

*КУРСОВА РОБОТА*

**КР.ІП –12.00.00.000 ПЗ**

**Група ІП-20-2**

**Кашуба Наталія**

2021

Міністерство освіти і науки України  
Івано-Франківський національний технічний університет нафти та газу  
Інститут інформаційних технологій

## КУРСОВИЙ ПРОЄКТ

з дисципліни «Основи об'єктно-орієнтованого програмування»  
на тему: Розробка прикладної програми «Моделювання та аналіз АІС  
автопідприємства міста»

Студентки 1 курсу групи ІП-20-2

спеціальності «Інженерія програмного забезпечення»

Кашуби Н.М.

Керівник

зав. кафедрою Шекета В.І.

Національна шкала \_\_\_\_\_

Кількість балів: \_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_

м. Івано-Франківськ



Міністерство освіти і науки України  
Івано-Франківський національний технічний університет нафти та газу  
Інститут інформаційних технологій

ЗАТВЕРДЖУЮ

зав. кафедри ІПЗ, проф., д.т.н.

\_\_\_\_\_ **В.І. Шекета**

«\_\_\_\_\_» \_\_\_\_\_2021р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

На курсовий проєкт з дисципліни «ООП» студенту Кашубі Н.М. групи ІП-20-2.

ТЕМА: Розробка прикладної програми з проєктування інформаційних систем згідно завдання.

Постановка задачі.

1. Розробити прикладну програму
2. Розробити інтерфейс користувача
3. Розробити та використати активні елементи
4. Розробити та використати шаблони не менше 5 шт.
5. Розрахувати швидкодію роботи з інтерфейсом користувача
6. Адаптувати розроблене прикладне програмне забезпечення до різних роздільних здатностей цифрових пристроїв.

Дата видачі «\_»\_2021р. Керівник \_\_\_\_\_

Завдання отримав \_\_\_\_\_

## ЗМІСТ

Анотація.....	2
Вступ.....	3
1. Аналіз сучасних тенденцій до оформлення проєкту з проєктування АІС.....	5
2. Положення меню.....	9
3. Розміщення активних та пасивних елементів на робочому столі.....	10
4. Прикладне програмування АІС та взаємодія користувача з нею.....	13
5. Особливості розробки активних програмних елементів.....	19
6. Особливості розробки активних програмних шаблонів.....	22
7. Моделювання та тестування інформаційної системи.....	23
7.1 Опис сучасного стану динамічних і статичних властивостей предметної області.....	23
7.2 Виділення та опис об'єктів в предметній в предметній області.....	27
Висновки.....	40
Література.....	42
Технічне завдання.....	43
Програмний код.....	44
Вміст текстових файлів, з яких було сформовано базу даних.....	99
Скріншоти етапів виконання.....	107

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## АНОТАЦІЯ

У цій курсовій роботі сформульовано основні теоретичні положення створення інформаційних систем. Викладено основні принципи проєктування інформаційного забезпечення. Розглянуто сучасні підходи до цієї проблеми. Висвітлено сучасний стан технологій розробки прикладного програмного забезпечення з проєктування інформаційних систем. Описано майже всі сучасні інформаційні технології для баз даних. Наведено класифікацію баз даних та інформаційних систем. Описано основні принципи об'єктно-орієнтованого програмування на прикладі об'єктно-орієнтованої мови програмування C++, таких як інкапсуляція, наслідування, асоціація тощо, та обґрунтовано доцільність їх застосування для розробки прикладних програм. Досліджено переваги vector та string перед звичайними масивами. Послідовно викладено переваги та доцільність застосування найбільш поширених типів даних для конкретних змінних. Змодельовано взаємодію між класами за допомогою діаграми класів UML. Наведено деякі особливості розроблення консольних додатків. Проведено аналіз предметної області (автопідприємства міста). Доведено доцільність автоматизації обліку автопідприємств. Розроблено базу даних автопідприємства та прикладну програму з консольним інтерфейсом, що реалізує базові потреби типового автопідприємства. Протестовано роботу прикладної програми для автопідприємства. Наведено приклади виконання цієї програми.

