersDepartedSpecifiedDayFlewAbroad(departure_day);
   system("pause"); }
   break;
case 3:
   system("cls");
```

```
            cout << "Has the passenger checked in luggage (1 for yes, 0 for no): ";
            bool checked_luggage;
            cin >> checked_luggage;
            passenger.GetAllPassangersCheckedBaggage(checked_luggage);
            system("pause");
            break;
        case 4:
            system("cls");
            cout << "Enter passenger gender (M/F): ";
            char gender;
            cin >> gender;
            passenger.GetAllPassangersGender(gender);
            system("pause");
            break;
        case 5:
            system("cls");
            cout << "Enter passenger age: ";
            int age;
            cin >> age;
            passenger.GetAllPassangersAge(age);
            system("pause");
            break;
        }
        system("cls");
        break;
```

```
        case 12:
            system("cls");
            cout << "Select a request:" << endl;
            cout << "1.Get a list and the total number of free and reserved seats on the specified flight" << endl;
            cout << "2.Get a list and the total number of free and reserved seats for the specified day" << endl;
            cout << "3.Get a list and the total number of free and reserved seats according to the specified route" << endl;
            cout << "4.Get a list and the total number of free and reserved seats by price" << endl;
            cout << "5.Get a list and the total number of free and reserved seats by departure time." << endl;
            cout << "6. Exit" << endl;
            int choice_12;
            cin >> choice_12;
            switch (choice_12)
            {
            case 1:
               system("cls");
               {cout << "Enter flight: ";
               string flight;
               cin >> flight;
               passenger.GetFreeReservedPlacesFlight(flight);
               system("pause");
               break;}
            case 2:
               system("cls");
```

```
    {cout << "Enter departure day: ";
    string departure_day;
    cin >> departure_day;
    passenger.GetFreeReservedPlacesDepartureDay(departure_day);}
    system("pause");
    break;
  case 3:
    system("cls");
    {cout << "Enter flight: ";
    string flight;
    cin >> flight;
    passenger.GetFreeReservedPlacesFlight(flight);
    system("pause");
    break; }
    system("pause");
    break;
  case 4:
    system("cls");
    cout << "Enter ticket price: ";
    int ticket_price;
    cin >> ticket_price;
    passenger.GetFreeReservedPlacesTicketPrice(ticket_price);
    system("pause");
    break;
  case 5:
```

```
            system("cls");
            cout << "Enter departure time: ";
            string departure_time;
            cin >> departure_time;
            passenger.GetFreeReservedPlacesDepartureTime(departure_time);
            system("pause");
            break;
        }
        system("cls");
        break;
    case 13:
        system("cls");
        cout << "Select a request:" << endl;
        cout << "1.Get the total number of issued tickets for a certain flight" << endl;
        cout << "2.Get the total number of issued tickets on the specified day" << endl;
        cout << "3.Get the total number of given tickets for a specific route" << endl;
        cout << "4.Get the total number of issued tickets by ticket price" << endl;
        cout << "5.Get the total number of issued tickets by age" << endl;
        cout << "6.Get the total number of issued tickets for a certain flight, by gender" << endl;
        cout << "7. Exit" << endl;
        int choice_13;
        cin >> choice_13;
        switch (choice_13)
        {
        case 1:
```

```
            system("cls");
            {cout << "Enter flights: ";
            string flights;
            cin >> flights;
            flight.GetAllFlightIsuedTickets(flights);
            system("pause"); }
            break;
        case 2:
            system("cls");
            {cout << "Enter day(DD.MM.YYYY): ";
            string day;
            cin >> day;
            flight.GetAllFlightIsuedTicketsInDay(day);
            system("pause"); }
            break;
        case 3:
            system("cls");
            {cout << "Enter direction: ";
            string direction;
            cin >> direction;
            flight.GetAllFlightIsuedTicketsDirection(direction);
            system("pause"); }
            break;
        case 4:
            system("cls");
```

```
            cout << "Enter ticket price: ";
            int ticket_price;
            cin >> ticket_price;
            flight.GetAllFlightIsuedTicketsPrice(ticket_price);
            system("pause");
            break;
        case 5:
            system("cls");
            cout << "Enter age: ";
            int age;
            cin >> age;
            passenger.GetAllNumbersAge(age);
            system("pause");
            break;
        case 6:
            system("cls");
            cout << "Enter gender (M/F): ";
            char gender;
            cin >> gender;
            passenger.GetAllNumbersGender(gender);
            system("pause");
            break;
        }
        system("cls");
        break;
```

```
        case 14:
            system("cls");
            break;
        default:
            cout << "Invalid option. Try again." << endl;
        }
    } while (option != 15);
}

int main() {

    ifstream WelcomPage("WelcomePage.txt");
    string templine;
    while (getline(WelcomPage, templine)) {
        std::cout << templine << '\n';
        Sleep(30);
    }
    WelcomPage.close();
    int choice;
    cout << "\t\t\tEnter 1 to continue, 0 to exit" << endl;
    cin >> choice;
    if (choice == 1)
    {
        Menu();
    }
```

```
    return 0;

