

Міністерство освіти і науки України Харківський
національний університет радіоелектроніки

Факультет

Комп'ютерних наук

Кафедра

Системотехніки

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни Об'єктно-орієнтоване програмування
(назва дисципліни)

тема Розробка об'єктно-орієнтованої програмної системи «Автобаза»

Керівник

доцент кафедри СТ, Білова Т.Г.

(підпис, дата, посада, прізвище, ініціали)

Студент

КНТ-21-5 Крутипорох Р.О.

(група, підпис, дата, прізвище, ініціали)

Робота захищена з оцінкою «_____»
«__»_____2022р.

Комісія:

доцент каф. СТ, с.н.с Коваленко А.І.

(підпис, дата, посада, прізвище, ініціали)

доцент каф. СТ, доц. Білова Т.Г.

(підпис, дата, посада, прізвище, ініціали)

доцент каф. СТ, Ситнікова П. Е.

(підпис, дата, посада, прізвище, ініціали)

Харків 2022

Факультет комп'ютерних наук

Кафедра системотехніки

Освітньо-професійна програма Комп'ютерні науки та технології

Курс 1 група КНТ-21-5 семестр 2

ЗАВДАННЯ

на курсову роботу

студенту Крутипороху Руслану Олеговичу

1. Тема роботи: Розробка об'єктно-орієнтованої програмної системи «Автобаза»

2. Термін здачі студентом закінченої роботи 13.06.2022

3. Вихідні дані до проекту: Розробити об'єктно-орієнтовану програмну систему «Автобаза» з ієрархією класів та підтримкою поліморфізму мовою програмування C++. Програмна система виконує функції, визначені у варіанті завдання: Визначити базовий клас АВТОМОБІЛЬ (державний номер автомобіля, марка автомобіля, вантажопідйомність, норма витрат пального, місце приписки) та ВОДІЙ (прізвище та ініціали водія, табельний номер, оклад). Використовуючи множинне успадкування, визначити похідний клас РЕЙС з додатковими полями даних: дата рейсу, кінцевий пункт рейсу, кілометраж, перевезено (тонн), розхід пального за рейс. Визначити конструктори, деструктор, методи встановлення та визначення значень полів даних. Операційна система – Windows 7 або вище, програмне забезпечення: інтегроване середовище MS Visual Studio, редактор UML-діаграм Software Ideas Modeler. Методичне забезпечення – методичні вказівки до курсової роботи.

4. Зміст пояснювальної записки (перелік питань, які підлягають розробці): провести аналіз предметної області та визначити сутності, об'єкти, їх атрибути та функції; сформулювати та оформити вимоги до програмної системи; розробити об'єктну модель предметної області та сформулювати словник; провести UML моделювання, розробити діаграму класів (Class Diagram); реалізувати мовою C++ спроектовану програмну систему та описати її; розробити інтерфейс користувача; виконати тестування розробленої програми; підготувати відповідно до ГОСТ 19.401-78 програмний документ «Текст програми».

5. Перелік графічного матеріалу: Схема об'єктної моделі, діаграма класів (Class Diagram), алгоритми, приклади екранних форм.

6. Дата видачі завдання: 05.04.2022

Керівник роботи

Білова Тетяна Георгіївна

Студент

Крутипорох Руслан Олегович

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсового проекту	Термін виконання	Примітка
1	Видача теми, узгодження і затвердження теми	05.04.2022	виконано
2	Аналіз предметної області	06.04.2022	виконано
3	Формулювання вимог до програми, розробка словника системи	13.04.2022	виконано
4	Розробка діаграми класів	15.04.2022	виконано
5	Розробка базового класу та похідних класів	20.04.2022	виконано
6	Розробка інтерфейсу програми у вигляді текстового меню.	27.04.2022	виконано
7	Розробка функцій-членів класів, що призначені для роботи та обробки полів даних класу АВТОМОБІЛЬ, ВОДІЙ, РЕЙС. Перевірка працездатності функцій.	03.05.2022	виконано
8	Розробка функцій-членів класів, що призначені для видалення /додавання елементів класу АВТОМОБІЛЬ, ВОДІЙ, РЕЙС. Перевірка працездатності функцій.	10.05.2022	виконано
9	Розробка функцій-членів класів, що призначені для сортування, пошуку, ведення /виведення елементів масиву даних типу РЕЙС. Перевірка працездатності функцій.	18.05.2022	виконано
10	Розробка головної функції main(). Перевірка працездатності системи.	25.05.2022	виконано
11	Тестування розробленого програмного забезпечення	04.06.2022	виконано
12	Підготовка пояснювальної записки та її додатків	13.06.2022	виконано

Керівник роботи _____ Білова Тетяна Георгіївна
(Підпис) (прізвище, ім'я, по батькові)

Студент _____ Крутипорох Руслан Олегович
(Підпис) (прізвище, ім'я, по батькові)

«_____» _____ 2022 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 51 с., 27 рис., 2 додатків, 5 джерел інформації.

КЛАСС, ПОЛІМОРФНИЙ КЛАСТЕР, ВІРТУАЛЬНА ФУНКЦІЯ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ, ПРОГРАМНА СИСТЕМА, ІНТЕРФЕЙС КОРИСТУВАЧА, АБСТРАКТНИЙ КЛАС, СИСТЕМА ОБРОБКИ ЗАМОВЛЕНЬ

Мета роботи – створення програмної системи «Автобаза» ,заснованої на основних методах об'єктно-орієнтованого програмування та методах автоматизації промислових систем.

Об'єкт розробки – об'єктна модель функціонування автобази.

Предмет розробки – інформаційні технології та програмні методи об'єктно-орієнтованої розробки програмних систем.

Методи розробки ґрунтуються на використанні інтегрованого середовища розробки середовища Microsoft Visual Studio 2019, мови програмування C++.

Проведено аналіз предметної області, визначені основні завдання системи, опис вхідної та вихідної інформації, розроблена ієрархічна класова модель системи, виконана діаграма класів, розроблено програмне забезпечення, що реалізує основні функції системи: можливість контролювати та редагувати об'єкти класів, вивід тестувальних функцій до користувача через окремий пункт меню.

Результатом розробки є програмна система "Автобаза", що може приймати та оформлювати замовлення від користувача-замовника для автобази з доставки будь-якого виду вантажу(сировина, мебель, техніка тощо) . Система розраховує кількість рейсів на одне замовлення, загальні витрати пального та створює звіти з логістичних рейсів в окремий текстовий файл.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	5
ВСТУП.....	6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	7
1.1 Аналіз предметної області	7
1.2 Постановка задачі	8
2 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ.....	10
2.1 Загальний опис діаграми класів	10
2.2 Опис класу-інтерфейсу «Menu»	11
2.3 Опис класу «Auto»	12
2.4 Опис класу «Driver».....	13
2.5 Опис похідного від класів «Driver» та «Auto» класу «Route»	14
2.6 Опис ієрархії класів-контейнерів: «Container», «SetofDrivers», «Timetable».....	15
2.7 Опис ієрархії класів-винятків «ErrorMessage», «ArrayMessage», «RangeMessage».....	16
2.8 Опис класу «Order».....	17
2.9 Опис класу «Manager».....	18
3 ПРОГРАМНА РЕАЛІЗАЦІЯ	19
3.1 Опис структури	19
3.2 Опис інтерфейсу	19
ВИСНОВКИ.....	32
ПОСИЛАННЯ.....	33
Додаток А.....	34
Додаток Б.....	36

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

UML – Unified Modeling Language, уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

ООП – Об'єктно-орієнтоване програмування

Гетер - спеціальний метод класу, що дозволяє отримувати атрибути з private сектору.

Сетер- спеціальний метод класу, що дозволяє встановлювати значення атрибутів з private сектору

ВСТУП

В останні роки автотранспортні підприємства в сучасних умовах жорстокої конкуренції потребують інноваційних рішень. Автобаза – невід’ємне підприємство кожного сучасного бізнесу, що побудований на логістиці. Автобази бувають двох категорій – військові та промислові. В курсовій роботі розглянена розробка програмної системи саме промислової автобази. В ході виконання курсової роботи була виконана диджиталізація всіх бухгалтерських та менеджерських процесів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Завданням курсової роботи є побудова програмного забезпечення, що диджиталізує відношення клієнт-автобаза. При розгляданні теми та завдання більш детально, було виділено наступні сутності (малюнок 1.1).

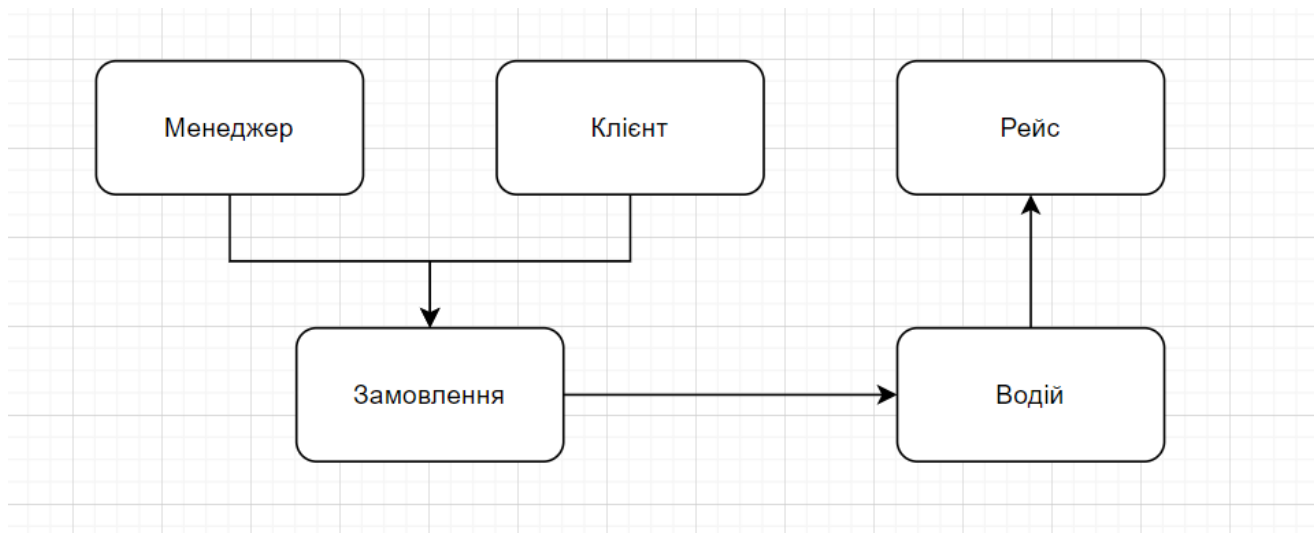


Рисунок 1.1 – Об'єктна модель програмної системи

Таке відношення моделей демонструє роботу програми. Менеджер та клієнт впливають на сутність замовлення, тим часом, замовлення має вплив на водія, водій на рейс.

Словник програмної системи:

Автобаза – автотранспортне підприємство, що забезпечує перевезення вантажів, а також технічне обслуговування та ремонт автомобілів.

Логістика(не входить до програмної системи, визначається як абстрактна сутність) – наука про оптимальне управління матеріальними, інформаційними та фінансовими потоками в економічних адаптивних системах із синергічними зв'язками.

Автомобіль – самохідна колісна машина, яка приводиться в рух

встановленим у неї двигуном і призначена для перевезення людей, вантажу, буксування транспортних засобів, виконання спеціальних робіт та перевезення спеціального устаткування безрейковими дорогами.

Водій – особа, що керує транспортним засобом.

Рейс – шлях транспортного засобу за визначеним маршрутом.

Розклад рейсів – графік, або таблиця, що містить відомості про час, місце та послідовність виконання рейсу.

Замовлення – доручення виготовити, виконати, підготувати або доставити що-небудь у певний, наперед визначений строк.

Менеджер – це людина, що має планувати, організовувати, координувати та контролювати певні процеси на підприємстві.

Після аналізу предметної області було створено наступну діаграму сутностей. (малюнок 1.2)

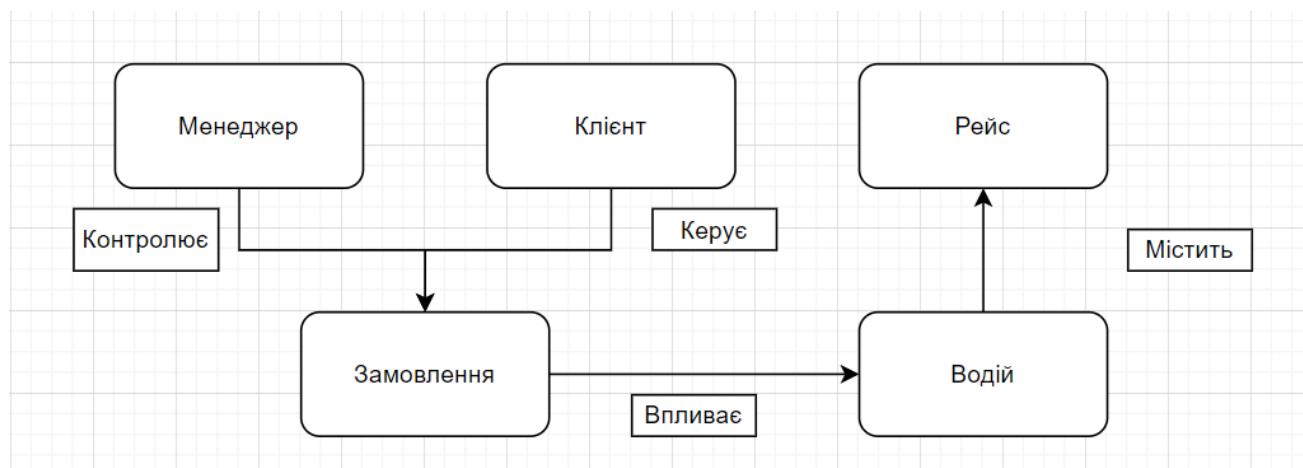


Рисунок 1.2 – Результуюча модель предметної області

1.2 Постановка задачі

Варіант №17 — Вибрано «Автобаза». Необхідно створити програмне забезпечення для диджиталізації відношення клієнт-автобаза. Програма повинна дозволяти користувачеві вводити дані, обробляти їх, виводити дані на екран. Завдання необхідно виконати за допомогою С++ - мова програмування з використанням об'єктно-орієнтованого програмування.