Ключові слова: інформаційні технології, C++, об'єктно-орієнтоване програмування, автоматизація, база даних, інформаційна система, прикладна програма, консольний інтерфейс, автопідприємство.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## ВСТУП

Розробка програмного забезпечення наприкінці XX ст. — на початку XXI ст. виділилася в окрему важливу галузь економіки — індустрію програмного забезпечення. Процес розробки комп'ютерних програм вимагає висококваліфікованої праці, і розвивається, загалом, повільніше, ніж процес вдосконалення апаратної бази комп'ютерів. Інженерія програмного забезпечення відносно недавно стала окремою професією.

Розробка програмного забезпечення містить у собі багато стадій: проєктування, програмування, тестування, впровадження і підтримку.

Проєктування починається із формулювання вимог до програмного забезпечення і створення специфікацій — документів, у яких описані функції, що їх повинна виконувати програма. На наступному етапі створюється загальний дизайн програми: розбиття її на окремі блоки та визначення взаємодії між ними. На етапі безпосереднього програмування створюється текстовий код програми на одній чи декількох мовах програмування. Після компіляції коду, програмний продукт обов'язково проходить тестування, у процесі якого визначається відповідність продукту специфікаціям, знаходяться і виправляються помилки.

Перед впровадженням програмний продукт потребує документації — опису можливостей, посібників користувача, системи допомоги. Після впровадження програмного забезпечення, що для програмних продуктів вимагає маркетингу, системи дистрибуції, реклами тощо, програмне забезпечення потребує підтримки. Необхідність у підтримці виникає внаслідок швидкого розвитку комп'ютерів, що зумовлює необхідність взаємодії програмного продукту з іншими, новішими програмами й новою матеріальною базою. Часто підтримка нових можливостей забезпечується випуском нових версій програмного продукту.

Завданням для цієї курсової роботи є розробка прикладного програмного забезпечення, що взаємодіє з базою даних типового автопідприємства міста. Для цього потрібно, щоб програма якимось чином взаємодіяла з базою даних, забезпечувала можливість швидкого пошуку автотранспорту чи працівників за певними ознаками, виводила на екран важливу інформацію. Не менш важливими за загальну функціональність є зручний інтерфейс та раціональне використання пам'яті.

Програма, розроблена у цій курсовій роботі, використовує консольний інтерфейс. Для забезпечення зручності користування програмою необхідно, щоб головне меню було чітким та зрозумілим, щоб у разі помилки виконання програми на екран виводилося повідомлення про те, чому саме програма не

Арк.

3

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

працює так, як очікувалося, наприклад, який саме файл не було знайдено. Також потрібно передбачити реакцію програми у разі введення користувачем неіснуючої команди.

Пам'ять повинна використовуватися якнайраціональніше, інакше програмне забезпечення буде неконкурентоспроможним. Для цього у курсовій роботі перед звичайними масивами надано перевагу таким динамічним структурам, як `string` та `vector`. Ще однією ключовою перевагою першого є те, що він спрощує роботу з текстом, надаючи широкий набір функцій для цього. Перевага другого в тому, що він, також, не має фіксованого розміру і розширюється автоматично при додаванні нового значення (через метод `push_back()`) і має додаткові можливості для роботи з масивами.

У цій роботі багато уваги також приділяється читабельності програмного коду. Адже дуже важливо, щоб назви класів, атрибутів та методів чітко відповідали їхньому призначенню. Це суттєво полегшує подальшу розробку, вдосконалення чи налагодження програми. Всі класи, атрибути та методи мають назви літературною англійською мовою, завдяки чому код є зрозумілим не лише для автора, а й для інших програмістів, що полегшить подальше вдосконалення чи використання.

Актуальність цієї курсової роботи зумовлена тим, що у типових автопідприємствах вся інформація про автотранспорт чи працівників не систематизована, а розрізнена в окремих файлах, через що керівництво автопідприємств не може своєчасно отримувати та опрацьовувати інформацію, а також змушене постійно виконувати одні й ті самі дії. Метою автоматизації є полегшення ведення обліку бізнес-процесів типового автопідприємства, а також створення єдиної бази даних та забезпечення виконання деяких базових запитів.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



## 1. АНАЛІЗ СУЧАСНИХ ТЕНДЕНЦІЙ ДО ОФОРМЛЕННЯ ПРОЄКТУ З ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ.

Розробка інформаційної системи (ІС) – від початкової фази до розгортання – складається з трьох послідовних і поступальних етапів: аналізу, проектування і реалізації.