}
```

class_WorkerAIR

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <Windows.h>
using namespace std;

struct Worker {
    string name;
    int age;
    int department;
    bool is_department_head;
    int brigade;
    string flight;
    int experience_years;
    char gender;
    bool has_children;
    int number_children;
    int salary;
    int medical_examination;
    int year_medical_examination;
};


class WorkerAIR {
private:

    vector<Worker> workers;
    void WorkerAIR::writeWorkerToFile(Worker w) {
    ofstream file("Worker.txt", ios::app);
    file << w.name << endl;
    file << w.age << endl;
    file << w.department << endl;
    file << w.is_department_head << endl;
    file << w.brigade << endl;
    file << w.flight << endl;
    file << w.experience_years << endl;
    file << w.gender << endl;
    file << w.has_children << endl;
    file << w.number_children << endl;
    file << w.salary << endl;
    file << w.medical_examination << endl;
    file << w.year_medical_examination << endl;
    file.close();
    system("cls");
}    void DataOutput(Worker w)
    {
        cout << "Name: " << w.name << endl;
        cout << "Age: " << w.age << endl;
        cout << "Department: " << w.department << endl;
        cout << "Is department head: " << (w.is_department_head ? "Yes" : "No") << endl;
        cout << "Brigade: " << w.brigade << endl;
        cout << "The flight it serves: " << w.flight << endl;
```

```
        cout << "Experience (years): " << w.experience_years << endl;
        cout << "Gender: " << w.gender << endl;
        cout << "Has children: " << (w.has_children ? "Yes" : "No") << endl;
        cout << "Number of children: " << w.number_children << endl;
        cout << "Salary: " << w.salary << endl;
        cout << endl;
    }

public:

    WorkerAIR::WorkerAIR() {
    ifstream file("Worker.txt");
    string name, flight;
    bool is_department_head, has_children;
    int experience_years, salary, department, number_children, brigade, age,
medical_examination, year_medical_examination;
    char gender;
    while (file >> name >> age >> department >> is_department_head >> brigade >> flight >>
experience_years >> gender >> has_children >> number_children >> salary >>
medical_examination >> year_medical_examination) {
        Worker w = { name, age, department, is_department_head, brigade, flight,
experience_years, gender, has_children, number_children, salary, medical_examination,
year_medical_examination };
        workers.push_back(w);
    }
    file.close();
}
    void WorkerAIR::printWorkersWithExperience(int experience) {
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.experience_years == experience) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllWorkers()
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        DataOutput(w);
        m_number_employees++;
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllDepartmentHeads()
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.is_department_head == true) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeesDepartment(int m_department)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.department == m_department) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
```

```
void WorkerAIR::GetOllEmployeeSexualCharacteristic(char gender)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.gender == gender) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeesAge(int age)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.age == age) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeesNumberChildren(int number_children)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.number_children == number_children) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeeSalary(int salary)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.salary == salary) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}

void WorkerAIR::GetOllEmployeeInBrigade(int brigade)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.brigade == brigade) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeeInOllBrigade()
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.brigade != 0) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeeServingSpecificFlight(string flight)
```

```
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.flight == flight) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeeTotalAverageSalaryBrigade(int brigade)
{
    int m_number_employees = 0;
    int salary = 0;
    for (Worker w : workers) {
        if (w.brigade == brigade) {
            DataOutput(w);
            m_number_employees++;
            salary += w.salary;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
    cout << "Total average salary in brigade: " << (salary / m_number_employees) << endl;
    system("cls");
}

void WorkerAIR::GetOllPilotsMedicalExamination()
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.medical_examination == 1) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllPilotsNotMedicalExaminationYear(int year)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.year_medical_examination == year) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllEmployeeGenderPilots(char gender)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.gender == gender && w.department == 1) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllPilotsAge(int age)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.age == age && w.department == 1) {
            DataOutput(w);
            m_number_employees++;
        }
    }
```

```
    cout << "\nNumber of employees: " << m_number_employees << endl;
}
void WorkerAIR::GetOllPilotsSalary(int salary)
{
    int m_number_employees = 0;
    for (Worker w : workers) {
        if (w.salary == salary && w.department == 1) {
            DataOutput(w);
            m_number_employees++;
        }
    }
    cout << "\nNumber of employees: " << m_number_employees << endl;
}};

class Pilot : public Worker {
public:
    string license_type;
    int flight_hours;

    Pilot(string name, int age, int department, bool is_department_head, int brigade,
string flight, int experience_years, char gender, bool has_children, int number_children,
int salary, int medical_examination, int year_medical_examination, string license_type, int
flight_hours) : Worker{ name, age, department, is_department_head, brigade, flight,
experience_years, gender, has_children, number_children, salary, medical_examination,
year_medical_examination }, license_type(license_type), flight_hours(flight_hours) {}
};

class Engineer : public Worker {
public:
    string specialty;
    int completed_projects;

    Engineer(string name, int age, int department, bool is_department_head, int brigade,
string flight, int experience_years, char gender, bool has_children, int number_children,
int salary, int medical_examination, int year_medical_examination, string specialty, int
completed_projects) : Worker{ name, age, department, is_department_head, brigade, flight,
experience_years, gender, has_children, number_children, salary, medical_examination,
year_medical_examination }, specialty(specialty), completed_projects(completed_projects) {}
};
```

class_PlaneAIR

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <Windows.h>
using namespace std;

struct Plane{
       string type;
       int age_aircraft;
       string flight;
       int number_seats;
       string time_spent_airport;
       string time_arrival_airport;
       int number_flights_made;
       bool passed_technical_inspection;
       string period_time_technical_inspection;
       bool sent_repair;
       string day_sending_repair;
       int number_repairs;
       int number_flights_repaired;
};

class PlaneAIR
```

```
{
private:
	vector<Plane> planes;
	void PlaneAIR::writePlaneToFile(Plane p)
{
    ofstream file("Plane.txt", ios::app);
    file << p.type << endl;
    file << p.age_aircraft << endl;
    file << p.flight << endl;
    file << p.number_seats << endl;
    file << p.time_spent_airport << endl;
    file << p.time_arrival_airport << endl;
    file << p.number_flights_made << endl;
    file << p.passed_technical_inspection << endl;
    file << p.period_time_technical_inspection << endl;
    file << p.sent_repair << endl;
    file << p.day_sending_repair << endl;
    file << p.number_repairs << endl;
    file << p.number_flights_repaired << endl;
    file.close();
    system("cls");
}
	void DataOutputPlane(Plane p)
	{
		cout << "Type: " << p.type << endl;
		cout << "Age of the aircraft: " << p.age_aircraft << endl;
		cout << "Flight: " << p.flight << endl;
		cout << "Number of seats: " << p.number_seats << endl;
		cout << "Time spent at airport: " << p.time_spent_airport << endl;
		cout << "Time of arrival at airport: " << p.time_arrival_airport << endl;
		cout << "Number of flights made: " << p.number_flights_made << endl;
		cout << "Passed technical inspection: " << (p.passed_technical_inspection ?
"Yes" : "No") << endl;
		cout << "Period of time for technical inspection: " <<
p.period_time_technical_inspection << endl;
		cout << "Sent for repair: " << (p.sent_repair ? "Yes" : "No") << endl;
		cout << "Day sent for repair: " << p.day_sending_repair << endl;
		cout << "Number of repairs: " << p.number_repairs << endl;
		cout << "Number of flights repaired: " << p.number_flights_repaired << endl;
		cout << endl;
	}
public:
	PlaneAIR::PlaneAIR() {
    // Read planes from file
    ifstream file("Plane.txt");
    string type, flight, time_spent_airport, time_arrival_airport,
period_time_technical_inspection, day_sending_repair;
    bool passed_technical_inspection, sent_repair;
    int age_aircraft, number_seats, number_flights_made, number_repairs,
number_flights_repaired;
    while (file >> type >> age_aircraft >> flight >> number_seats >> time_spent_airport >>
time_arrival_airport >> number_flights_made >> passed_technical_inspection >>
period_time_technical_inspection >> sent_repair >> day_sending_repair >> number_repairs >>
number_flights_repaired) {
        Plane p = { type, age_aircraft, flight, number_seats, time_spent_airport,
time_arrival_airport, number_flights_made, passed_technical_inspection,
period_time_technical_inspection, sent_repair, day_sending_repair, number_repairs,
number_flights_repaired };
        planes.push_back(p);
    }
    file.close();
}

void PlaneAIR::GetOllPlanes()
{
    int m_number_planes = 0;
    for (Plane p : planes) {
        DataOutputPlane(p);
```

```
            m_number_planes++;
        }
        Sleep(30);
        cout << "\nNumber of planes: " << m_number_planes << endl;