Для виконання роботи необхідно враховувати теоретичні відомості щодо обсягу програми. Створення кількох класів для опису об'єктів, методів маніпулювання даними, реалізації успадкування, перевантаження операцій. Під час виконання курсової роботи розроблено об'єктно-орієнтовану програмну систему «Автобаза». Тема цієї роботи збігається з темою курсу завдання.

Розроблена система має такі поняття: загальний об'єм перевезень, звіт з логістичних рейсів, вантажопідйомність авто, панель менеджера. Під час розробки програмного забезпечення були дотримані наступні вимоги:

- розробити класи «АВТОМОБІЛЬ», «ВОДІЙ», «ЗАМОВЛЕННЯ», «МЕНЕДЖЕР»;
- визначити конструктори ініціалізації, деструктори та методи для зміни та читання значень полів цього класу;
- перевантажити операції: () встановлення полів даних, операцію присвоєння об'єктів =, потокові операції введення >>, та виведення << об'єктів;
- використовуючи множинне успадкування , визначити похідний клас «РЕЙС» з додатковими полями даних;
- розробити клас «РОЗКЛАД» перевезень , який містить масив об'єктів класу «РЕЙС»;
- визначити загальний об'єм перевезень та загальні витрати пального.

При виборі архітектурного стилю перевагу віддали стилю консолі, оскільки на його розробку потрібно менше часу та ресурсів, тому він дозволяє зосередитися виключно на теоретичній частині розробки такого об'єктно-орієнтованого програмного середовища, а також покращити навички ООП, розробивши цю програму.

2 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальний опис діаграми класів

Для реалізації завдання курсової роботи були створені такі класи: «АВТОМОБІЛЬ», «ВОДІЙ», «МЕНЮ», «МЕНЕДЖЕР», «ЗАМОВЛЕННЯ», клас-контейнер «КОНТЕЙНЕР», клас-контейнер «РОЗКЛАД РЕЙСІВ», клас-контейнер «МНОЖИНА ВОДІЇВ», «РЕЙС», клас-виключення «ВИКЛЮЧЕННЯ», клас-виключення «НЕДОСТУПНИЙ ІНДЕКС», клас-виключення «ЗАБАГАТО ЕЛЕМЕНТІВ У МАСИВІ». Створено власну ієрархію виключень.

Загальна діаграма класів показана на рисунку 2.1.

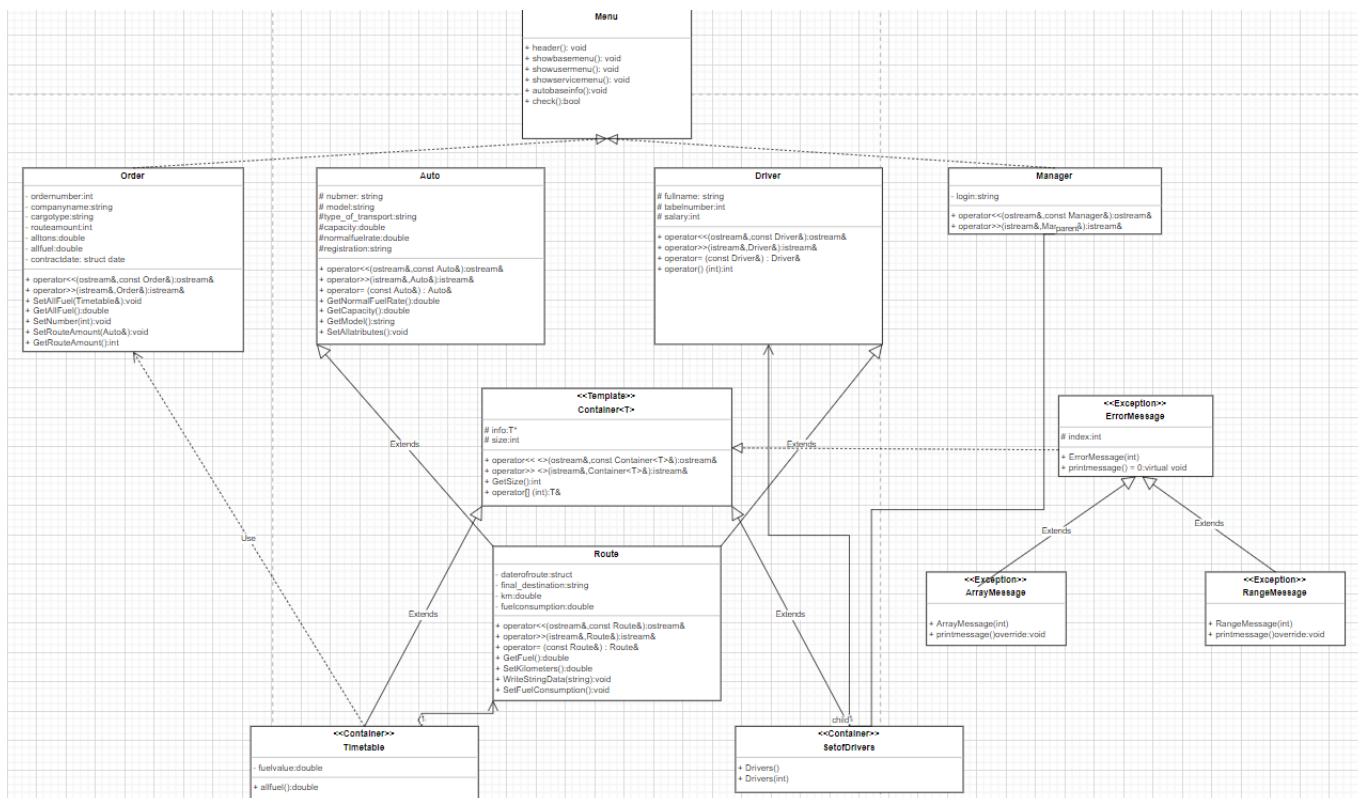


Рисунок 2.1 - Загальна діаграма класів

2.2 Опис класу-інтерфейсу «Menu»

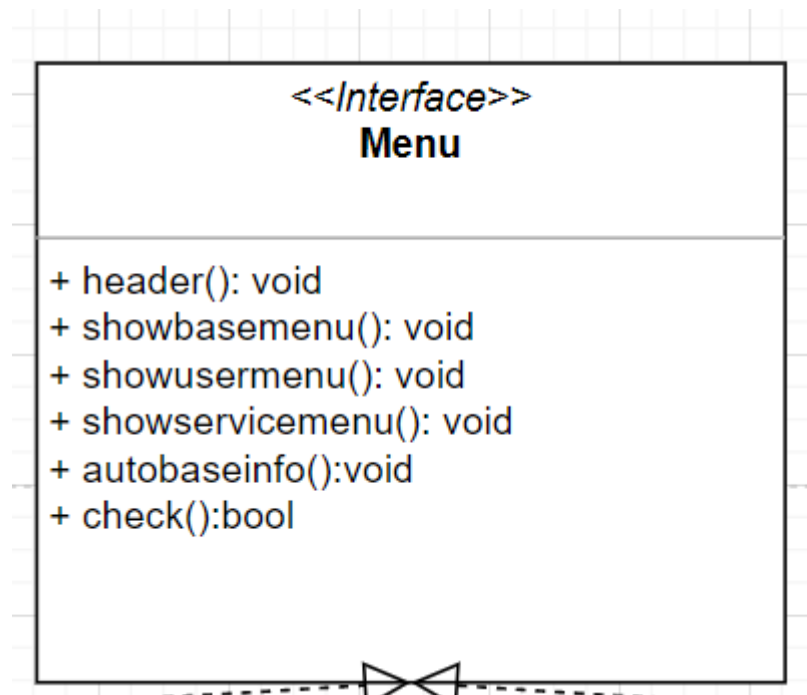


Рисунок 2.2 - Схема класу «Меню»

Клас-інтерфейс створений для відображення головного меню всієї програмної системи, містить в собі public методи, 5 – void, 1 – bool:

- **header():** відображає в консолі дизайн стартового меню;
- **showbasemenu():** відображає головне меню програмної системи;
- **showusermenu():** відображає меню користувача-замовника;
- **showservicemenu():** відображає меню менеджера;
- **autobaseinfo():** відображає загальну інформацію про автобазу «Express»;
- **check():** булевий метод, що перевіряє авторизацію менеджера в системі.

2.3 Опис класу «Auto»

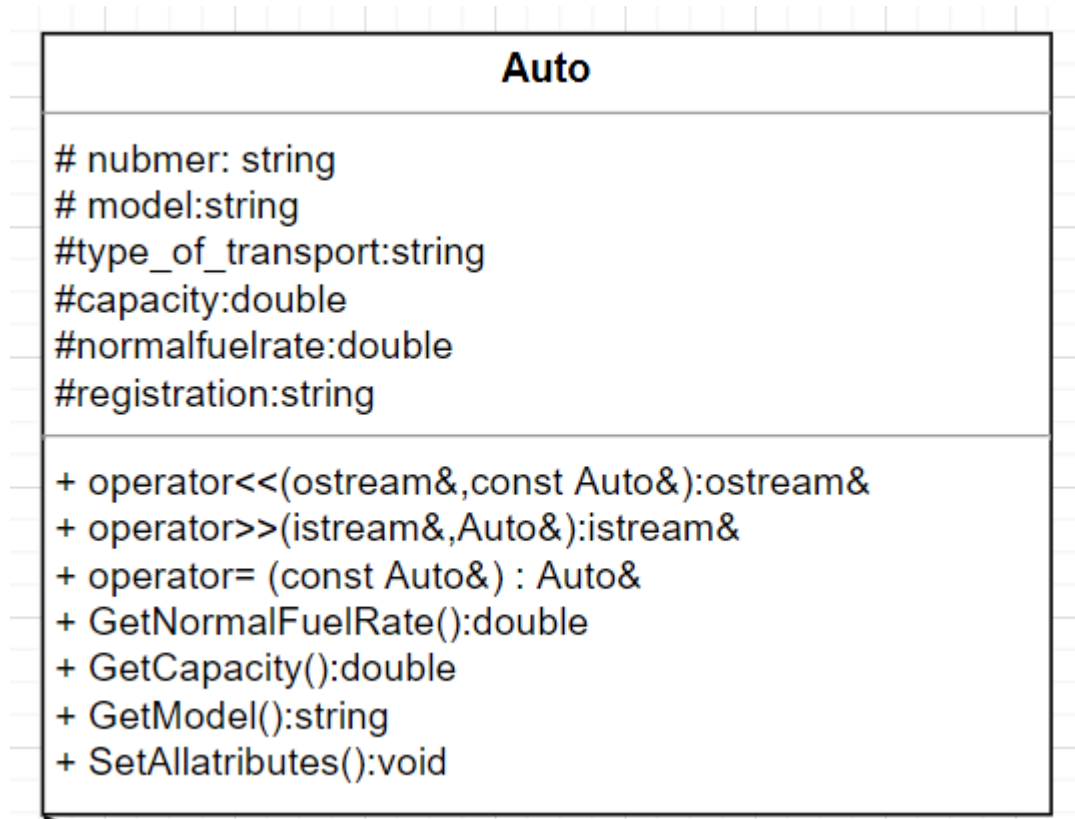


Рисунок 2.3 - Схема класу «Auto»

Клас Авто входить до головної ієрархії програмної системи, його роль – створити «константний» об’єкт класу(автобаза має авто одного виду), він має 6 protected атрибутів та 5 public методів:

- Number: номер автомобіля;
- Model: модель автомобіля;
- Type_of_transport: тип транспорту;
- Capacity: вантажопідйомність;
- Normalfuelrate: норма витрат пального;
- Registration: місце приписки авто.

Методи та оператори:

- Перевантажені оператори << та >> для виведення/введення об’єктів;
- Перевантажений оператор = для присвоювання об’єктів класу Auto;
- GetNormalFuelRate(), гетер, що створений для отримання поля normalfuelrate;

- GetCapacity(), гетер, що створений для отримання поля capacity;
- GetModel(), гетер, що створений для отримання поля model;
- Setallatributes(), сетер, що встановлює значення всіх полів класу Авто для похідного класу Рейс(дефолтні значення).