Проектування ІС – логічно складна, трудомістка і тривала робота, що вимагає високої кваліфікації фахівців, що беруть участь в ній. Проте до теперішнього часу проектування ІС нерідко виконується на інтуїтивному рівні неформалізованими методами, що включають елементи мистецтва, практичний досвід, експертні оцінки і дорогі експериментальні перевірки якості функціонування ІС. Крім того, в процесі створення і функціонування ІС інформаційні потреби користувачів постійно змінюються або уточнюються, що ще більше ускладнює розробку і супровід таких систем.

Основна доля трудовитрат при створенні ІС доводиться на прикладне програмне забезпечення (ПЗ) і бази даних (БД). Виробництво ПЗ сьогодні – найбільша галузь світової економіки, в якій зайняті близько трьох мільйонів фахівців (програмістів, розробників ПЗ і т. п.).

В основі програмної інженерії лежить одна фундаментальна ідея: проектування ПЗ є формальним процесом, який можна вивчати і удосконалювати. Освоєння і правильне застосування методів і засобів створення ПЗ дозволять підвищити якість ІС, забезпечити керованість процесу проектування ІС і збільшити термін її життя.

Для успішної реалізації проекту об'єкт проектування (ПЗ) має бути передусім адекватно описаний, тобто мають бути побудовані повні і несуперечливі моделі архітектури ПО, що обумовлює сукупність структурних елементів системи і зв'язків між ними, поведінку елементів системи в процесів їх взаємодії, а також ієрархію підсистем, що об'єднують структурні елементи.

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Тенденції розвитку сучасних інформаційних технологій призводять до постійного зростання складності ІС, створюваних в різних областях діяльності. Сучасні великі проекти ІС характеризуються, як правило, наступними особливостями:

- 1) складність опису (досить велика кількість функцій, процесів, елементів даних і складні взаємозв'язки між ними), що вимагає ретельного моделювання і аналізу даних і процесів;
- 2) наявність сукупності тісно взаємодіючих компонентів (підсистем), що мають свої локальні завдання і цілі функціонування, використовують нерегламентовані запити до даних великого об'єму;
- 3) відсутність прямих аналогів, що обмежує можливість використання якихнебудь типових проектних рішень і прикладних систем;
- 4) необхідність інтеграції (композиції) існуючих і знову таких, що розробляються додатків;
- 5) функціонування в неоднорідному середовищі на декількох апаратних платформах;
- 6) роз'єднаність і різноманітність окремих груп розробників по рівню кваліфікації і традиціям використання тих або інших інструментальних засобів, що склалися.

Автопідприємство постійно змінюється, зростає, адаптується. Змінюються і бізнес-процеси. Відповідно з'являється необхідність створити якісну систему обліку та управління, яка б допомогла автопідприємству знизити операційні витрати. Основним призначенням програмного забезпечення для автопідприємства є автоматизація ведення обліку. Так, за кожним маршрутом закріплені окремі автобуси та маршрутні таксі. Том ведеться облік числа перевезених пасажирів, на підставі чого проводиться перерозподіл транспорту з одного маршруту на інший. Також береться до уваги обсяг вантажних перевезень

Арк.  
6

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

для вантажного транспорту, інтенсивність використання транспорту допоміжного характеру, число ремонтів і витрати на ремонт, інтенсивність роботи бригад по ремонту, число заміненних і відремонтованих вузлів агрегатів.

Спеціалізоване програмне забезпечення локального масштабу повинне автоматизувати та контролювати бізнес-процеси, зберігати інформацію про співробітників, автомобілі та об'єкти гаражного господарства. Дане програмне забезпечення сприятиме подальшому розвитку та масштабуванню автопідприємства, а також збільшенню прибутку. Це – комплексне рішення для автопідприємств, яке охоплює оперативний облік в управлінні перевезеннями та фінансами.