    }
    void PlaneAIR::GetOllPlanesAtSpecifiedTime(string time)
    {
        int m_number_planes = 0;
        for (Plane p : planes) {
            if (p.time_arrival_airport == time) {
                DataOutputPlane(p);
                m_number_planes++;
            }
        }
        cout << "\nNumber of planes: " << m_number_planes << endl;
    }
    void PlaneAIR::GetOllNumberFlightsMade(int number)
    {
        int m_number_planes = 0;
        for (Plane p : planes) {
            if (p.number_flights_made == number) {
                DataOutputPlane(p);
                m_number_planes++;
            }
        }
        cout << "\nNumber of planes: " << m_number_planes << endl;
    }
    void PlaneAIR::GetOllPeriodTimeTechnical(string period)
    {
        int m_number_planes = 0;
        for (Plane p : planes) {
            if (p.period_time_technical_inspection == period) {
                DataOutputPlane(p);
                m_number_planes++;
            }
        }
        cout << "\nNumber of planes: " << m_number_planes << endl;
    }

    void PlaneAIR::GetOllDaySendingRepair(string day_sending_repair)
    {
        int m_number_planes = 0;
        for (Plane p : planes) {
            if (p.day_sending_repair == day_sending_repair) {
                DataOutputPlane(p);
                m_number_planes++;
            }
        }
        cout << "\nNumber of planes: " << m_number_planes << endl;
    }
    void PlaneAIR::GetOllNumberRepairs(int number_repairs)
    {
        int m_number_planes = 0;
        for (Plane p : planes) {
            if (p.number_repairs == number_repairs) {
                DataOutputPlane(p);
                m_number_planes++;
            }
        }
        cout << "\nNumber of planes: " << m_number_planes << endl;
    }
    void PlaneAIR::GetOllNumberFlightsRepaired(int number_flights_repaired)
    {
        int m_number_planes = 0;
        for (Plane p : planes) {
            if (p.number_flights_repaired == number_flights_repaired) {
                DataOutputPlane(p);
```

```
            m_number_planes++;
        }
    }
    cout << "\nNumber of planes: " << m_number_planes << endl;
}
void PlaneAIR::GetOllAgeAircraft(int age_aircraft)
{
    int m_number_planes = 0;
    for (Plane p : planes) {
        if (p.age_aircraft == age_aircraft) {
            DataOutputPlane(p);
            m_number_planes++;
        }
    }
    cout << "\nNumber of planes: " << m_number_planes << endl;
}};
```

class_FlightAIR

```
#include "PlaneAIR_Functions.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <Windows.h>
using namespace std;

struct Flight {
       string route;
       int number;
       int flight_time;
       int ticket_price;
       bool flight_cancellation;
       string direction;
       int sold_ticket;
       int number_issued_tickets_during_delay;
       string type;
       bool delayed_flight;
       string reason_flight_delay;
       string departure_time;
       string flight_categories;
       int number_passengers;
       string departure_day;
};

class FlightAIR
{
private:
       vector<Flight> flights;
       void FlightAIR::writeFlightToFile(Flight f) {
    ofstream file("Flight.txt", ios::app);
    file << f.route << endl;
    file << f.number << endl;
    file << f.flight_time << endl;
    file << f.ticket_price << endl;
    file << f.flight_cancellation << endl;
    file << f.direction << endl;
    file << f.sold_ticket << endl;
    file << f.number_issued_tickets_during_delay << endl;
    file << f.type << endl;
    file << f.delayed_flight << endl;
    file << f.reason_flight_delay << endl;
    file << f.departure_time << endl;
    file << f.flight_categories << endl;
    file << f.number_passengers << endl;
```

```
    file << f.departure_day << endl;
    file.close();
    system("cls");
}
       void DataOutputFlight(Flight f)
       {
              cout << "Route: " << f.route << endl;
              cout << "Number: " << f.number << endl;
              cout << "Flight time: " << f.flight_time << endl;
              cout << "Ticket price: " << f.ticket_price << endl;
              cout << "Flight cancellation: " << (f.flight_cancellation ? "Yes" : "No") <<
endl;
              cout << "Direction: " << f.direction << endl;
              cout << "Sold ticket: " << f.sold_ticket << endl;
              cout << "Number of issued tickets during delay: " <<
f.number_issued_tickets_during_delay << endl;
              cout << "Type: " << f.type << endl;
              cout << "Delayed flight: " << (f.delayed_flight ? "Yes" : "No") << endl;
              cout << "Reason for flight delay: " << f.reason_flight_delay << endl;
              cout << "Departure time: " << f.departure_time << endl;
              cout << "Flight categories: " << f.flight_categories << endl;
              cout << "Number of passengers: " << f.number_passengers << endl;
              cout << "Departure day: " << f.departure_day << endl;
              cout << endl;
       }
public:
       FlightAIR::FlightAIR() {
    // Read flights from file
    ifstream file("Flight.txt");
    string route, direction, type, reason_flight_delay, departure_time, flight_categories,
departure_day;
    bool flight_cancellation, delayed_flight;
    int number, flight_time, ticket_price, sold_ticket, number_issued_tickets_during_delay,
number_passengers;
    while (file >> route >> number >> flight_time >> ticket_price >> flight_cancellation >>
direction >> sold_ticket >> number_issued_tickets_during_delay >> type >> delayed_flight >>
reason_flight_delay >> departure_time >> flight_categories >> number_passengers >>
departure_day) {
        Flight f = { route, number, flight_time, ticket_price, flight_cancellation,
direction, sold_ticket, number_issued_tickets_during_delay, type, delayed_flight,
reason_flight_delay, departure_time, flight_categories, number_passengers, departure_day };
        flights.push_back(f);
    }
    file.close();
}


void FlightAIR::GetFlightsSpecifiedRoute(string route)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.route == route) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsTime(int flight_time)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.flight_time == flight_time) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
```

```
}
void FlightAIR::GetFlightsTicketPrice(int ticket_price)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.ticket_price == ticket_price) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsAllCriteria(string route, int flight_time,  int ticket_price)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.route == route && f.flight_time == flight_time && f.ticket_price ==
ticket_price) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}

void FlightAIR::GetFlightsCanceledFlightsFull()
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.flight_cancellation == true) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsCanceledDirection(string direction)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.flight_cancellation == true && f.direction == direction) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsCanceledSpecifiedRoute(string route)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.flight_cancellation == true && f.route == route) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsCancelUnclaimedSeats(int seats)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.flight_cancellation == true && (120- f.number_passengers)== seats) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
```

```
}
void FlightAIR::GetFlightsCancelPercentage(int percentage)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.flight_cancellation == true && ((120 - f.number_passengers)*10/12) ==
percentage) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}

void FlightAIR::GetFlightsAllDelayed()
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.delayed_flight == true) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsReasonFlightDelay(string reason_flight_delay)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.delayed_flight == true && f.reason_flight_delay == reason_flight_delay) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsDelayedSpecifiedRoute(string route)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.delayed_flight == true && f.route == route) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetFlightsDelayedTicketsDuringDelay(int number_issued_tickets_during_delay)
{
    int m_number_flights = 0;
    for (Flight f : flights) {
        if (f.delayed_flight == true && f.number_issued_tickets_during_delay ==
number_issued_tickets_during_delay) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}

void FlightAIR::GetAllFlightsTypeAvarage(string type)
{
    int m_number_flights = 0;
    int m_number_sold_ticket = 0;
    int amount_tickets_sold = 0;
    for (Flight f : flights) {
        if (f.type == type) {
            DataOutputFlight(f);
```