2.4 Опис класу «Driver»

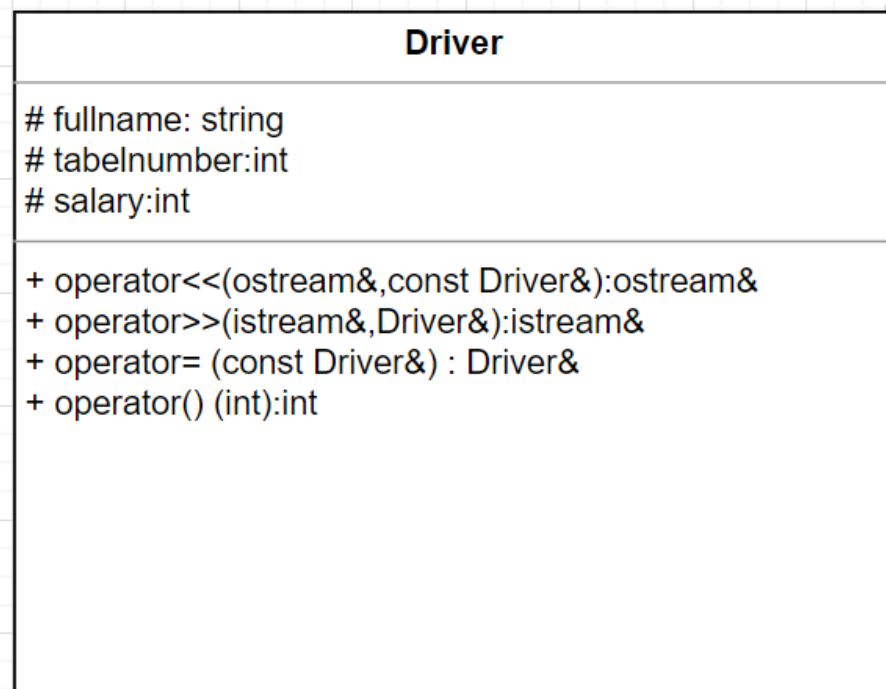


Рисунок 2.4 - Схема класу «Driver»

Клас Водій містить 3 protected поля, та 4 public методи/оператори:

- перевантажені оператори << та >> для виведення/введення об'єктів;
- перевантажений оператор = для присвоювання об'єктів класу Driver;
- перевантажений оператор () для збільшення/зменшення зарплати об'єктів класу Driver.

Оператори:

- перевантажені оператори << та >> для виведення/введення об'єктів;
- перевантажений оператор = для присвоювання об'єктів класу Driver;
- перевантажений оператор () для збільшення/зменшення зарплати об'єктів класу Driver.

2.5 Опис похідного від класів «Driver» та «Auto» класу «Route»

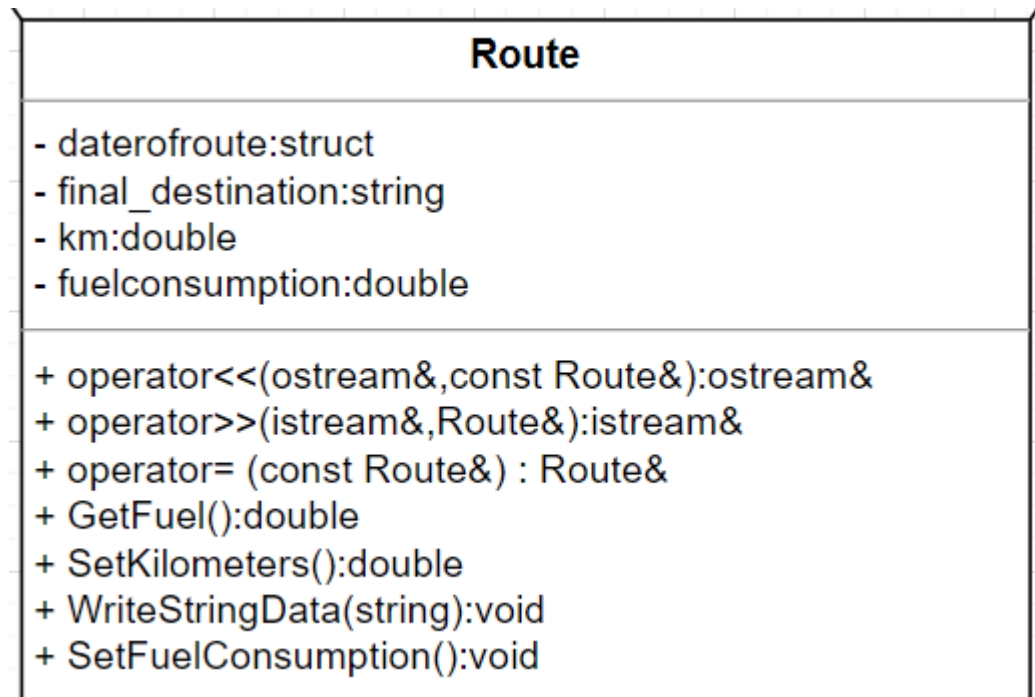


Рисунок 2.5 - Схема класу «Route»

Містить в собі 4 private поля та 7 методів:

Поля:

- Dateofroute: вкладена структура, що містить поля day, month, year;
- Final_destination: кінцевий пункт рейсу;
- Km: дистанція;
- Fuelconsumption: витрати пального.

Методи/оператори:

- перевантажені оператори << та >> для виведення/введення об'єктів.
- перевантажений оператор = для присвоювання об'єктів класу Route.
- GetFuel(): гетер, що повертає fuel.
- SetKilometers(): сетер, що встановлює km.
- WriteStringData(): метод, що робить звіт з рейсів у окремий текстовий файл.
- SetFuelConsumption(): сетер, що встановлює fuelconsumption.

2.6 Опис ієрархії класів-контейнерів: «Container», «SetofDrivers», «Timetable»

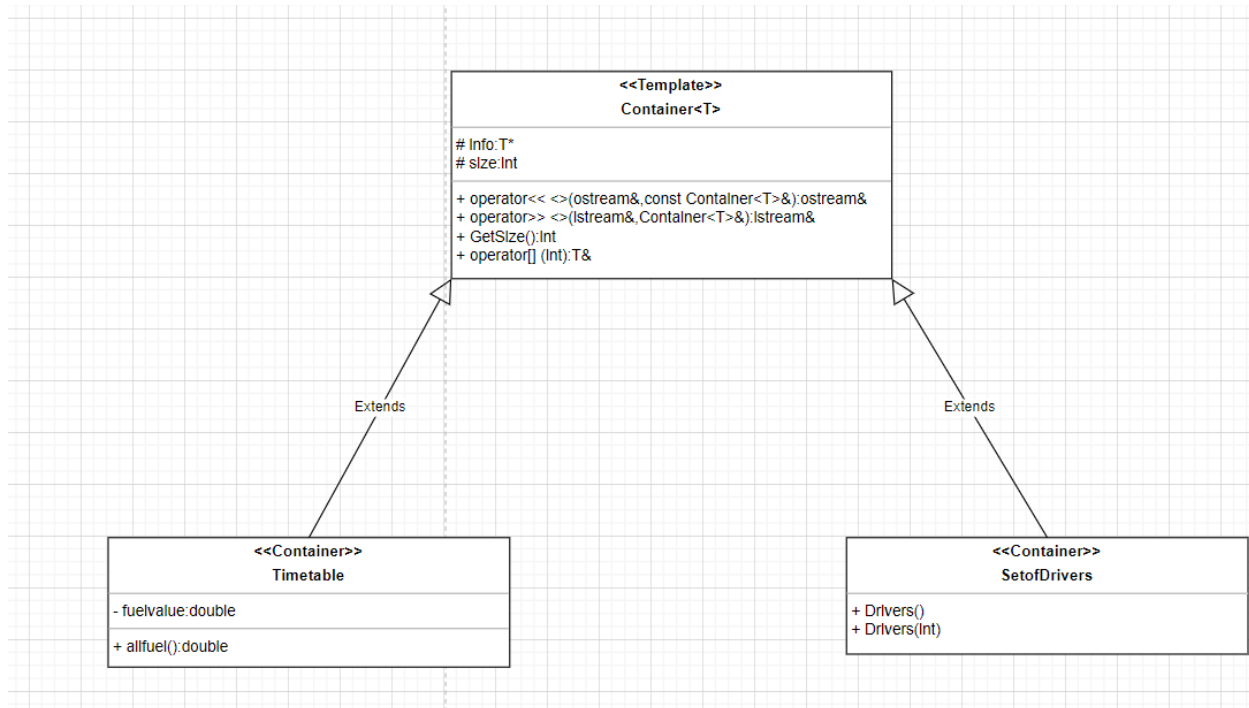


Рисунок 2.3 – Схема ієрархії контейнерів

Шаблонний клас-контейнер «Container» (динамічний масив) містить в собі вказівник шаблонного типу `T` та `size` – розмір масиву, гетер, що повертає `size` та перевантажені оператори введення/виведення об'єктів `>>`, `<<` та доступу `[]` похідні класи успадковують всі поля та методи.

Клас `Timetable` містить додаткове поле-лічильник `fuelvalue` та метод `allfuel()` який в циклі рахує загальні витрати пального за всі рейси одного замовлення.

2.7 Опис ієрархії класів-винятків «ErrorMessage», «ArrayMessage», «RangeMessage»

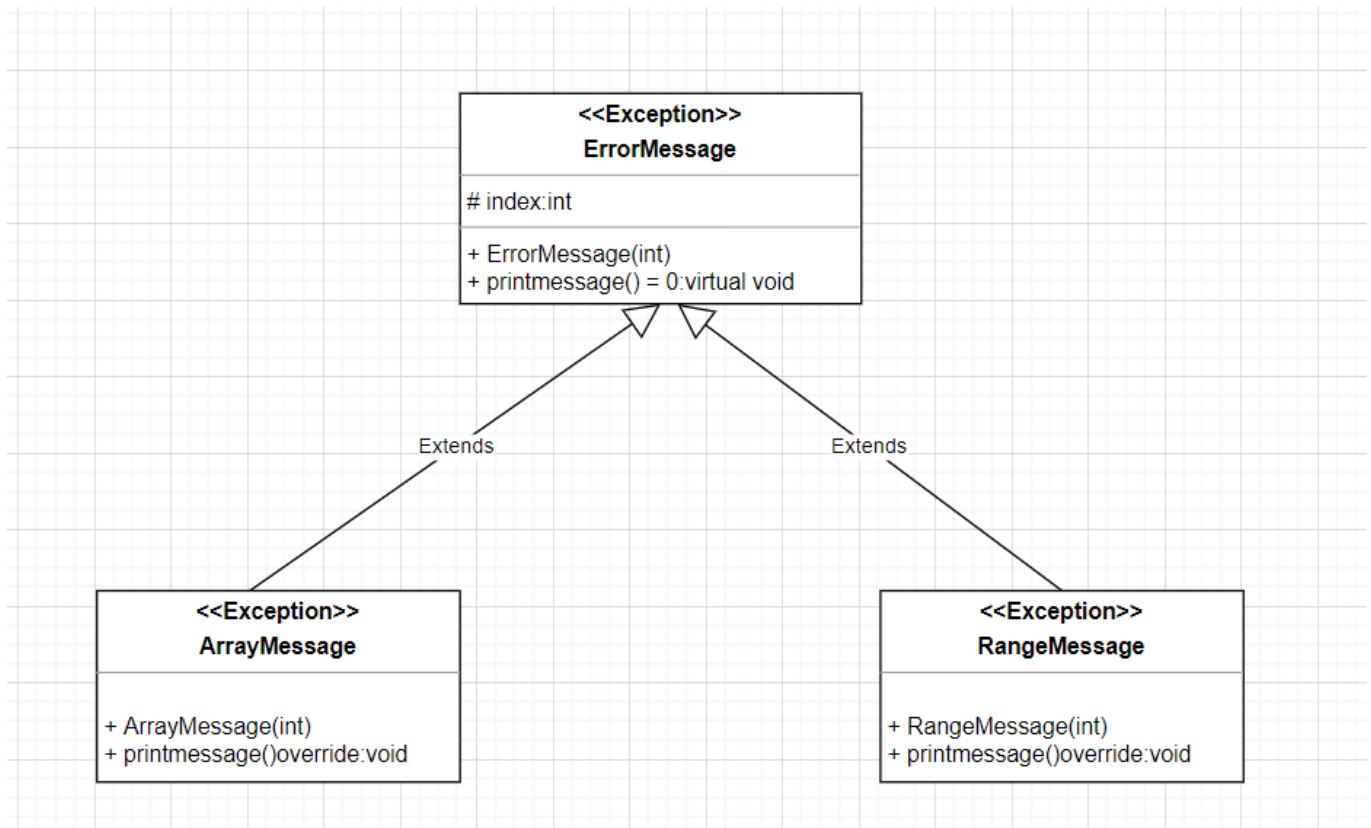


Рисунок 2.7 – Схема ієрархії класів-винятків

Абстрактний базовий клас-виняток **ErrorMessage** містить protected поле **index** та чисто віртуальний метод **printmessage()**, похідні класи перевизначають цей метод в залежності від того, якого типу помилка (діапазон або занадто великий масив).