Можна виокремити наступні переваги автоматизації:

- ❖ контроль за ефективністю роботи водіїв;
- ❖ оцінка доцільності виконання замовлення;
- ❖ розрахунок вартості перевезення;
- ❖ оптимізація кількості співробітників;
- ❖ аналіз ефективності роботи обслуговуючого персоналу;
- ❖ контроль за рухом коштів;
- ❖ покращення якості обслуговування замовників послуг;
- ❖ скорочення витрат;
- ❖ можливість проводити складні розрахунки автоматично;
- ❖ локалізація всієї інформації про діяльність автопідприємства в одному місці та можливість використовувати її для прийняття поточних і стратегічних рішень.

Автоматизація особлива необхідна для автопідприємства за наявності таких факторів:

Арк.

7

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

- ❖ вся інформація розрізнена і знаходиться в окремих файлах;
- ❖ існує залежність виконання завдань від окремих співробітників;
- ❖ постійне дублювання одних і тих самих дій через відсутність доступу до поточних рахунків чи замовлень;
- ❖ керівники підприємства і його підрозділів не можуть своєчасно відслідковувати оперативну та аналітичну інформацію.

Отже, автопідприємство потребує якісного програмного продукту, в ідеалі – потрібно розробляти лінійку програмних продуктів для автоматизації. Ці продукти вже в базовій версії повинні покривати всі бізнес-процеси автопідприємств, а також гнучко підлаштовуватися під унікальну специфіку. Також важливі такі характеристики як

- ❖ зручний інтерфейс,
- ❖ можливість масштабування,
- ❖ розподіл прав користувачів,
- ❖ подальша підтримка та розвиток продукту,
- ❖ вартість володіння продуктом,
- ❖ висока продуктивність,
- ❖ можливість інтеграції з іншими інформаційними системами.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## 2. ПОЛОЖЕННЯ МЕНЮ.

Програма має меню, що відображає основні можливості взаємодії з інформаційною системою. Оскільки наша програма – це консольний додаток, то на екран буде виведено список доступних видів запитів. Деякі елементи меню міститимуть розгалуження, а деякі – ні. Наприклад, запит, що забезпечує вивід даних про автопарк підприємства, не потребує додаткових уточнень, тому він не міститиме розгалужень, а запит, що забезпечує вивід загального числа водіїв по підприємству, або для зазначеної автомашини, обов'язково повинен містити розгалуження, щоб користувач міг обрати чого саме він хоче – інформації про водіїв автопідприємства загалом, чи лише для конкретної автомашини.

Поряд з назвою і описом запиту необхідно вказати цифру або літеру, яка викличе виконання необхідного запиту. Наприклад, якщо реалізувати такі запити (отримати дані про автопарк підприємства; отримати перелік і загальне число водіїв по підприємству, або для зазначеної автомашини; отримати розподіл водіїв по автомобілях; отримати дані про розподіл пасажирського транспорту по маршрутах; отримати відомості про пробіг автотранспорту певної категорії або конкретної автомашини за вказаний день, місяць, рік), то було б добре, якби меню мало такий вигляд:

[0]Перегляд даних про автопарк підприємства

Перелік і загальне число водіїв

[a]по підприємству

[b]для конкретної автомашини

[1]Перегляд розподілу водіїв по автомобілях

Переглянути пробіг

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

[c]для категорії

[d]для конкретної автомашини

### 3. РОЗМІЩЕННЯ АКТИВНИХ ТА ПАСИВНИХ ЕЛЕМЕНТІВ НА РОБОЧОМУ СТОЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ.

Дана інформаційна система є, по суті, реляційною базою даних. Реляційна база даних — база даних, заснована на реляційній моделі даних. Слово «реляційний» походить від англ. relation.

Є три складові частини реляційної моделі даних:

- 1) структурна;
- 2) маніпуляційна;
- 3) цілісна.

Структурна частина моделі визначає, що єдиною структурою даних є відношення. Відношення зручно представляти у формі таблиць, де кожен рядок є кортеж, а кожен стовпець — атрибут, визначений на деякому домені. Даний неформальний підхід до поняття відношення дає більш звичну для розробників і користувачів форму представлення, де реляційна база даних подається як кінцевий набір таблиць.