```
            m_number_flights++;
            amount_tickets_sold += f.sold_ticket;
            m_number_sold_ticket++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
    cout << "\nThe average number of sold tickets for certain routes: " <<
(amount_tickets_sold / m_number_sold_ticket);
}
void FlightAIR::GetAllFlightDuration(int flight_time)
{
    int m_number_flights = 0;

    for (Flight f : flights) {
        if (f.flight_time == flight_time) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetAllFlightDepartureTime(string departure_time)
{
    int m_number_flights = 0;

    for (Flight f : flights) {
        if (f.departure_time == departure_time) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}

void FlightAIR::GetAllFlightCategories(string flight_categories)
{
    int m_number_flights = 0;

    for (Flight f : flights) {
        if (f.flight_categories == flight_categories) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}
void FlightAIR::GetAllFlightDirection(string direction)
{
    int m_number_flights = 0;

    for (Flight f : flights) {
        if (f.direction == direction) {
            DataOutputFlight(f);
            m_number_flights++;
        }
    }
    cout << "\nNumber of flights: " << m_number_flights << endl;
}


void FlightAIR::GetAllFlightIsuedTickets(string flight)
{
    int m_number_issued_tickets = 0;

    for (Flight f : flights) {
        if (f.route == flight)
        {
            m_number_issued_tickets +=f.number_issued_tickets_during_delay;
```

```
        }
    }
    cout << "\nThe number of issued tickets: " << m_number_issued_tickets << endl;
}
void FlightAIR::GetAllFlightIsuedTicketsInDay(string day)
{
    int m_number_issued_tickets = 0;

    for (Flight f : flights) {
        if (f.departure_day == day)
        {
            m_number_issued_tickets += f.number_issued_tickets_during_delay;
        }
    }
    cout << "\nThe number of issued tickets: " << m_number_issued_tickets << endl;
}
void FlightAIR::GetAllFlightIsuedTicketsDirection(string direction)
{
    int m_number_issued_tickets = 0;

    for (Flight f : flights) {
        if (f.direction == direction)
        {
            m_number_issued_tickets += f.number_issued_tickets_during_delay;
        }
    }
    cout << "\nThe number of issued tickets: " << m_number_issued_tickets << endl;
}
void FlightAIR::GetAllFlightIsuedTicketsPrice(int ticket_price)
{
    int m_number_issued_tickets = 0;

    for (Flight f : flights) {
        if (f.ticket_price == ticket_price)
        {
            m_number_issued_tickets += f.number_issued_tickets_during_delay;
        }
    }
    cout << "\nThe number of issued tickets: " << m_number_issued_tickets << endl;
}
   };
```

class_PassengerAIR

```
#include "PlaneAIR_Functions.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <Windows.h>
using namespace std;

struct Passenger {
       string name;
       int age;
       char gender;
       string flight;
       string departure_day;
       bool flew_abroad;
       bool checked_luggage;
       int place_row;
       int place_number;
       bool hand_ticket;
       int ticket_price;
```

```
        string departure_time;
};

class PassengerAIR
{
private:
        vector<Passenger> passengers;
        void PassengerAIR::writePassengerToFile(Passenger p) {
        ofstream file("Passenger.txt", ios::app);
        file << p.name << endl;
        file << p.age << endl;
        file << p.gender << endl;
        file << p.flight << endl;
        file << p.departure_day << endl;
        file << p.flew_abroad << endl;
        file << p.checked_luggage << endl;
        file << p.place_row << endl;
        file << p.place_number << endl;
        file << p.hand_ticket << endl;
        file << p.ticket_price << endl;
        file << p.departure_time << endl;
        file.close();
        system("cls");
}       void DataOutputPassenger(Passenger p)
        {
                cout << "Name: " << p.name << endl;
                cout << "Age: " << p.age << endl;
                cout << "Gender: " << p.gender << endl;
                cout << "Flight: " << p.flight << endl;
                cout << "Departure day: " << p.departure_day << endl;
                cout << "Flew abroad: " << (p.flew_abroad ? "Yes" : "No") << endl;
                cout << "Checked luggage: " << (p.checked_luggage ? "Yes" : "No") << endl;
                cout << "Place row: " << p.place_row << endl;
                cout << "Place number: " << p.place_number << endl;
                cout << "Ticket price : " << p.ticket_price << endl;
                cout << "Departure time: " << p.departure_time << endl;
                cout << endl;
        }
        bool Audit(int place_row, int place_number)
        {
                for (Passenger p : passengers) {
                        if (p.place_row== place_row && p.place_number == place_number) {
                                cout << "\nThe place is occupied " << endl;
                                system("pause");
                                system("cls");
                        }
                        else
                        {
                                return true;
                                system("cls");
                        }
                }
        }

public:
        PassengerAIR::PassengerAIR() {
        ifstream file("Passenger.txt");
        string name, departure_day, flight, departure_time;
        int age, place_row, place_number, ticket_price;
        char gender;
        bool flew_abroad, checked_luggage, hand_ticket;
        while (file >> name >> age >> gender >> flight >> departure_day >> flew_abroad >>
checked_luggage >> place_row >> place_number >> hand_ticket >> ticket_price >>
departure_time) {
                Passenger p = { name, age, gender, flight,  departure_day, flew_abroad,
checked_luggage, place_row, place_number, hand_ticket, ticket_price, departure_time };
                passengers.push_back(p);
        }
```

```
	file.close();
}

void PassengerAIR::GetAllPassangersDepartedSpecifiedDay(string departure_day)
{
	int m_number_passengers = 0;
	for (Passenger p : passengers) {
		if (p.departure_day == departure_day) {
			DataOutputPassenger(p);
			m_number_passengers++;
		}
	}
	cout << "\nNumber of passengers: " << m_number_passengers << endl;
}
void PassengerAIR::GetAllPassangersDepartedSpecifiedDayFlewAbroad(string departure_day)
{
	int m_number_passengers = 0;
	for (Passenger p : passengers) {
		if (p.departure_day == departure_day && p.flew_abroad==true) {
			DataOutputPassenger(p);
			m_number_passengers++;
		}
	}
	cout << "\nNumber of passengers: " << m_number_passengers << endl;
}
void PassengerAIR::GetAllPassangersCheckedBaggage(bool checked_luggage)
{
	int m_number_passengers = 0;
	for (Passenger p : passengers) {
		if (p.checked_luggage == checked_luggage) {
			DataOutputPassenger(p);
			m_number_passengers++;
		}
	}
	cout << "\nNumber of passengers: " << m_number_passengers << endl;
}
void PassengerAIR::GetAllPassangersGender(char gender)
{
	int m_number_passengers = 0;
	for (Passenger p : passengers) {
		if (p.gender == gender) {
			DataOutputPassenger(p);
			m_number_passengers++;
		}
	}
	cout << "\nNumber of passengers: " << m_number_passengers << endl;
}
void PassengerAIR::GetAllPassangersAge(int age)
{
	int m_number_passengers = 0;
	for (Passenger p : passengers) {
		if (p.age == age) {
			DataOutputPassenger(p);
			m_number_passengers++;
		}
	}
	cout << "\nNumber of passengers: " << m_number_passengers << endl;
}
void PassengerAIR::GetAllNumbersAge(int age)
{
	int m_number_issued_tickets = 0;

	for (Passenger p : passengers) {
		if (p.age == age)
		{
			m_number_issued_tickets += p.hand_ticket;
		}
	}
```