2.8 Опис класу «Order»

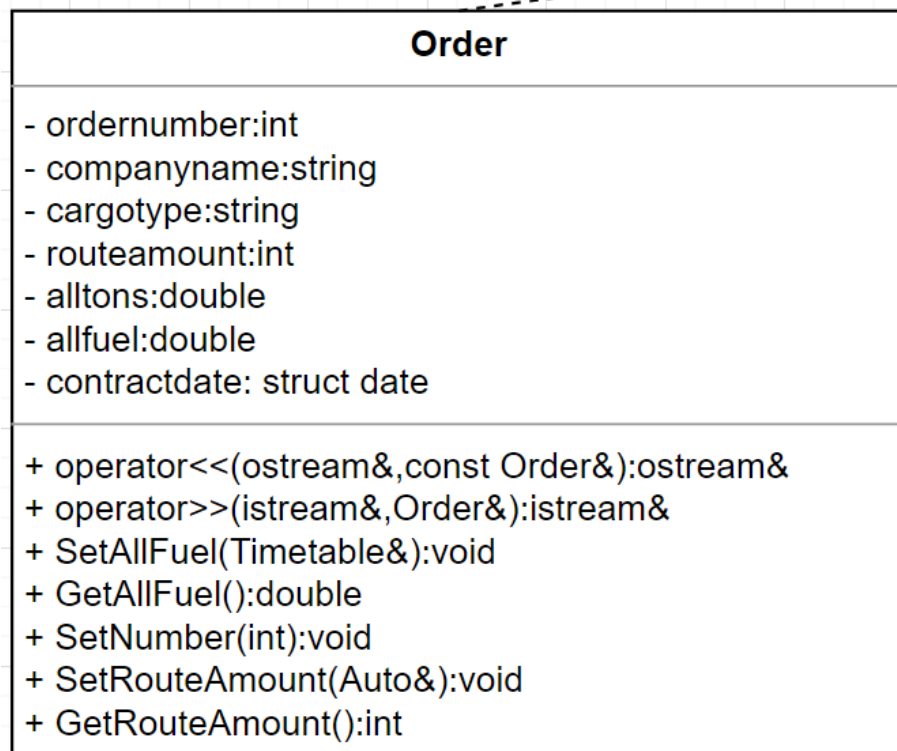


Рисунок 2.8 – Схема класу «Order»

Клас «Order» містить в собі 7 private полів та 7 public методів:

Поля:

- Ordernumber: номер замовлення;
- Companyname: назва компанії;
- Cargotype: тип вантажу;
- Routeamount: кількість рейсів;
- lltons: кількість вантажу в тонах;
- Allfuel: загальні витрати пального;
- Contractdate: дата підписання контракту.

Методи/оператори:

- перевантажені оператори введення/виведення >>,<< об'єктів;
- SetAllFuel(): сетер, що встановлює значення поля allfuel;
- GetAllFuel(): гетер, що отримує значення поля allfuel;
- SetNumber(): сетер, що встановлює значення ordernumber;

- SetRouteAmount(): сетер, що встановлює значення routeamount;
- GetRouteAmount(): гетер, що отримує значення поля routeamount.

2.9 Опис класу «Manager»

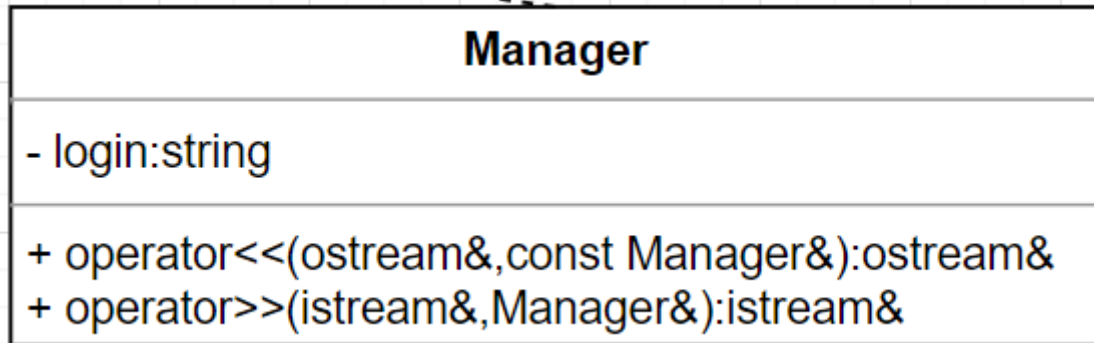


Рисунок 2.9 – Схема класу «Manager»

Клас «Manager» містить лише одне private поле login та перевантажені оператори введення/виведення об'єктів >>,<<

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис структури

Програмна система складається з 14 файлів типу .h та з 13 типу .cpp .

Два хедери – додаткових ,це «MainHeader», що підключає до головної програми усі створені класи та «Header», що містить підключення усіх бібліотек, що використовує програмна система. У головній програмі , що знаходиться в AutoBase.cpp , через блоки if – else if – else реалізоване головне меню да підменю кожного пункту.

3.2 3.2 Опис інтерфейсу

При запуску програми відображається головне меню програмної системи «Автобаза», де користувач може обрати доступний функціонал(рисунок 3.1)

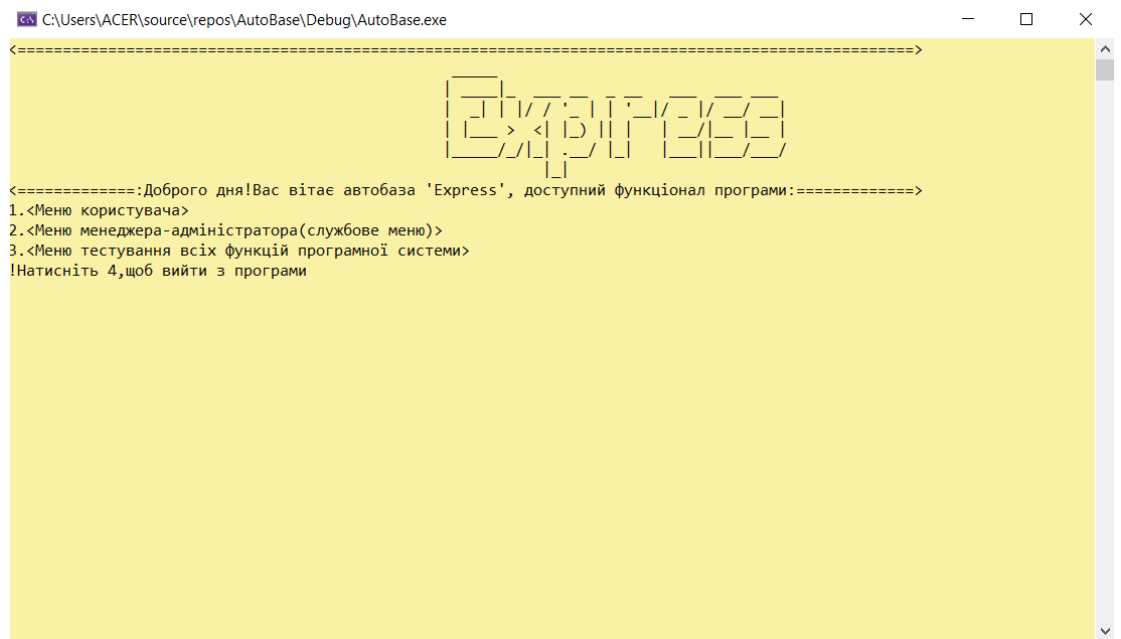


Рисунок 3.1 – Головний інтерфейс програми

1.При виборі 1 пункту меню (Меню користувача) , відображається підменю (рисунок 3.2)



Рисунок 3.2 – Підменю, меню користувача-замовника

1. При виборі 1-го пункту меню користувача-замовника відображається інформація про авто, що буде використано для рейсів (рисунки 3.3)

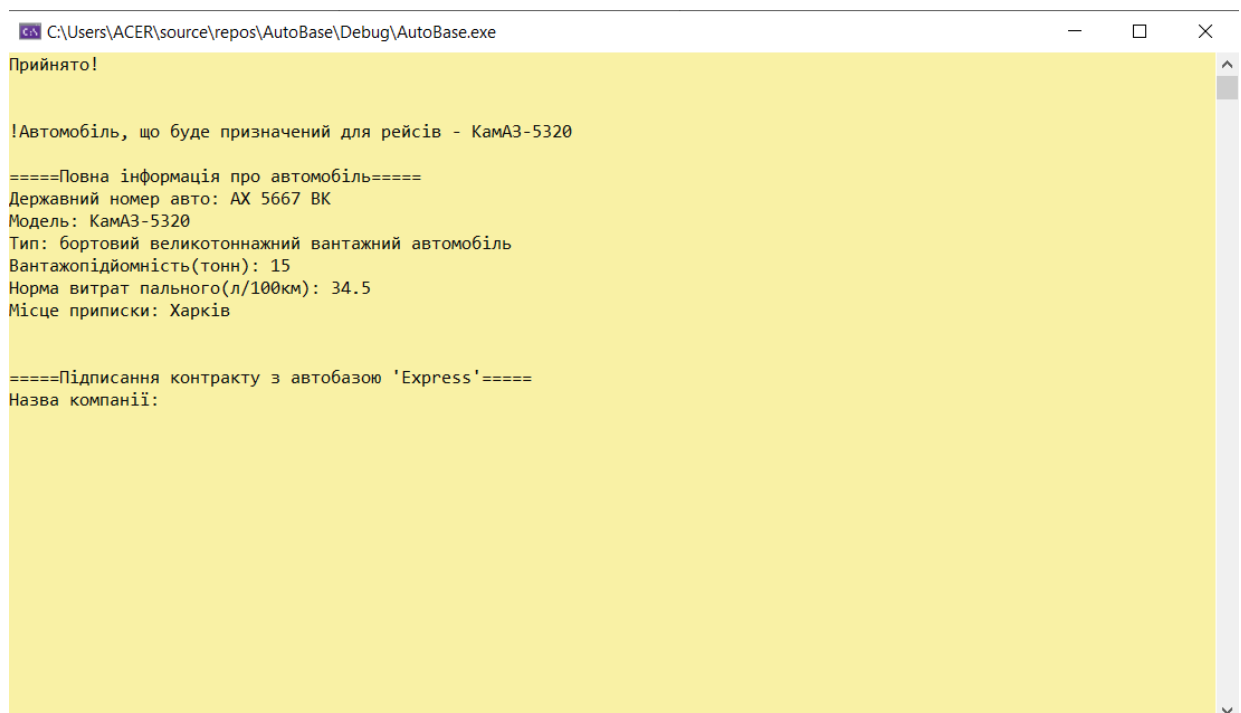
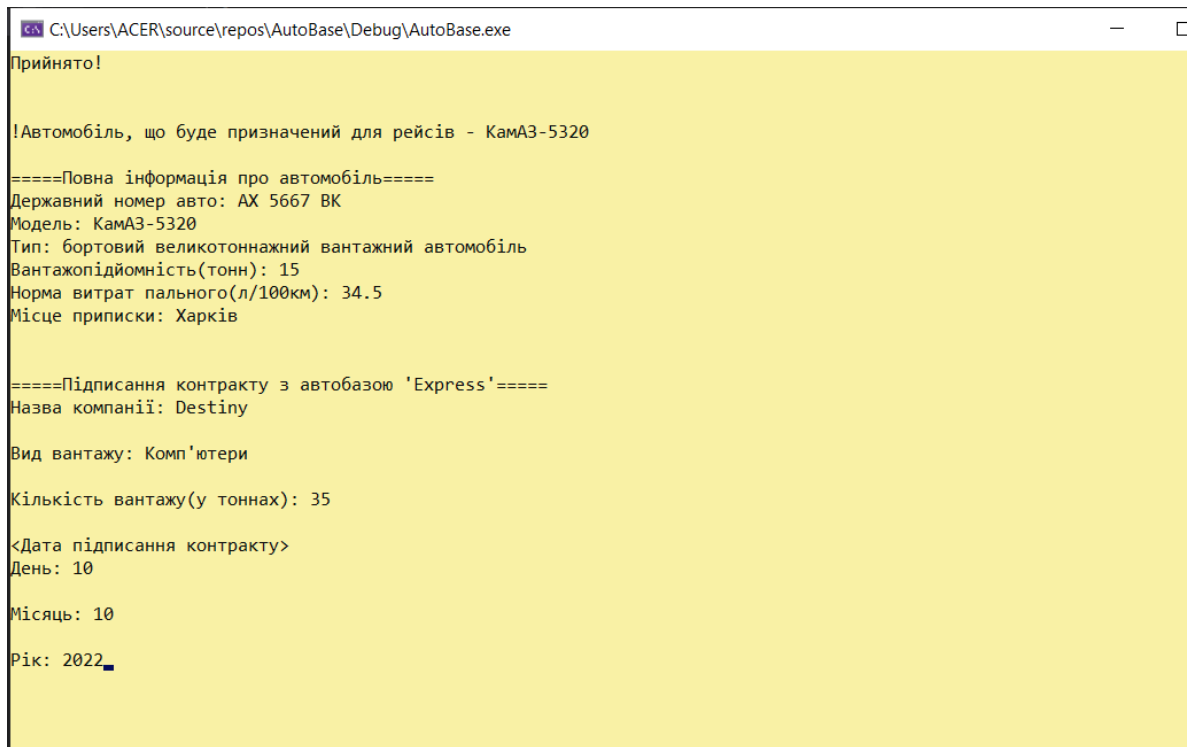


Рисунок 3.3- Інформація про автомобіль

Підписання контракту з автобазою, вхідні дані:

- Назва компанії: Destiny
- Вид вантажу: Комп'ютери
- Кількість вантажу(в тоннах): 35
- Дата підписання контракту: 10.10.2022



```
C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
Прийнято!

!Автомобіль, що буде призначений для рейсів - КамАЗ-5320

====Повна інформація про автомобіль====
Державний номер авто: АХ 5667 ВК
Модель: КамАЗ-5320
Тип: бортовий великотоннажний вантажний автомобіль
Вантажопідйомність(тонн): 15
Норма витрат пального(л/100км): 34.5
Місце приписки: Харків

====Підписання контракту з автобазою 'Express'====
Назва компанії: Destiny
Вид вантажу: Комп'ютери
Кількість вантажу(у тоннах): 35
<Дата підписання контракту>
День: 10
Місяць: 10
Рік: 2022_
```

Рисунок 3.4 - Підпис контракту

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe

```

Вид вантажу: Комп'ютери

Кількість вантажу(у тоннах): 35

<Дата підписання контракту>
День: 10

Місяць: 10

Рік: 2022

Контракт підписаний!
Номер замовлення 1
!Кількість рейсів для замовлення: 3

Замовлення, що буде занесено до бази:
=====Інформація про замовлення=====
Номер замовлення: 1
Назва компанії-замовника : Destiny
Вид вантажу: Комп'ютери
Кількість вантажу(у тоннах): 35
Кількість рейсів: 3
Загальні витрати пального: 0
<Дата підписання контракту з автобазою>
День: 10
Місяць: 10
Рік: 2022
Натисніть будь-яку клавішу █

```

Рисунок 3.5 – Замовлення , що буде занесене до бази

Програма самостійно розраховувала скільки потрібно рейсів на одне замовлення в залежності від загального об'єму перевезень, загальні витрати пального 0, ця інформація буде розрахована далі.

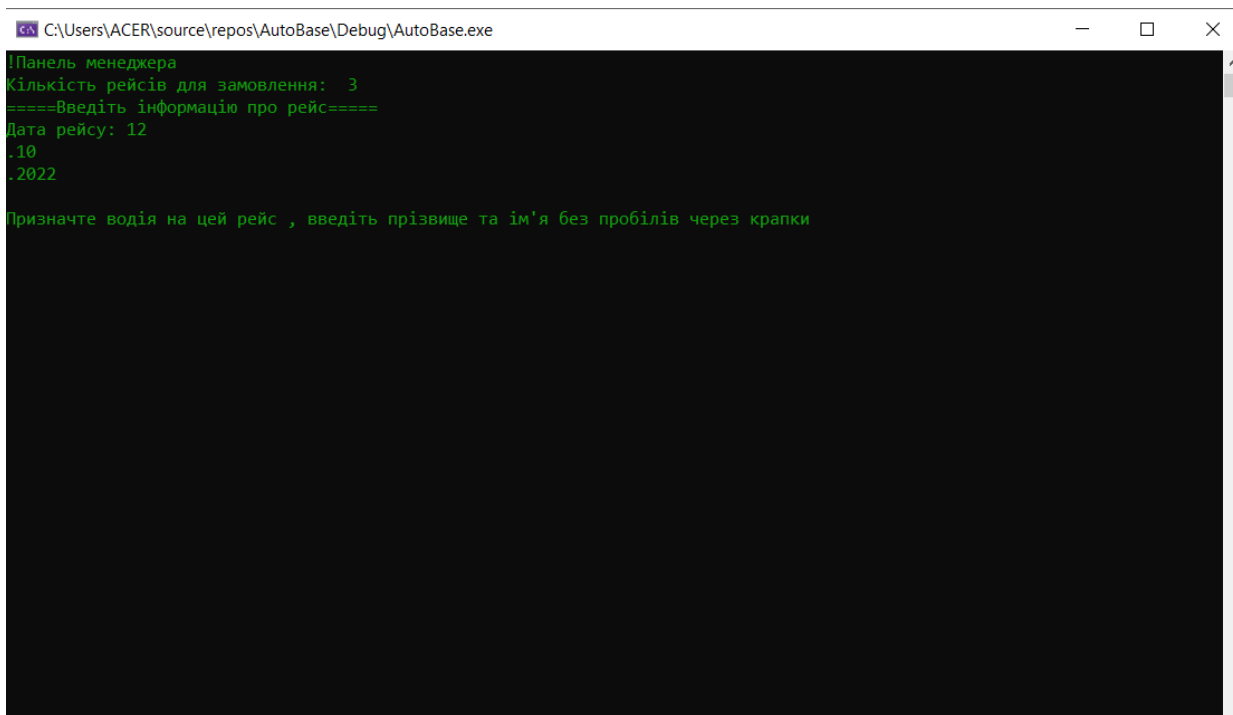


Рисунок 3.6 – Панель менеджера

Відкривається панель менеджера, який повинен заповнити інформацію про рейси.

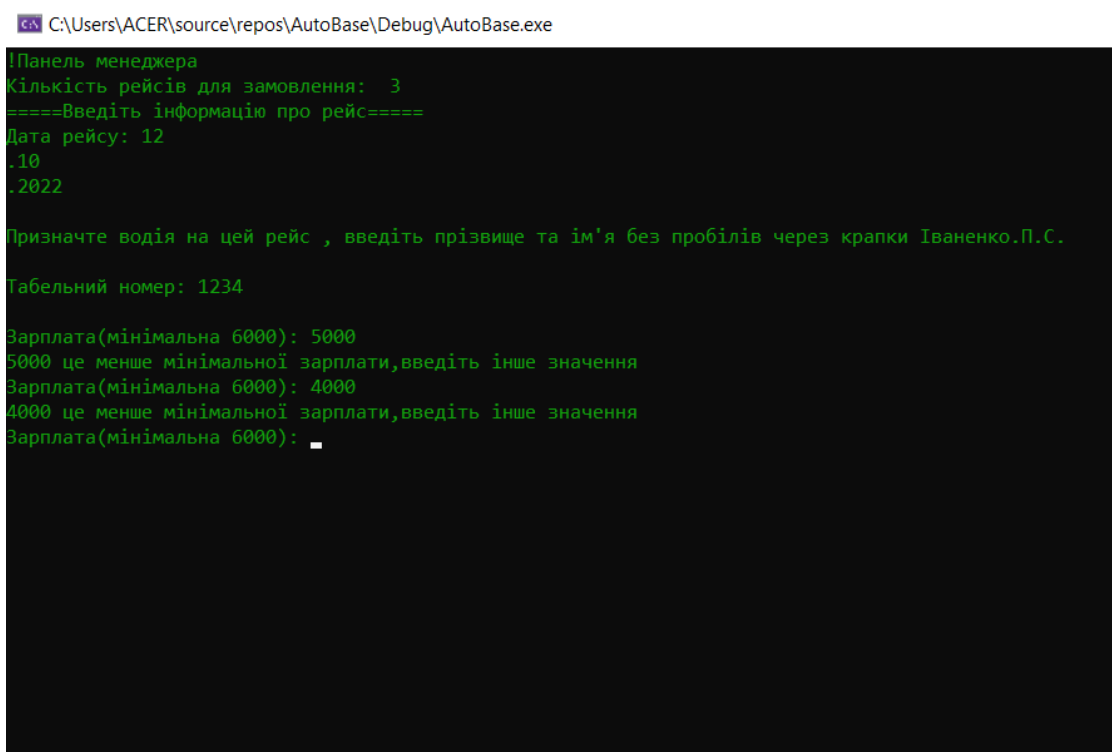


Рисунок 3.7 – Демонстрація виключень

Якщо менеджер вводить зарплату водія меншу, ніж за мінімальну, то спрацьовує виключення і треба ввести дані знову.


```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
Панель менеджера
Кількість рейсів для замовлення: 3
=====Введіть інформацію про рейс=====
Дата рейсу: 12
.10
.2022

Призначте водія на цей рейс , введіть прізвище та ім'я без пробілів через крапки Іваненко.П.С.

Табельний номер: 1234

Зарплата(мінімальна 6000): 5000
5000 це менше мінімальної зарплати,введіть інше значення
Зарплата(мінімальна 6000): 4000
4000 це менше мінімальної зарплати,введіть інше значення
Зарплата(мінімальна 6000): 6500
Кінцевий пункт рейсу(Доступні міста на період воєнного стану - Київ, Запоріжжя, Суми, Чернігів, Лозова) укажіть місто доставки: Херсон
Херсон це недоступне місто!
укажіть місто доставки: Маріуполь
Маріуполь це недоступне місто!
укажіть місто доставки: Карлівка
Карлівка це недоступне місто!
укажіть місто доставки: 

```

Рисунок 3.8 – Демонстрація виключення (кінцевий пункт рейсу)

Якщо менеджер введе недоступне місто, спрацює виключення і програма буде очікувати доступне місто з перелічених.

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
Призначте водія на цей рейс , введіть прізвище та ім'я без пробілів через крапки Іваненко.П.С.

Табельний номер: 1234

Зарплата(мінімальна 6000): 5000
5000 це менше мінімальної зарплати,введіть інше значення
Зарплата(мінімальна 6000): 4000
4000 це менше мінімальної зарплати,введіть інше значення
Зарплата(мінімальна 6000): 6500
Кінцевий пункт рейсу(Доступні міста на період воєнного стану - Київ, Запоріжжя, Суми, Чернігів, Лозова) укажіть місто доставки: Херсон
Херсон це недоступне місто!
укажіть місто доставки: Маріуполь
Маріуполь це недоступне місто!
укажіть місто доставки: Карлівка
Карлівка це недоступне місто!
укажіть місто доставки: Київ
=====
Інформація успішно введена!

=====Введіть інформацію про рейс=====
Дата рейсу: 12
.10
.2022

Призначте водія на цей рейс , введіть прізвище та ім'я без пробілів через крапки Сидоров.М.М.

Табельний номер: 12345

Зарплата(мінімальна 6000): 7777
Кінцевий пункт рейсу(Доступні міста на період воєнного стану - Київ, Запоріжжя, Суми, Чернігів, Лозова) укажіть місто доставки: Київ
=====
Інформація успішно введена!

=====Введіть інформацію про рейс=====
Дата рейсу: 12
.10
.10

Призначте водія на цей рейс , введіть прізвище та ім'я без пробілів через крапки Василенко.П.Р.

Табельний номер: 1331

Зарплата(мінімальна 6000): 8888
Кінцевий пункт рейсу(Доступні міста на період воєнного стану - Київ, Запоріжжя, Суми, Чернігів, Лозова) укажіть місто доставки: Київ
=====
Інформація успішно введена!

Програмно було розраховано загальні витрати пального: 994.428

```

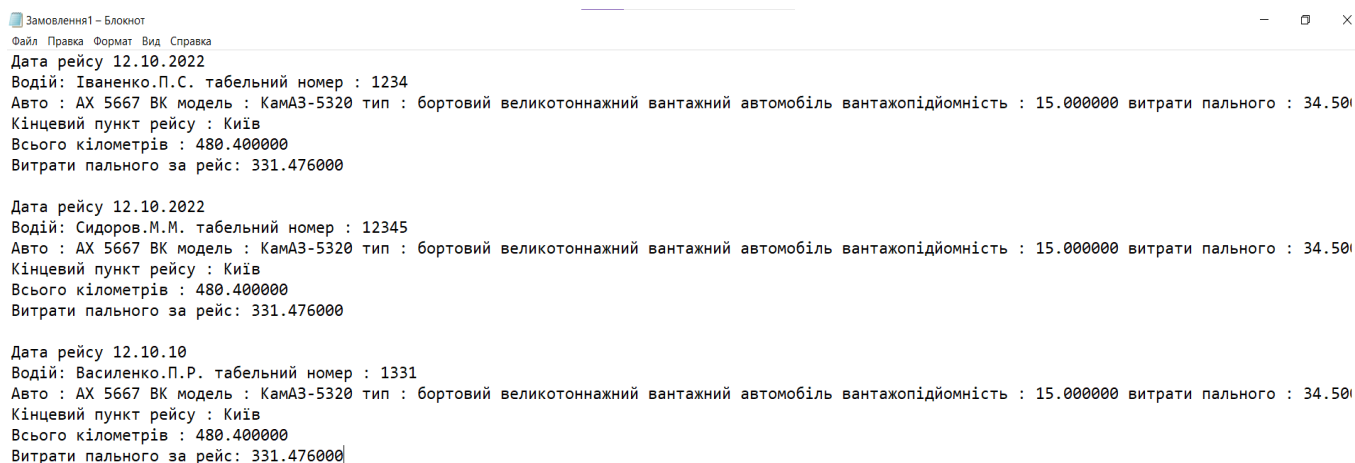
Рисунок 3.9 – Розрахування загальних витрат пального

Після введення всіх потрібних даних про рейс , програма самостійно розрахувала загальні витрати пального : 994.4. Це залежить від міста, яке буде обране менеджером.

```
Введіть назву файлу для створення звіту з рейсів по замовленню у форматі 'Замовлення<номер>.txt' Замовлення1.txt
Звіт з вашого замовлення успішно створений!
Натисніть будь-яку клавішу
```

Рисунок 3.10 – Створення звіту з логістичних рейсів

Програма підтримує функціонал створення звіту з логістичних рейсів для замовника.