```
	cout << "\nThe number of issued tickets: " << m_number_issued_tickets << endl;
}
void PassengerAIR::GetAllNumbersGender(char gender)
{
	int m_number_issued_tickets = 0;

	for (Passenger p : passengers) {
		if (p.gender == gender)
		{
			m_number_issued_tickets += p.hand_ticket;
		}
	}
	cout << "\nThe number of issued tickets: " << m_number_issued_tickets << endl;
}
void PassengerAIR::GetFreeReservedPlacesFlight(string flight)
{
	int m_number_free_places = 0;
	int m_number_reserved_seats = 0;
	vector<Passenger> p(passengers);
	cout << "Reserved seats: " << endl;
	for (Passenger p : passengers) {
		if (p.flight == flight)
		{
			for (int i = 1; i <= 8; i++)
			{
				for (int n = 1; n <= 9; n++)
				{
					if (i == p.place_row && n == p.place_number)
					{
						cout << i << ":" << n << "  ";
						m_number_reserved_seats++;
					}
				}
			}
		}
	}
	cout << "\nFree places: " << endl;
	for (int i = 1; i <= 8; i++) {
		for (int n = 1; n <= 9; n++) {
			bool found = false;
			for (Passenger p : passengers) {
				if (p.flight == flight && i == p.place_row && n ==
p.place_number) {
					found = true;
					break;
				}
			}
			if (!found) {
				cout << i << ":" << n << "  ";
				m_number_free_places++;
			}
		}
	}

	cout << "\nThe number of reserved seats: " << m_number_reserved_seats << endl;
	cout << "\nThe number of free places: " << m_number_free_places << endl;
}
void PassengerAIR::GetFreeReservedPlacesDepartureDay(string departure_day)
{
	int m_number_free_places = 0;
	int m_number_reserved_seats = 0;
	vector<Passenger> p(passengers);
	cout << "Reserved seats: " << endl;
	for (Passenger p : passengers) {
		if (p.departure_day == departure_day)
		{
			for (int i = 1; i <= 8; i++)
			{
```

```
                        for (int n = 1; n <= 9; n++)
                        {
                                if (i == p.place_row && n == p.place_number)
                                {
                                        cout << i << ":" << n << "  ";
                                        m_number_reserved_seats++;
                                }
                        }
                }
        }
        cout << "\nFree places: " << endl;
        for (int i = 1; i <= 8; i++) {
                for (int n = 1; n <= 9; n++) {
                        bool found = false;
                        for (Passenger p : passengers) {
                                if (p.departure_day == departure_day && i == p.place_row && n ==
p.place_number) {
                                        found = true;
                                        break;
                                }
                        }
                        if (!found) {
                                cout << i << ":" << n << "  ";
                                m_number_free_places++;
                        }
                }
        }

        cout << "\nThe number of reserved seats: " << m_number_reserved_seats << endl;
        cout << "\nThe number of free places: " << m_number_free_places << endl;
}
void PassengerAIR::GetFreeReservedPlacesTicketPrice(int ticket_price)
{
        int m_number_free_places = 0;
        int m_number_reserved_seats = 0;
        vector<Passenger> p(passengers);
        cout << "Reserved seats: " << endl;
        for (Passenger p : passengers) {
                if (p.ticket_price == ticket_price)
                {
                        for (int i = 1; i <= 8; i++)
                        {
                                for (int n = 1; n <= 9; n++)
                                {
                                        if (i == p.place_row && n == p.place_number)
                                        {
                                                cout << i << ":" << n << "  ";
                                                m_number_reserved_seats++;
                                        }
                                }
                        }
                }
        }
        cout << "\nFree places: " << endl;
        for (int i = 1; i <= 8; i++) {
                for (int n = 1; n <= 9; n++) {
                        bool found = false;
                        for (Passenger p : passengers) {
                                if (p.ticket_price == ticket_price && i == p.place_row && n ==
p.place_number) {
                                        found = true;
                                        break;
                                }
                        }
                        if (!found) {
                                cout << i << ":" << n << "  ";
                                m_number_free_places++;
```

```
                        }
                }
        }

        cout << "\nThe number of reserved seats: " << m_number_reserved_seats << endl;
        cout << "\nThe number of free places: " << m_number_free_places << endl;
}
void PassengerAIR::GetFreeReservedPlacesDepartureTime(string departure_time)
{
        int m_number_free_places = 0;
        int m_number_reserved_seats = 0;
        vector<Passenger> p(passengers);
        cout << "Reserved seats: " << endl;
        for (Passenger p : passengers) {
                if (p.departure_time == departure_time)
                {
                        for (int i = 1; i <= 8; i++)
                        {
                                for (int n = 1; n <= 9; n++)
                                {
                                        if (i == p.place_row && n == p.place_number)
                                        {
                                                cout << i << ":" << n << "  ";
                                                m_number_reserved_seats++;
                                        }
                                }
                        }
                }
        }
        cout << "\nFree places: " << endl;
        for (int i = 1; i <= 8; i++) {
                for (int n = 1; n <= 9; n++) {
                        bool found = false;
                        for (Passenger p : passengers) {
                                if (p.departure_time == departure_time && i == p.place_row && n
== p.place_number) {
                                        found = true;
                                        break;
                                }
                        }
                        if (!found) {
                                cout << i << ":" << n << "  ";
                                m_number_free_places++;
                        }
                }
        }

        cout << "\nThe number of reserved seats: " << m_number_reserved_seats << endl;
        cout << "\nThe number of free places: " << m_number_free_places << endl;
}
};
```

Додаток Б

Результат виконання програми

```
Select an option:
1. List of employees
2. Employees (serve a certain flight in divisions, age, average salary)
3. Pilots (passed medical examination: filter by years, sex, age, salary)
4. Planes (by time, arrival at the airport, number of flights)
5. State of the aircraft (service period, age of the aircraft, number of repairs, number of flights before repair)
6. Flights (according to the specified route, flight duration, ticket price, all these criteria)
7. Canceled flights (in the specified direction, on the specified route, by the number unclaimed seats, the percentage unclaimed seats)
8. Delayed flights (indicating the reason for the specified route, the number of issued tickets for the time of delay)
9. List, number of flights (aircraft of this type, average number of tickets sold, flight duration, ticket price, departure time)
10. List, number of flights (of the specified category, on a certain direction, with the specified type of aircraft)
11. Passengers (of a certain flight, that departed on the specified day, flew abroad on this day, by baggage registration, gender, age)
12. List, number of free and reserved seats(the specified flight, specified day, according to the specified route, price, at the time of departure)
13. Number of tickets (for a specific flight, on a specified day, on a specific route, by ticket price, by age, gender)
14. Exit
```

Пункт 1

```
Select a request:
1. Get a list of all employees and their number
2. Get a list of department heads
3. Get a list of employees of the department
4. Get a list of employees with experience
5. Get a list of employees with sexual characteristic
6. Get a list of employees with age
7. Get the number of children of an employee
8. Get the amount of the employee's salary
9. Exit
```

1)

```
Name: Sofia
Age: 19
Department: 6
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 1750


Number of employees: 24
```

2)

```
Name: Vika
Age: 20
Department: 6
Is department head: Yes
Brigade: 0
The flight it serves: NOFounded
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 2000


Number of employees: 6
```

3)

```
1.Pilots
2.Dispatchers
3.Technigues
4.Cashieres
5.Security
6.Directory service
Enter department (1-6):
```

```
Name: Smith
Age: 30
Department: 1
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 5
Gender: F
Has children: Yes
Number of children: 1
Salary: 2500