```
Замовлення1 - Блокнот
Файл  Правка  Формат  Вид  Справка

Дата рейсу 12.10.2022
Водій: Іваненко.П.С. табельний номер : 1234
Авто : АХ 5667 ВК модель : КамАЗ-5320 тип : бортовий великотоннажний вантажний автомобіль вантажопідйомність : 15.000000 витрати пального : 34.50
Кінцевий пункт рейсу : Київ
Всього кілометрів : 480.400000
Витрати пального за рейс: 331.476000

Дата рейсу 12.10.2022
Водій: Сидоров.М.М. табельний номер : 12345
Авто : АХ 5667 ВК модель : КамАЗ-5320 тип : бортовий великотоннажний вантажний автомобіль вантажопідйомність : 15.000000 витрати пального : 34.50
Кінцевий пункт рейсу : Київ
Всього кілометрів : 480.400000
Витрати пального за рейс: 331.476000

Дата рейсу 12.10.10
Водій: Василенко.П.Р. табельний номер : 1331
Авто : АХ 5667 ВК модель : КамАЗ-5320 тип : бортовий великотоннажний вантажний автомобіль вантажопідйомність : 15.000000 витрати пального : 34.50
Кінцевий пункт рейсу : Київ
Всього кілометрів : 480.400000
Витрати пального за рейс: 331.476000
```

Рисунок 3.11 – Звіт з логістичних рейсів по замовленню 1

Створення звіту виконано , всі дані , що були записані програмою автоматично, введені коректно.

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
=====Автобаза 'Express'=====
Заснована у 2022 році в місті Харкові. Приймає замовлення
з доставки грузу різного типу: сировина, мебель, техніка і т.д.
Має в своєму автопарку близько 300 машин 'КамАЗ-5320'.
Головний програміст програмної системи: Крутипорох Руслан Олегович.
© 2022 Express

1.<Меню користувача>
2.<Меню менеджера-адміністратора(службове меню)>
3.<Меню тестування всіх функцій програмної системи>
!Натисніть 4,щоб вийти з програми

```

Рисунок 3.12 – Другий пункт підменю

Після завершення першого пункту, програма повертає нас до головного меню, далі був протестований другий пункт підменю, що відображає коротку інформацію про автобазу «Експрес» (рисунок 3.12)

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
=====Автобаза 'Express'=====
Заснована у 2022 році в місті Харкові. Приймає замовлення
з доставки грузу різного типу: сировина, мебель, техніка і т.д.
Має в своєму автопарку близько 300 машин 'КамАЗ-5320'.
Головний програміст програмної системи: Крутипорох Руслан Олегович.
© 2022 Express

1.<Меню користувача>
2.<Меню менеджера-адміністратора(службове меню)>
3.<Меню тестування всіх функцій програмної системи>
!Натисніть 4,щоб вийти з програми
2
Доступ до системи закодований. Введіть ключ(рік заснування автобази): 2022

```

Рисунок 3.13 – Тестування другого пункту головного меню

При виборі другого пункту головного меню «Меню менеджера-адміністратора» програма перевіряє доступ до системи за спеціальним ключем (2022). Якщо користувач введе неправильне значення – програма аварійно завершиться.

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
Ключ вірний! Будь-ласка авторизуйтесь: Введіть логін: manager

Авторизація успішна!, доброго дня manager
<=====: 1.Зареєструвати нових водіїв:=====>
<=====: 2.Список замовлень у базі даних:=====>
<=====: 3.Повернутися до головного меню:=====>

```

Рисунок 3.14 – Меню менеджера-адміністратора

Після авторизації(введення логіну) відображається меню менеджера-адміністратора.

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
=====Інформація про замовлення=====
Номер замовлення: 1
Назва компанії-замовника : Destiny
Вид вантажу: Комп'ютери
Кількість вантажу(у тоннах): 35
Кількість рейсів: 3
Загальні витрати пального: 994.428
<Дата підписання контракту з автобазою>
День: 10
Місяць: 10
Рік: 2022

=====Інформація про замовлення=====
Номер замовлення: 2
Назва компанії-замовника : Woods
Вид вантажу: Дерево(сировина)
Кількість вантажу(у тоннах): 15
Кількість рейсів: 1
Загальні витрати пального: 102.534
<Дата підписання контракту з автобазою>
День: 15
Місяць: 10
Рік: 2022

Натисніть будь-яку клавішу

```

Рисунок 3.15 – Тестування 2 пункту меню менеджера

Відображається інформація про всі замовлення , що на даний момент є в базі даних автобаз «Експрес» (для тестування було оформлено ще одне замовлення

від компанії «Woods»).

```
C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
Ключ вірний! Будь-ласка авторизуйтесь: Введіть логін: івів

Авторизація успішна!, доброго дня івів
<===== 1.Зареєструвати нових водіїв:=====>
<===== 2.Список замовлень у базі даних:=====>
<===== 3.Повернутися до головного меню:=====>
1
!Прийнято

Кількість нових водіїв: 3
=====Введіть дані про водія=====
Прізвище та ініціали: Іванов.Д.І.
Табельний номер: 2452
Зарплата(мінімальна 6000): 9000
Інформація успішно введена!

=====Введіть дані про водія=====
Прізвище та ініціали: Захаров.А.А.
Табельний номер: 51
Зарплата(мінімальна 6000): 9990
Інформація успішно введена!

=====Введіть дані про водія=====
Прізвище та ініціали: Дзюба.З.Ф.
Табельний номер: 4123
Зарплата(мінімальна 6000): 8000
Інформація успішно введена!

!Нові працівники
=====Інформація про водія=====
Прізвище та ініціали: Іванов.Д.І.
Табельний номер: 2452
Зарплата: 9000
=====Інформація про водія=====
Прізвище та ініціали: Захаров.А.А.
Табельний номер: 51
Зарплата: 9990
=====Інформація про водія=====
Прізвище та ініціали: Дзюба.З.Ф.
Табельний номер: 4123
Зарплата: 8000

!Збільшити зарплату новим працівникам? 1.так 2.ні
1
```

Рисунок 3.16 – тестування 1 пункту меню менеджера

Після вибору 1 пункту меню менеджера програма запитує скільки буде зареєстровано нових робітників . Після введення числа – заповнення даних про водіїв. Також є можливість підвищити зарплату новим працівникам.

```

!Збільшити зарплату новим працівникам? 1.так 2.ні
1
Збільшити на ? 200

Нові дані:
=====Інформація про водія=====
Прізвище та ініціали: Іванов.Д.І.
Табельний номер: 2452
Зарплата: 9200
=====Інформація про водія=====
Прізвище та ініціали: Захаров.А.А.
Табельний номер: 51
Зарплата: 10190
=====Інформація про водія=====
Прізвище та ініціали: Дзюба.З.Ф.
Табельний номер: 4123
Зарплата: 8200

1.<Меню користувача>
2.<Меню менеджера-адміністратора(службове меню)>
3.<Меню тестування всіх функцій програмної системи>
!Натисніть 4,щоб вийти з програми

```

Рисунок 3.17 – результат вибору підвищення зарплати

Менеджер пише суму на яку буде підвищена зарплата нових працівників. Далі будуть виведені на екран оновлені дані працівників.

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
!Цей розділ створений для тестування конструкторів, методів та операторів,що не використовувалися
в основній програмі

=====Введіть інформацію про авто=====
Державний номер автомобіля: _

```

Рисунок 3.18 – 3 пункт головного меню – тестування

Цей пункт створений для тестування всіх конструкторів, методів та операторів , що не увійшли до основного функціоналу програмної системи .

```

=====Введіть інформацію про авто=====
Державний номер автомобіля: 232131
Модель: TESLA
Тип: електрокар
Вантажопідйомність: 5
Норма витрат пального: 0
Місце приписки: Київ
Інформація успішно введена!
=====Повна інформація про автомобіль=====
Державний номер авто: VALUE
Модель: VALUE2
Тип: VALUE3
Вантажопідйомність(тонн): 124.2
Норма витрат пального(л/100км): 245.3
Місце приписки: VALUE4

=====Повна інформація про автомобіль=====
Державний номер авто: VALUE
Модель: VALUE2
Тип: VALUE3
Вантажопідйомність(тонн): 124.2
Норма витрат пального(л/100км): 245.3
Місце приписки: VALUE4

=====Повна інформація про автомобіль=====
Державний номер авто: VALUE
Модель: VALUE2
Тип: VALUE3
Вантажопідйомність(тонн): 124.2
Норма витрат пального(л/100км): 245.3
Місце приписки: VALUE4

=====Повна інформація про автомобіль=====
Державний номер авто: 232131
Модель: TESLA
Тип: електрокар
Вантажопідйомність(тонн): 5
Норма витрат пального(л/100км): 0
Місце приписки: Київ

=====Інформація про водія=====
Прізвище та ініціали: Петренко.С.І.
Табельний номер: 1234
Зарплата: 6000

```

Рисунок 3.19 – результат тестування 1

```

C:\Users\ACER\source\repos\AutoBase\Debug\AutoBase.exe
Табельний номер: 1234
Зарплата: 6000

=====Інформація про водія=====
Прізвище та ініціали: Петренко.С.І.
Табельний номер: 1234
Зарплата: 6000

=====Інформація про рейс: Харків - VALUE6=====
Дата рейсу: 1.2.3
=====Інформація про водія=====
Прізвище та ініціали: VALUE
Табельний номер: 120
Зарплата: 120
=====Повна інформація про автомобіль=====
Державний номер авто: VALUE2
Модель: VALUE3
Тип: VALUE4
Вантажопідйомність(тонн): 131
Норма витрат пального(л/100км): 2
Місце приписки: VALUE5
Кілометраж: 321
Розхід пального за рейс: 121
=====

=====Інформація про рейс: Харків - VALUE6=====
Дата рейсу: 1.2.3
=====Інформація про водія=====
Прізвище та ініціали: VALUE
Табельний номер: 120
Зарплата: 120
=====Повна інформація про автомобіль=====
Державний номер авто: VALUE2
Модель: VALUE3
Тип: VALUE4
Вантажопідйомність(тонн): 131
Норма витрат пального(л/100км): 2
Місце приписки: VALUE5
Кілометраж: 321
Розхід пального за рейс: 121
=====

!Тестування завершено, всі методи, оператори працюють коректно.
1.<Меню користувача>
2.<Меню менеджера-адміністратора(службове меню)>
3.<Меню тестування всіх функцій програмної системи>
!Натисніть 4, щоб вийти з програми

```

Рисунок 3.20 – результат тестування 2

Всі методи, конструктори та оператори працюють коректно

ВИСНОВКИ

Під час виконання курсової роботи було проведено аналіз предметної області, повторено теоретичні матеріали з предметної області, розроблено програму мовою програмування C++ із використанням середовища розробки Microsoft Visual Studio 2019. Для виконання роботи були використані матеріали лекцій та навчальна література з об'єктно-орієнтованого програмування (ООП). Була сформована мета та шлях її досягнення. Визначено алгоритм роботи, за допомогою якого була розроблена програма. В результаті отримано практичні навички в області розробки об'єктно-орієнтованих програм, побудови діаграм класів UML та проектування програм.

Під час роботи було виявлено, що ООП є найкращим рішенням для розробки великих за обсягом програмних проектів.

Результатом стала програма AutoBase, що виконує роботу менеджера та бухгалтера одночасно. Це значно облегчує обробку замовлень та математичні розрахунки, що необхідні при аналізі даних.

ПОСИЛАНИЯ

1. Страуструп Б. Программирование: принципы и практика с использованием C++, 2-е изд. : Пер. с англ. – М.: ООО "И.Д. Вильямс" , 2016. 1328 с.
2. Хортон А. Visual C++ 2010: полный курс: ООО «И.Д. Вильямс», 2011. 1216 с.
3. Рао, Сиддхартха Освой самостоятельно C++ по одному часу в день, 8-е изд.: Пер. с англ. С.Рао. – СПб.: ООО «Альфа-книга», 2017. – 752 с.
4. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений: Вильямс, 2008. – 720с.
5. Шилдт, Герберт. C++: руководство для начинающих, 2-е издание.: Пер.с англ./ Г.Шилдт. – М.: Издательский дом "Вильямс", 2005. – 672 с.

Додаток А

Міністерство освіти і науки України

Затверджую Керівник
курсової роботи доцент кафедри
системотехніки

_____ Т. Г. Білова
(Підпис, дата)

ПРОГРАМНА СИСТЕМА «АВТОБАЗА»

Текст програми

Аркуш затвердження

Студент групи _____ КНТ- 21-5
(Назва групи)

-
_____ Крутипорох Руслан Олегович
(Підпис, дата, прізвище, ім'я, по батькові)

Міністерство освіти и науки України

ПРОГРАМНА СИСТЕМА «АВТОБАЗА»
аркушів 50

2022

ДОДАТОК Б

Файл «AutoBase.cpp»

```

#include "MainHeader.h" //Підключення головного хедеру
int main()
{
    SetConsoleCP(1251); //Встановлення підтримки української мови
    SetConsoleOutputCP(1251);

    Manager manager; //Об'єкт Менеджер
    vector<Order> orders; //Ініціалізація "Бази" замовлень

    int menu = 0; //Ініціалізація меню/подменю
    int podmenu = 0;
    int numberoforders = 0; //Лічильник, що рахує кількість замовлень
    Menu mainmenu; mainmenu.header();
    while (menu != 4) {
        if (menu == 0)
        {
            system("Color E0"); //Встановлення кольору консолі
            mainmenu.showbasemenu();
            cin >> menu;
        }
        else if (menu == 1) {
            system("cls");
            mainmenu.showusermenu();
            while (podmenu != 3) {
                cin >> podmenu;
                if (podmenu == 1) { //Створення замовлення
                    system("cls");
                    cout << "Прийнято!\n\n" << endl;
                    Auto defaultauto("AX 5667 ВК", "КамАЗ-5320", "бортовий великотоннажний
вантажний автомобіль", 15, 34.5, "Харків"); //Дефолтне авто автобази

                    cout << "!Автомобіль, що буде призначений для рейсів - " <<
defaultauto.GetModel() << endl << endl; cout << defaultauto << endl << endl;
                    Order order; cin >> order;

                    order.SetNumber(numberoforders + 1); order.SetRouteAmount(defaultauto);
                    cout << "\n\nЗамовлення, що буде занесено до бази: " << endl; cout <<
order; cout << "Натисніть будь-яку клавішу "; string flag; cin >> flag;

                    system("cls"); system("Color 02");

                    cout << "!Панель менеджера" << endl; //Передача управління
менеджеру, зміна теми консолі
                    cout << "Кількість рейсів для замовлення: " << order.GetRouteAmount() <<
endl;

                    try {
                        Timetable timetable(order.GetRouteAmount());
                        cin >> timetable; order.SetAllFuel(timetable); cout << "\nПрограмно
було розраховано загальні витрати пального: " << order.GetAllFuel() << endl;
                        string filename; cout << "Введіть назву файлу для створення звіту з
рейсів по замовленню у форматі 'Замовлення<номер>.txt' ";
                        cin >> filename; cout << "Звіт з вашого замовлення успішно створений!"
<< endl;

                        for (int i = 0; i < timetable.GetSize(); i++) { //Створення звіту
                            timetable[i].WriteStringData(filename);
                        }
                    }
                }
            }
        }
    }
}