Number of employees: 3
```

```
Name: yura
Age: 46
Department: 3
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 7
Gender: M
Has children: No
Number of children: 0
Salary: 1700


Number of employees: 2
```

4)

```
Name: Sofia
Age: 19
Department: 6
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 1750


Number of employees: 8
```

5)

```
Enter age:48
Name: yura
Age: 48
Department: 3
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 7
Gender: M
Has children: No
Number of children: 0
Salary: 1700


Number of employees: 1
```

6)

```
Enter number children:2
Name: Andrew
Age: 34
Department: 3
Is department head: Yes
Brigade: 0
The flight it serves: NOFounded
Experience (years): 5
Gender: M
Has children: Yes
Number of children: 2
Salary: 2000


Number of employees: 1
```

7)

```
Name: yura
Age: 48
Department: 3
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 7
Gender: M
Has children: No
Number of children: 0
Salary: 1700


Number of employees: 2
```

8)

Пункт 2

```
Select a request:
1.Get workers in the brigade
2.Get workers in oll department
3.Get workers in the department
4.Get a list of employees by serviced flight
5.Get a list of employees with age
6.Get a list of employees total average salary in brigade
7. Exit
```

```
Name: Sofia
Age: 19
Department: 6
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 1750


Number of employees: 18
```

1)

```
Name: Sofia
Age: 19
Department: 6
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 1750


Number of employees: 18
```

2)

```
1.Pilots
2.Dispatchers
3.Technigues
4.Cashieres
5.Security
6.Directory service
Enter department (1-6):
```

3)

```
Name: Smith
Age: 30
Department: 1
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 7
Gender: F
Has children: Yes
Number of children: 1
Salary: 2500


Number of employees: 3
```

```
Name: Sofia
Age: 19
Department: 6
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 1750


Number of employees: 18
```

4)

```
Enter age:48
Name: yura
Age: 48
Department: 3
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 7
Gender: M
Has children: No
Number of children: 0
Salary: 1700


Number of employees: 1
```

5)

```
Name: Sofia
Age: 19
Department: 6
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: No
Number of children: 0
Salary: 1750


Number of employees: 18
Total average salary in brigade: 1772
```

6)

Пункт 3

```
Select a request:
1.Get a list of pilots who have passed a medical examination
2.Get a list of pilots who did not pass it in the specified year
3.Get a list of pilots with sexual characteristic
4.Get a list of pilots with age
5.Get the amount of pilots salary
6. Exit
```

1)
```
Name: Andriana
Age: 30
Department: 4
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 1
Gender: F
Has children: Yes
Number of children: 1
Salary: 1750


Number of employees: 2
Press any key to continue . . .
```

2)
```
Enter year: 2023
Name: Natalya
Age: 30
Department: 1
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 5
Gender: F
Has children: Yes
Number of children: 1
Salary: 2500


Number of employees: 1
```

3)
```
Enter sexual characteristic(M or F):M
Name: Yura
Age: 18
Department: 1
Is department head: Yes
Brigade: 0
The flight it serves: NOFounded
Experience (years): 3
Gender: M
Has children: No
Number of children: 0
Salary: 20000


Number of employees: 1
```

4)
```
Enter age:31
Name: Smith
Age: 31
Department: 1
Is department head: No
Brigade: 1
The flight it serves: Ukraine-Poland
Experience (years): 7
Gender: F
Has children: Yes
Number of children: 1
Salary: 2500


Number of employees: 1
```

5)
```
Enter the salary amount:20000
Name: Yura
Age: 18
Department: 1
Is department head: Yes
Brigade: 0
The flight it serves: NOFounded
Experience (years): 3
Gender: M
Has children: No
Number of children: 0
Salary: 20000


Number of employees: 1
```

Пункт 4

```
Select a request:
1.Get a list of all plane and their number
2.Get a list of all the planes that are at the airport at the specified time
3.Get a list and the total number of planes by arrival time at the airport
4.Get a list and the total number of planes by the number of flights made.
5. Exit
```

```
Type: U380
Age of the aircraft: 12
Flight: Ivano-Frankivsk-Lviv
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 9:00
Number of flights made: 34
Passed technical inspection: No
Period of time for technical inspection: NotPass
Sent for repair: No
Day sent for repair: N/A
Number of repairs: 0
Number of flights repaired: 0


Number of planes: 3
```

1)

```
Enter time: 12:40
Type: A380
Age of the aircraft: 12
Flight: Ukraine-Poland
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 12:40
Number of flights made: 58
Passed technical inspection: Yes
Period of time for technical inspection: morning
Sent for repair: Yes
Day sent for repair: 05.05.2023
Number of repairs: 1
Number of flights repaired: 58


Number of planes: 1
```

2)

```
Type: U380
Age of the aircraft: 12
Flight: Ivano-Frankivsk-Lviv
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 9:00
Number of flights made: 34
Passed technical inspection: No
Period of time for technical inspection: NotPass
Sent for repair: No
Day sent for repair: N/A
Number of repairs: 0
Number of flights repaired: 0


Number of planes: 2
```

3)

```
Enter number flights made: 58
Type: A380
Age of the aircraft: 12
Flight: Ukraine-Poland
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 12:40
Number of flights made: 58
Passed technical inspection: Yes
Period of time for technical inspection: morning
Sent for repair: Yes
Day sent for repair: 05.05.2023
Number of repairs: 1
Number of flights repaired: 58


Number of planes: 1
```

4)

Пункт 5

```
Select a request:
1.Get a list and the total number of aircraft that have passed technical inspection for a certain period of time
2.Get a list and the total number of aircraft sent for repair at the specified time
3.Get a list and the total number of aircraft repaired a given number of times
4.Get a list and the total number of aircraft according to the number of flights before repair
5.Get a list and the total number of aircraft by age of the aircraft.
6. Exit
```

```
Enter a time period(morning, noon, evening, night): morning
Type: A380
Age of the aircraft: 12
Flight: Ukraine-Poland
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 12:40
Number of flights made: 58
Passed technical inspection: Yes
Period of time for technical inspection: morning
Sent for repair: Yes
Day sent for repair: 05.05.2023
Number of repairs: 1
Number of flights repaired: 58


Number of planes: 1
```

1)

```
Enter day sending repair: 05.05.2023
Type: A380
Age of the aircraft: 12
Flight: Ukraine-Poland
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 12:40
Number of flights made: 58
Passed technical inspection: Yes
Period of time for technical inspection: morning
Sent for repair: Yes
Day sent for repair: 05.05.2023
Number of repairs: 1
Number of flights repaired: 58


Number of planes: 1
```

2)

```
Enter number repairs: 1
Type: A380
Age of the aircraft: 12
Flight: Ukraine-Poland
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 12:40
Number of flights made: 58
Passed technical inspection: Yes
Period of time for technical inspection: morning
Sent for repair: Yes
Day sent for repair: 05.05.2023
Number of repairs: 1
Number of flights repaired: 58


Number of planes: 1
```

3)

```
Enter number flights repaired: 58
Type: A380
Age of the aircraft: 12
Flight: Ukraine-Poland
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 12:40
Number of flights made: 58
Passed technical inspection: Yes
Period of time for technical inspection: morning
Sent for repair: Yes
Day sent for repair: 05.05.2023
Number of repairs: 1
Number of flights repaired: 58


Number of planes: 1
```

4)

```
Type: U380
Age of the aircraft: 12
Flight: Ivano-Frankivsk-Lviv
Number of seats: 72
Time spent at airport: 3
Time of arrival at airport: 9:00
Number of flights made: 34
Passed technical inspection: No
Period of time for technical inspection: NotPass
Sent for repair: No
Day sent for repair: N/A
Number of repairs: 0
Number of flights repaired: 0


Number of planes: 3
```

5)

Пункт 6

```
Select a request:
1.Get a list of flights on the specified route
2.Get a list of flights by flight duration
3.Get a list of flights by ticket price
4.Get a list of flights by all criteria at once
5. Exit
```

```
Enter route: Ukraine-Poland
Route: Ukraine-Poland
Number: 72
Flight time: 2
Ticket price: 50
Flight cancellation: No
Direction: Europe
Sold ticket: 10
Number of issued tickets during delay: 2
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 12:00
Flight categories: International
Number of passengers: 8
Departure day: 10/05/2023


Number of flights: 1
```

1)

```
Enter flight time: 1
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 17:00
Flight categories: Internal
Number of passengers: 25
Departure day: 10/05/2023


Number of flights: 1
```

2)

```
Enter ticket price: 50
Route: Ukraine-Poland
Number: 72
Flight time: 2
Ticket price: 50
Flight cancellation: No
Direction: Europe
Sold ticket: 10
Number of issued tickets during delay: 2
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 12:00
Flight categories: International
Number of passengers: 8
Departure day: 10/05/2023
```

3)
```
Number of flights: 1
```

```
Enter route: Ukraine-Poland
Enter flight time: 2
Enter ticket price: 50
Route: Ukraine-Poland
Number: 72
Flight time: 2
Ticket price: 50
Flight cancellation: No
Direction: Europe
Sold ticket: 10
Number of issued tickets during delay: 2
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 12:00
Flight categories: International
Number of passengers: 8
Departure day: 10/05/2023
```

4)
```
Number of flights: 1
```

Пункт 7

```
Select a request:
1.Get the list and the total number of canceled flights in full
2.Get a list and the total number of canceled flights at the indicated direction
3.Get the list and total number of canceled flights on the specified route
4.Get a list and the total number of canceled flights by the number of unclaimed seats
5.Get a list and the total number of canceled flights in full by the percentage of unclaimed seats
6. Exit
```

```
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 17:00
Flight categories: Internal
Number of passengers: 25
Departure day: 10/05/2023
```

1)
```
Number of flights: 2
```

```
Enter direction: America
Route: Ukraine-America
Number: 72
Flight time: 4
Ticket price: 150
Flight cancellation: Yes
Direction: America
Sold ticket: 24
Number of issued tickets during delay: 0
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 22:00
Flight categories: International
Number of passengers: 24
Departure day: 10/05/2023
```

2)
```
Number of flights: 1
```

```
Enter route: Ukraine-America
Route: Ukraine-America
Number: 72
Flight time: 4
Ticket price: 150
Flight cancellation: Yes
Direction: America
Sold ticket: 24
Number of issued tickets during delay: 0
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 22:00
Flight categories: International
Number of passengers: 24
Departure day: 10/05/2023


Number of flights: 1
```

3)

```
Enter seats: 48
Route: Ukraine-America
Number: 72
Flight time: 4
Ticket price: 150
Flight cancellation: Yes
Direction: America
Sold ticket: 24
Number of issued tickets during delay: 0
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 22:00
Flight categories: International
Number of passengers: 24
Departure day: 10/05/2023


Number of flights: 1
```

4)

Пункт 8

```
Select a request:
1.Get the full list and total number of delayed flights
2.Get the list and total number of delayed flights fromthe specified reason
3.Get a list and the total number of delayed flights on the specified route
4.Get a list and the total number of delayed flights for the number of issued tickets during the delay.
5. Exit
```

```
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A380
Delayed flight: Yes
Reason for flight delay: N/A
Departure time: 17:00
Flight categories: Internal
Number of passengers: 36
Departure day: 10/05/2023


Number of flights: 1
```

1)

```
Enter reason:
Number of flights: 0
```

2)

```
Enter route: Ivano-Frankivsk-Lviv
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A380
Delayed flight: Yes
Reason for flight delay: Wind
Departure time: 17:00
Flight categories: Internal
Number of passengers: 36
Departure day: 10/05/2023


Number of flights: 1
```

3)

```
Enter the number of tickets issued during the delay. 45
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A380
Delayed flight: Yes
Reason for flight delay: Wind
Departure time: 17:00
Flight categories: Internal
Number of passengers: 36
Departure day: 10/05/2023


Number of flights: 1
```

4)

Пункт 9

```
Select a request:
1.Get a list and the total number of flights operated by aircraft of a given type
2.Get a list and the total number of flights by flight duration
3.Get a list of the total number of flights by ticket price
4.Get the list and total number of flights by departure time
5. Exit
```

```
Enter type: A390
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A390
Delayed flight: Yes
Reason for flight delay: Wind
Departure time: 17:00
Flight categories: Internal
Number of passengers: 36
Departure day: 10/05/2023


Number of flights: 1
```

1)

```
Enter flight time(in hourse): 1
Route: Ivano-Frankivsk-Lviv
Number: 72
Flight time: 1
Ticket price: 10
Flight cancellation: Yes
Direction: Ukraine
Sold ticket: 70
Number of issued tickets during delay: 45
Type: A390
Delayed flight: Yes
Reason for flight delay: Wind
Departure time: 17:00
Flight categories: Internal
Number of passengers: 36
Departure day: 10/05/2023


Number of flights: 1
```

2)

```
Enter ticket price: 50
Route: Ukraine-Poland
Number: 72
Flight time: 2
Ticket price: 50
Flight cancellation: No
Direction: Europe
Sold ticket: 10
Number of issued tickets during delay: 2
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 12:00
Flight categories: International
Number of passengers: 8
Departure day: 10/05/2023


Number of flights: 1
```

3)

```
Enter departure time: 22:00
Route: Ukraine-America
Number: 72
Flight time: 4
Ticket price: 150
Flight cancellation: Yes
Direction: America
Sold ticket: 24
Number of issued tickets during delay: 0
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 22:00
Flight categories: International
Number of passengers: 24
Departure day: 10/05/2023


Number of flights: 1
```

4)

Пункт 10

```
Select a request:
1.Get the list and total number of flights of the specified category
2.Get a list and the total number of flights in a certain direction
3. Exit
```

```
Route: Ukraine-America
Number: 72
Flight time: 4
Ticket price: 150
Flight cancellation: Yes
Direction: America
Sold ticket: 24
Number of issued tickets during delay: 0
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 22:00
Flight categories: International
Number of passengers: 24
Departure day: 10/05/2023


Number of flights: 2
```

1)

```
Enter direction: Europe
Route: Ukraine-Poland
Number: 72
Flight time: 2
Ticket price: 50
Flight cancellation: No
Direction: Europe
Sold ticket: 10
Number of issued tickets during delay: 2
Type: A380
Delayed flight: No
Reason for flight delay: N/A
Departure time: 12:00
Flight categories: International
Number of passengers: 8
Departure day: 10/05/2023