```

```

        catch (ErrorMessage* error) {           //Ловимо виключення
            string flag;
            error->printmessage();
            cout << "Натисніть будь-яку клавішу "; cin >> flag; system("Color E0");
system("cls"); menu = 0;
            break;
        }
        catch (...) { cout << "Помилка створення"; }

        orders.push_back(order);           //збільшення бази на одне замовлення
        numberoforders++;
        cout << "Натисніть будь-яку клавішу "; cin >> flag;
        system("Color E0");
    }
    else if (podmenu == 2) {               //Інформація про автобазу
        system("cls");
        mainmenu.autobaseinfo();
        menu = 0;
        cout << endl << endl;
    }
    else if (podmenu == 3) //Вихід до головного меню
    {
        system("cls");
        podmenu = 0;
        menu = 0;
        break;
    }
    else {
        cout << "Невірно вибраний варіант,спробуйте ще" << endl;
    }
    break;
}
else if (menu == 2) { //Меню менеджера
    if (menu == 0)
    {
        mainmenu.showservicemenu();
        cin >> menu;
    }
    menu = 0;
    if (!mainmenu.check()) {               //Перевірка на ключ
        cout << "Ключ невірний! аварійне завершення програми"; break;
    }
    else {
        system("cls"); system("Color 02");
        cout << "Ключ вірний! Будь-ласка авторизуйтеся: "; cin >> manager; cout << endl
<< manager; //авторизація за логіном
        mainmenu.showservicemenu();
        while (podmenu != 4) {
            cin >> podmenu;
            if (podmenu == 1) {           //Реєстрація нових водіїв
                cout << "!Прийнято\n\n"; int size;
                cout << "Кількість нових водіїв: "; cin >> size;
                try {
                    Drivers drivers(size);
                    cin >> drivers;

                    cout << "!Нові працівники" << endl;
                    cout << drivers << endl << endl;

                    int answer; cout << "!Збільшити зарплату новим працівникам? 1.так
2.ні \n"; cin >> answer; //Додаткова функція
                    if (answer == 1) {
                        int up; cout << "Збільшити на ? "; cin >> up;
                        for (int i = 0; i < drivers.GetSize(); i++) {

```

```

        drivers[i](up);
    }
    cout << "\n\nНові дані:\n"; cout << drivers;
}
else if (answer == 2) { menu = 0; break; }
else { menu = 0; break; }

}
catch (ErrorMessage* error) {          //ЛОВИМО ВИКЛЮЧЕННЯ
    string flag;
    error->printmessage();
    cout << "Натисніть будь-яку клавішу "; cin >> flag; system("Color
E0"); system("cls"); menu = 0;
    break;
}
catch (...) {
    cout << "Помилка" << endl;
}

menu = 0;
break;
}
else if (podmenu == 2) {          //Відображення "бази" замовлень
    system("cls");
    try {
        if (orders.size() == 0) { throw orders.size(); }
        for (int i = 0; i < orders.size(); i++) {
            cout << orders[i] << endl;
        }
    }
    catch (...) {
        cout << "!Список замовлень порожній" << endl;
    }
    string flag;
    cout << "Натисніть будь-яку клавішу "; cin >> flag; cout << endl <<
endl;

    menu = 0;
    break;
}
else if (podmenu == 3) { //Вихід до головного меню
    system("cls");
    podmenu = 0;
    menu = 0;
    break;
}

else {
    cout << "Невірно вибраний варіант, спробуйте ще" << endl;
}
break;
}
}

}
else if (menu == 3) {          //Розділ тестування
    system("Color 02"); system("cls");
    cout << "!Цей розділ створений для тестування конструкторів, методів та операторів, що
не використовувалися\nв основній програмі\n\n\n";

    //Auto
    Auto auto1("VALUE", "VALUE2", "VALUE3", 124.2, 245.3, "VALUE4"); // Конструктор зі
списком ініціалізації
    Auto auto2(auto1);          // Конструктор копіювання класу Auto
    Auto auto3; auto3 = auto2; //Оператор =
    Auto auto4; cin >> auto4; //Оператор >>

```

```

        cout << auto1 << endl << auto2 << endl << auto3 << endl << auto4 << endl << endl;
//Виведення об'єктів
////////////////////////////////////
//Driver
Driver driver1("Петренко.С.І.", 1234, 6000); // Конструктор зі списком ініціалізації
Driver driver2(driver1); // Конструктор копіювання
Driver driver3 = driver2; //Оператор =
cout << driver1 << endl << driver2 << endl << driver3 << endl << endl; //Виведення
об'єктів
////////////////////////////////////
//Route
Route route1({1,2,3}, "VALUE", 120, 120, "VALUE2", "VALUE3", "VALUE4", 131, 2,
"VALUE5", "VALUE6", 321, 121); //Конструктор зі списком ініціалізації
Route route2; route2 = route1; //Оператор =
cout << route1 << endl << route2 << endl << endl;
////////////////////////////////////

cout << "!Тестування завершено, всі методи, оператори працюють коректно." << endl;
menu = 0;
}
else if (menu == 4) { break; } //Завершення програми
}

```

Файл «Auto.h»

```

#pragma once
#include "Header.h"
class Auto
{
protected:
    string number;
    string model;
    string type_of_transport;
    double capacity;
    double normalfuelrate;
    string registration;
public:
    Auto();
    Auto(string, string, string, double, double, string);
    Auto(const Auto&);
    friend ostream& operator<<(ostream&, const Auto&);
    friend istream& operator>>(istream&, Auto&);
    Auto& operator=(const Auto&);
    double GetNormalFuelRate();
    double GetCapacity();
    string GetModel();
    void SetAllatributes();
};

```

Файл «Auto.cpp»

```

#include "Auto.h"
Auto::Auto() {
    number = " ";
    model = " ";
    type_of_transport = " ";
    capacity = 0;
    normalfuelrate = 0;
    registration = " ";
}
Auto::Auto(string number1, string model1, string type, double capacity1, double
normalfuelrate1, string registr) : number(number1), model(model1), type_of_transport(type),
capacity(capacity1), normalfuelrate(normalfuelrate1), registration(registr) {}
Auto::Auto(const Auto& object) {
    number = object.number;
}

```



```

        model = object.model;
        type_of_transport = object.type_of_transport;
        capacity = object.capacity;
        normalfuelrate = object.normalfuelrate;
        registration = object.registration;
    }
    void Auto::SetAllatributes() {
        number = "AX 5667 BK";
        model = "КамАЗ-5320";
        type_of_transport = "бортівий великотоннажний вантажний автомобіль";
        capacity = 15;
        normalfuelrate = 34.5;
        registration = "Харків";
    }
    double Auto::GetCapacity() {
        return capacity;
    }
    string Auto::GetModel()
    {
        return model;
    }
    double Auto::GetNormalFuelRate() {
        return normalfuelrate;
    }
    Auto& Auto::operator=(const Auto& object) {
        number = object.number;
        model = object.model;
        type_of_transport = object.type_of_transport;
        capacity = object.capacity;
        normalfuelrate = object.normalfuelrate;
        registration = object.registration;
        return *this;
    }
    ostream& operator<<(ostream& output, const Auto& object) {
        output << "====Повна інформація про автомобіль====" << endl;
        output << "Державний номер авто: " << object.number << endl;
        output << "Модель: " << object.model << endl;
        output << "Тип: " << object.type_of_transport << endl;
        output << "Вантажопідйомність(тонн): " << object.capacity << endl;
        output << "Норма витрат пального(л/100км): " << object.normalfuelrate << endl;
        output << "Місце приписки: " << object.registration << endl;
        return output;
    }
    istream& operator>>(istream& input, Auto& object) {
        cout << "====Введіть інформацію про авто====" << endl;
        cout << "Державний номер автомобіля: ";
        input >> object.number;
        cout << "Модель: ";
        input >> object.model;
        cout << "Тип: ";
        input >> object.type_of_transport;
        cout << "Вантажопідйомність: ";
        input >> object.capacity;
        cout << "Норма витрат пального: ";
        input >> object.normalfuelrate;
        cout << "Місце приписки: ";
        input >> object.registration;
        cout << "Інформація успішно введена!" << endl;
        return input;
    }
}

```

Файл «Driver.h»

```

#pragma once
#include "Header.h"

```

```

class Driver
{
protected:
    string fullname;
    int tabelnumber;
    int salary;
public:
    Driver();
    Driver(string, int, int);
    friend ostream& operator<<(ostream&, const Driver&);
    friend istream& operator>>(istream&, Driver&);
    Driver& operator=(const Driver&);
    int operator() (int);
};

```

Файл «Driver.cpp»

```

#include "Driver.h"
Driver::Driver() {
    fullname = " ";
    tabelnumber = 0;
    salary = 0;
}
Driver::Driver(string fio, int tabel, int zp) : fullname(fio), tabelnumber(tabel), salary(zp)
{}
Driver& Driver::operator=(const Driver& object) {
    fullname = object.fullname;
    tabelnumber = object.tabelnumber;
    salary = object.salary;
    return *this;
}
int Driver::operator() (int value) {
    salary += value;
    return salary;
}
ostream& operator<<(ostream& output, const Driver& object) {
    output << "====Інформація про водія====" << endl;
    output << "Прізвище та ініціали: " << object.fullname << endl;
    output << "Табельний номер: " << object.tabelnumber << endl;
    output << "Зарплата: " << object.salary << endl;
    return output;
}
istream& operator>>(istream& input, Driver& object) {
    cout << "====Введіть дані про водія====" << endl;
    cout << "Прізвище та ініціали: ";
    input >> object.fullname;
    cout << "Табельний номер: ";
    input >> object.tabelnumber; bool flag = false;
    while (flag != true) {
        try {
            cout << "Зарплата(мінімальна 6000): ";
            input >> object.salary;
            if (object.salary < 6000) { flag = false; throw object.salary; }
            else { flag = true; break; }
        }
        catch (const int ex) { cout << ex << " це менше мінімальної зарплати, введіть інше значення"; }
    }
    cout << "Інформація успішно введена!" << endl;
    return input;
}

```

Файл «Menu.h»

```

#pragma once

```



```

        return false;
    }
    else {
        return true;
    }
}

```

}

Файл «Order.h»

```

#pragma once
#include "Header.h"
#include "Route.h"
#include "Timetable.h"
class Order
{
private:
    struct date {
        int day;
        int month;
        int year;
        date() {
            day = 0;
            month = 0;
            year = 0;
        }
        date(int dd, int mm, int yyyy) {
            day = dd;
            month = mm;
            year = yyyy;
        }
    };
    int ordernumber;
    string companyname;
    string cargotype;
    int routeamount;
    double alltonns;
    double allfuel;
    date contractdate;
public:
    Order();
    friend ostream& operator<<(ostream&, const Order&);
    friend istream& operator>>(istream&, Order&);
    void SetAllFuel(Timetable&);
    double GetAllFuel();
    void SetNumber(int);
    void SetRouteAmount(Auto&);
    int GetRouteAmount();
};

```

Файл «Order.cpp»

```

#include "Order.h"
Order::Order() {
    ordernumber = 0;
    companyname = " ";
    cargotype = " ";
    routeamount = 0;
    alltonns = 0;
    allfuel = 0;
    contractdate.day = 0;
    contractdate.month = 0;
    contractdate.year = 0;
}
void Order::SetAllFuel(Timetable & object)
{

```

```

        allfuel = object.allfuel();
    }
    double Order::GetAllFuel()
    {
        return allfuel;
    }
    void Order::SetNumber(int number)
    {
        ordernumber = number;
        cout << "Номер замовлення " << ordernumber << endl;
    }
    void Order::SetRouteAmount(Auto& object)
    {
        routeamount = ceil(alltonns / object.GetCapacity());
        cout << "!Кількість рейсів для замовлення: " << routeamount << endl;
    }
    int Order::GetRouteAmount()
    {
        return routeamount;
    }
    ostream& operator<<(ostream& output, const Order& object) {
        output << "====Інформація про замовлення====" << endl;
        output << "Номер замовлення: " << object.ordernumber << endl;
        output << "Назва компанії-замовника : " << object.companyname << endl;
        output << "Вид вантажу: " << object.cargotype << endl;
        output << "Кількість вантажу(у тоннах): " << object.alltonns << endl;
        output << "Кількість рейсів: " << object.routeamount << endl;
        output << "Загальні витрати пального: " << object.allfuel << endl;
        output << "<Дата підписання контракту з автобазою>" << endl;
        output << "День: " << object.contractdate.day << endl;
        output << "Місяць: " << object.contractdate.month << endl;
        output << "Рік: " << object.contractdate.year << endl;
        return output;
    }
    istream& operator>>(istream& input, Order& object) {
        cout << "====Підписання контракту з автобазою 'Express'====" << endl;
        cout << "Назва компанії: "; input >> object.companyname; cout << endl;
        cout << "Вид вантажу: "; input >> object.cargotype; cout << endl;
        cout << "Кількість вантажу(у тоннах): "; input >> object.alltonns; cout << endl;
        cout << "<Дата підписання контракту>" << endl;
        cout << "День: "; input >> object.contractdate.day; cout << endl;
        cout << "Місяць: "; input >> object.contractdate.month; cout << endl;
        cout << "Рік: "; input >> object.contractdate.year; cout << endl;
        cout << "Контракт підписаний!" << endl;
        return input;
    }
}