Number of flights: 1
```

2)

Пункт 11

```
Select a request:
1.Get a list and the total number of passengers on this flight who departed on the specified day
2.Get a list and the total number of passengers on this flight who flew abroad on the specified day
3.Get a list and the total number of passengers on this flight, according to the indication of the baggage claim
4.Get the list and total number of passengers on this flight, by gender
5.Get the list and total number of passengers on this flight, by age.
6. Exit
```

```
Name: Viola
Age: 32
Gender: F
Flight: Ukraine-Poland
Departure day: 15.05.2023
Flew abroad: No
Checked luggage: Yes
Place row: 4
Place number: 5
Ticket price : 60
Departure time: 15:00


Number of passengers: 9
```

1)

```
Name: Yura
Age: 19
Gender: M
Flight: Ukraine-Poland
Departure day: 15.05.2023
Flew abroad: Yes
Checked luggage: Yes
Place row: 1
Place number: 8
Ticket price : 50
Departure time: 13:00


Number of passengers: 6
```

2)

```
Name: Viola
Age: 32
Gender: F
Flight: Ukraine-Poland
Departure day: 15.05.2023
Flew abroad: No
Checked luggage: Yes
Place row: 4
Place number: 5
Ticket price : 60
Departure time: 15:00


Number of passengers: 9
```

3)

```
Name: Viola
Age: 32
Gender: F
Flight: Ukraine-Poland
Departure day: 15.05.2023
Flew abroad: No
Checked luggage: Yes
Place row: 4
Place number: 5
Ticket price : 60
Departure time: 15:00


Number of passengers: 2
```

4)

```
Name: Yura
Age: 19
Gender: M
Flight: Ukraine-Poland
Departure day: 15.05.2023
Flew abroad: Yes
Checked luggage: Yes
Place row: 1
Place number: 8
Ticket price : 50
Departure time: 13:00


Number of passengers: 2
```

5)

Пункт 12

```
Select a request:
1.Get a list and the total number of free and reserved seats on the specified flight
2.Get a list and the total number of free and reserved seats for the specified day
3.Get a list and the total number of free and reserved seats according to the specified route
4.Get a list and the total number of free and reserved seats by price
5.Get a list and the total number of free and reserved seats by departure time.
6. Exit
```

```
Enter flight: Ukraine-Poland
Reserved seats:
2:4  1:1  1:4  6:3  3:9  1:8  4:5
Free places:
1:2  1:3  1:5  1:6  1:7  1:9  2:1  2:2  2:3  2:5  2:6  2:7  2:8  2:9  3:1  3:2  3:3  3:4  3:5  3:6  3:7  3:8  4:1  4:2
4:3  4:4  4:6  4:7  4:8  4:9  5:1  5:2  5:3  5:4  5:5  5:6  5:7  5:8  5:9  6:1  6:2  6:4  6:5  6:6  6:7  6:8  6:9  7:1
7:2  7:3  7:4  7:5  7:6  7:7  7:8  7:9  8:1  8:2  8:3  8:4  8:5  8:6  8:7  8:8  8:9
The number of reserved seats: 7

The number of free places: 65
```

1)

```
Enter departure day: 15.05.2023
Reserved seats:
2:4  1:1  1:4  6:3  3:9  1:8  4:5
Free places:
1:2  1:3  1:5  1:6  1:7  1:9  2:1  2:2  2:3  2:5  2:6  2:7  2:8  2:9  3:1  3:2  3:3  3:4  3:5  3:6  3:7  3:8  4:1  4:2
4:3  4:4  4:6  4:7  4:8  4:9  5:1  5:2  5:3  5:4  5:5  5:6  5:7  5:8  5:9  6:1  6:2  6:4  6:5  6:6  6:7  6:8  6:9  7:1
7:2  7:3  7:4  7:5  7:6  7:7  7:8  7:9  8:1  8:2  8:3  8:4  8:5  8:6  8:7  8:8  8:9
The number of reserved seats: 7

The number of free places: 65
```

2)

```
Enter flight: Europe
Reserved seats:

Free places:
1:1  1:2  1:3  1:4  1:5  1:6  1:7  1:8  1:9  2:1  2:2  2:3  2:4  2:5  2:6  2:7  2:8  2:9  3:1  3:2  3:3  3:4  3:5  3:6
3:7  3:8  3:9  4:1  4:2  4:3  4:4  4:5  4:6  4:7  4:8  4:9  5:1  5:2  5:3  5:4  5:5  5:6  5:7  5:8  5:9  6:1  6:2  6:3
6:4  6:5  6:6  6:7  6:8  6:9  7:1  7:2  7:3  7:4  7:5  7:6  7:7  7:8  7:9  8:1  8:2  8:3  8:4  8:5  8:6  8:7  8:8  8:9
The number of reserved seats: 0

The number of free places: 72
```

3)

```
Enter ticket price: 60
Reserved seats:
1:4  4:5
Free places:
1:1  1:2  1:3  1:5  1:6  1:7  1:8  1:9  2:1  2:2  2:3  2:4  2:5  2:6  2:7  2:8  2:9  3:1  3:2  3:3  3:4  3:5  3:6  3:7
3:8  3:9  4:1  4:2  4:3  4:4  4:6  4:7  4:8  4:9  5:1  5:2  5:3  5:4  5:5  5:6  5:7  5:8  5:9  6:1  6:2  6:3  6:4  6:5
6:6  6:7  6:8  6:9  7:1  7:2  7:3  7:4  7:5  7:6  7:7  7:8  7:9  8:1  8:2  8:3  8:4  8:5  8:6  8:7  8:8  8:9
The number of reserved seats: 2

The number of free places: 70
```

4)

```
Enter departure time: 13:00
Reserved seats:
1:1  1:4  6:3  3:9  1:8
Free places:
1:2  1:3  1:5  1:6  1:7  1:9  2:1  2:2  2:3  2:4  2:5  2:6  2:7  2:8  2:9  3:1  3:2  3:3  3:4  3:5  3:6  3:7  3:8  4:1
4:2  4:3  4:4  4:5  4:6  4:7  4:8  4:9  5:1  5:2  5:3  5:4  5:5  5:6  5:7  5:8  5:9  6:1  6:2  6:4  6:5  6:6  6:7  6:8
6:9  7:1  7:2  7:3  7:4  7:5  7:6  7:7  7:8  7:9  8:1  8:2  8:3  8:4  8:5  8:6  8:7  8:8  8:9
The number of reserved seats: 5
```

5)
```
The number of free places: 67
```

Пункт 13

```
Select a request:
1.Get the total number of issued tickets for a certain flight
2.Get the total number of issued tickets on the specified day
3.Get the total number of given tickets for a specific route
4.Get the total number of issued tickets by ticket price
5.Get the total number of issued tickets by age
6.Get the total number of issued tickets for a certain flight, by gender
7. Exit
```

1)
```
Enter flights: Ukraine-Poland

The number of issued tickets: 2
```

2)
```
Enter day(DD.MM.YYYY): 10/05/2023

The number of issued tickets: 47
```

3)
```
Enter direction: Ukraine

The number of issued tickets: 45
```

4)
```
Enter ticket price: 50

The number of issued tickets: 2
```

5)
```
Enter age: 32

The number of issued tickets: 1
```

6)
```
Enter gender (M/F): M

The number of issued tickets: 2
```