```

Файл «Manager.h»

```

#pragma once
#include "Header.h"
class Manager
{
private:
    string login;
public:
    friend ostream& operator<<(ostream&, const Manager&);
    friend istream& operator>>(istream&, Manager&);
};

```

Файл «Manager.cpp»

```

#include "Manager.h"
ostream& operator<<(ostream& output, const Manager& object) {
    output << "Авторизація успішна!, доброго дня " << object.login << endl;
}

```

```

        return output;
    }
    istream& operator>>(istream& input, Manager& object) {
        cout << "Введіть логін: "; input >> object.login;
        return input;
    }
}

```

Файл «Container.h»

```

#pragma once
#include "Header.h"
#include "ErrorMessage.h"
#include "RangeMessage.h"
#include "ArrayMessage.h"
template <typename T> class Container;
template <typename T> ostream& operator<<(ostream&, const Container<T>&);
template <typename T> istream& operator>>(istream&, Container<T>&);

template <typename T> class Container
{
protected:
    T* info;
    int size;
public:
    Container();
    Container(int);
    friend ostream& operator<< <>(ostream&, const Container<T>&);
    friend istream& operator>> <>(istream&, Container<T>&);
    ~Container();
    int GetSize();
    T& operator[] (int);
};
template <typename T> Container<T>::Container() {
    info = nullptr;
    size = 0;
}
template <typename T> Container<T>::Container(int value) {
    if (value > 10) throw new ArrayMessage(value);

    info = new T[value];
    size = value;
}
template <typename T> Container<T>::~~Container() {
    delete[] info;
}
template <typename T> ostream& operator<<(ostream& output, const Container<T>& object) {
    if (object.info == nullptr) {
        output << "Список порожній" << endl;
    }
    else {
        for (int i = 0; i < object.size; i++) {
            output << object.info[i] << " ";
        }
        output << endl;
    }
    return output;
}
template <typename T> istream& operator>>(istream& input, Container<T>& object) {
    for (int i = 0; i < object.size; i++) {
        input >> object.info[i]; cout << endl;
    }
    return input;
}
template <typename T> T& Container<T>::operator[] (int index) {
    if (index > size - 1 or index < 0) {

```

```

        throw new RangeMessage(index);
    }
    return info[index];
}
template <typename T> int Container<T>::GetSize() { return size; }

```

Файл «Drivers.h»

```

#pragma once
#include "Container.h"
#include "Driver.h"
class Drivers : public Container<Driver>
{
public:
    Drivers();
    Drivers(int);
};

```

Файл «Drivers.cpp»

```

#include "Drivers.h"
Drivers::Drivers() : Container<Driver>() {}
Drivers::Drivers(int size) : Container<Driver>::Container(size) {}

```

Файл «Timetable.h»

```

#pragma once
#include "Route.h"
#include "Header.h"
#include "Container.h"
class Timetable: public Container<Route>
{
private:
    double fuelvalue;
public:
    Timetable();
    Timetable(int);
    double allfuel();
};

```

Файл «Timetable.cpp»

```

#include "Timetable.h"
Timetable::Timetable() : Container<Route>() {
    fuelvalue = 0;
}
Timetable::Timetable(int size) : Container<Route>::Container(size){
    fuelvalue = 0;
}
double Timetable::allfuel()
{
    for (int i = 0; i < size; i++) {
        fuelvalue += info[i].GetFuel();
    }
    return fuelvalue;
}

```

Файл «ErrorMessage.h»

```

#pragma once
#include "Header.h"
class ErrorMessage
{
protected:

```

```

        int index;
public:
    ErrorMessage(int);
    virtual void printmessage() = 0;
};

```

Файл «ErrorMessage.cpp»

```

#include "ErrorMessage.h"
ErrorMessage::ErrorMessage(int value) {
    index = value;
}

```

Файл «ArrayMessage.h»

```

#pragma once
#include "ErrorMessage.h"
class ArrayMessage :
    public ErrorMessage
{
public:
    ArrayMessage(int);
    void printmessage() override;
};

```

Файл «ArrayMessage.cpp»

```

#include "ArrayMessage.h"
ArrayMessage::ArrayMessage(int size):ErrorMessage(size) {}
void ArrayMessage::printmessage() {
    cout << "Масив більше ніж на 10 елементів створювати заборонено" << endl;
}

```

Файл «RangeMessage.h»

```

#pragma once
#include "ErrorMessage.h"
class RangeMessage:public ErrorMessage
{
public:
    RangeMessage(int);
    void printmessage() override;
};

```

Файл «RangeMessage.cpp»

```

#include "RangeMessage.h"
RangeMessage::RangeMessage(int value) :ErrorMessage(value) {}
void RangeMessage::printmessage() {
    cout << "Елементу за даним індексом не існує - " << index << endl;
}

```

Файл «Route.h»

```

#pragma once
#include "Auto.h"
#include "Driver.h"
class Route :
    public Auto, public Driver
{
private:
    struct date {
        int day;
    };
};

```



```

        int month;
        int year;
        date() {
            day = 0;
            month = 0;
            year = 0;
        }
        date(int dd, int mm, int yyyy) {
            day = dd;
            month = mm;
            year = yyyy;
        }
    };
    date dateofroute;
    string final_destination;
    double km;
    double fuelconsumption;
public:
    Route();
    Route(date, string, int, int, string, string, string, double, double, string, string,
double, double);
    friend ostream& operator<<(ostream&, const Route&);
    friend istream& operator>>(istream&, Route&);
    Route& operator=(const Route&);
    double GetFuel();
    void SetKilometers();
    void WriteStringData(string);
    void SetFuelConsumption();
};

```

Файл «Route.cpp»

```

#include "Route.h"
Route::Route() : Driver(), Auto(){
    dateofroute.day = 0;
    dateofroute.month = 0;
    dateofroute.year = 0;
    final_destination = " ";
    km = 0;
    fuelconsumption = 0;
}
Route::Route(date dateroute, string name, int number, int zp, string number1, string model1,
string type, double capacity1, double normalfuelrate1, string registr, string finaldest, double
kilon, double fuelconsum)
: dateofroute(dateroute), Driver(name, number, zp), Auto(number1, model1, type, capacity1,
normalfuelrate1, registr), final_destination(finaldest), km(kilon), fuelconsumption(fuelconsum) {}
ostream& operator<<(ostream& output, const Route& object) {
    output << "====Інформація про рейс: Харків - " << object.final_destination << "====" <<
endl;
    output << "Дата рейсу: " << object.dateofroute.day << "." << object.dateofroute.month <<
"." << object.dateofroute.year << endl;
    output << "====Інформація про водія====" << endl;
    output << "Прізвище та ініціали: " << object.fullname << endl;
    output << "Табельний номер: " << object.tabelnumber << endl;
    output << "Зарплата: " << object.salary << endl;
    output << "====Повна інформація про автомобіль====" << endl;
    output << "Державний номер авто: " << object.number << endl;
    output << "Модель: " << object.model << endl;
    output << "Тип: " << object.type_of_transport << endl;
    output << "Вантажопідйомність(тонн): " << object.capacity << endl;
    output << "Норма витрат пального(л/100км): " << object.normalfuelrate << endl;
    output << "Місце приписки: " << object.registration << endl;
    output << "Кілометраж: " << object.km << endl;
    output << "Розхід пального за рейс: " << object.fuelconsumption << endl;
}

```

```

        output << "===== " << endl;
        return output;
    }
    istream& operator>>(istream& input, Route& object) {
        cout << "====Введіть інформацію про рейс====" << endl;
        cout << "Дата рейсу: "; input >> object.dateofroute.day; cout << "."; input >>
object.dateofroute.month; cout << "."; input >> object.dateofroute.year; cout << endl;
        cout << "Призначте водія на цей рейс , введіть прізвище та ім'я без пробілів через
крапки "; input >> object.fullname; cout << endl;
        cout << "Табельний номер: "; input >> object.tabelnumber; cout << endl;
        bool flag = false;
        while (flag != true) {
            try {
                cout << "Зарплата(мінімальна 6000): ";
                input >> object.salary;
                if (object.salary < 6000) { flag = false; throw object.salary; }
                else { flag = true; }
            }
            catch (const int ex) { cout << ex << " це менше мінімальної зарплати, введіть інше
значення" << endl; }
        }
        object.SetAllatributes();
        cout << "Кінцевий пункт рейсу(Доступні міста на період воєнного стану - Київ, Запоріжжя,
Суми, Чернігів, Лозова) ";
        while (object.final_destination != "Київ" or object.final_destination != "Запоріжжя" or
object.final_destination != "Суми" or object.final_destination != "Чернігів" or
object.final_destination != "Лозова") {
            try {
                cout << "укажіть місто доставки: "; input >> object.final_destination;
                if (object.final_destination == "Київ" or object.final_destination ==
"Запоріжжя" or object.final_destination == "Суми" or object.final_destination == "Чернігів" or
object.final_destination == "Лозова") {
                    break;
                }
                else {
                    throw object.final_destination;
                }
            }
            catch (const string str) {
                cout << str << " це недоступне місто!" << endl;
            }
        }
    };
    object.SetKilometers();
    object.SetFuelConsumption();
    cout << "=====\nІнформація успішно введена!" << endl;

    return input;
}

Route& Route::operator=(const Route& object) {
    dateofroute.day = object.dateofroute.day;
    dateofroute.month = object.dateofroute.month;
    dateofroute.year = object.dateofroute.year;
    fullname = object.fullname;
    tabelnumber = object.tabelnumber;
    salary = object.salary;
    number = object.number;
    model = object.model;
    type_of_transport = object.type_of_transport;
    capacity = object.capacity;
    normalfuelrate = object.normalfuelrate;
    registration = object.registration;
    final_destination = object.final_destination;
    km = object.km;
}

```

```

        fuelconsumption = object.fuelconsumption;
        return *this;
    }

double Route::GetFuel()
{
    return fuelconsumption;
}

void Route::SetKilometers()
{
    if (final_destination == "Київ") {
        km = 480.4;
    }
    if (final_destination == "Суми") {
        km = 183.8;
    }
    if (final_destination == "Лозова") {
        km = 148.6;
    }
    if (final_destination == "Запоріжжя") {
        km = 297.4;
    }
    if (final_destination == "Чернігів") {
        km = 540.3;
    }
}

void Route::WriteStringData(string FILE)
{
    string str;
    ifstream fin;
    try
    {
        fin.open(FILE);
    }
    catch (const exception&) {}
    if (fin.peek() == ifstream::traits_type::eof())
    {
        str = "\nДата рейсу " + to_string(dateofroute.day) + "." +
to_string(dateofroute.month) + "." + to_string(dateofroute.year) + "\nВодій: " + fullname + "
табельний номер : " + to_string(tabelnumber) + "\nАвто : " + number + " модель : " + model + "
тип : " + type_of_transport + " вантажопідйомність : " + to_string(capacity) + " витрати
пального : " + to_string(normalfuelrate) + "\nКінцевий пункт рейсу : " + final_destination +
"\nВсього кілометрів : " + to_string(km) + "\nВитрати пального за рейс : " +
to_string(fuelconsumption);
    }
    else
    {
        str = "\n\nДата рейсу " + to_string(dateofroute.day) + "." +
to_string(dateofroute.month) + "." + to_string(dateofroute.year) + "\nВодій: " + fullname + "
табельний номер : " + to_string(tabelnumber) + "\nАвто : " + number + " модель : " + model + "
тип : " + type_of_transport + " вантажопідйомність : " + to_string(capacity) + " витрати
пального : " + to_string(normalfuelrate) + "\nКінцевий пункт рейсу : " + final_destination +
"\nВсього кілометрів : " + to_string(km) + "\nВитрати пального за рейс: " +
to_string(fuelconsumption);
    }
    fin.close();
    ofstream fout;
    try {
        fout.open(FILE, ofstream::app);
        fout << str;
    }
    catch (const exception&) {}
}

```

```

}

void Route::SetFuelConsumption()
{
    const double fuelrate = 34.5;
    fuelconsumption = ((km / 100) * fuelrate) * 2;
}

```

Файл «Header.h»

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string>
#include "ErrorMessage.h"
#include <iterator>
#include <Windows.h>
#include <cmath>
#include <fstream>
#include <vector>
using namespace std;

```

Файл «MainHeader.h»

```

#pragma once
#include "Header.h"
#include "Auto.h"
#include "Container.h"
#include "Driver.h"
#include "Order.h"
#include "Route.h"
#include "Timetable.h"
#include "Menu.h"
#include "Manager.h"
#include "Drivers.h"

```