Маніпуляційна частина моделі визначає два фундаментальних механізми маніпулювання даними — реляційну алгебру і реляційне числення. Основною функцією маніпуляційної частини реляційної моделі є забезпечення заходів реляційності будь-якої конкретної мови реляційних БД: мова називається реляційною, якщо вона має не меншу виразність і потужність, ніж реляційна алгебра або реляційне числення.

Цілісна частина моделі визначає вимоги цілісності сутностей і цілісності посилань. Перша вимога полягає в тому, що будь-який кортеж будь-якого відношення відмінний від будь-якого іншого кортежу цього відношення, тобто

Арк.  
10

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

іншими словами, будь-яке відношення має володіти первинним ключем. Вимога цілісності щодо посилань, або вимога зовнішнього ключа полягає в тому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, на яке веде посилання, повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (тобто ні на що не вказувати).

Можна провести аналогію між елементами реляційної моделі даних і елементами моделі «сутність-зв'язок». Реляційні відносини відповідають наборам сутностей, а кортежі — сутностям. Тому, як і в моделі «сутність-зв'язок», стовпці в таблиці, що представляє реляційне відношення, називають атрибутами.

Кожен атрибут визначений на домені, тому домен можна розглядати як множина допустимих значень даного атрибуту. Кілька атрибутів одних відношень і навіть атрибути різних відношень можуть бути визначені на одному і тому ж домені.

Переваги реляційної моделі:

- 1) простота і доступність для розуміння користувачем. Єдиною використовуваною інформаційною конструкцією є «таблиця»;
- 2) суворі правила проектування, які базуються на математичному апараті;
- 3) повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;
- 4) для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті.

Недоліки реляційної моделі:

- 1) далеко не завжди предметна область може бути представлена у вигляді «таблиць»;

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

- 2) в результаті логічного проектування з'являється множина «таблиць». Це призводить до труднощів розуміння структури даних;
- 3) БД займає відносно багато зовнішньої пам'яті;
- 4) відносно низька швидкість доступу до даних.

Отже, дані будуть виведені в консоль у вигляді таблиць. Для того, щоб таблиці коректно відображалися в консолі, необхідно застосувати керуючі символи. Керуючі символи (або як їх ще називають – escape-послідовність) – символи які виштовхуються в потік виводу, з метою форматування виводу або друку деяких керуючих знаків C++ . Всі керуючі символи, при використанні, обрамляються подвійними лапками, якщо необхідно вивести якесь повідомлення, то керуючі символи можна записувати відразу в повідомленні, в будь-якому його місці. Керуючі символи C – це не основний спосіб форматowanego виведення, але найбільш простий і найбільш часто використовуваний. Найчастіше використовуються такі керуючі символи: \t (горизонтальна табуляція), \n (новий рядок). Саме їх, а ще \v (вертикальна табуляція) і використано у цій курсовій роботі.

Кожен стовпець називатиме властивість об'єкта, а в рядках міститимуться дані про об'єкти. Інколи для того, щоб забезпечити нормальні відступи між даними необхідно використати не одну табуляцію, а дві чи навіть три, і додати відступи. Також було б добре встановити вирівнювання по лівому краю за допомогою `setwiosflags(ios::left)`.

Щоб дані, що виводяться в консоль мали справді вигляд таблиці варто застосувати символи | та \_ . Отже, спочатку виводимо рядок з символів нижнього підкреслення, який буде свідчити про початок таблиці. Кожен рядок починається і закінчується символом |. У першому рядку вказано назви властивостей. Після кожного рядка йде рядок з символів \_ . Після рядка з назвами властивостей варто застосувати вертикальну табуляцію. Всі інші рядки просто починатимуться з нового рядка (\n).

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



Отже, таблиця даних про автомобілі матиме приблизно такий вигляд.

Номер автомобіля	Колір	Рік виробництва	Споживає л бензину на 100 км
АТ 4501	жовтий	2015	18,5
АТ 2485	чорний	1995	16,7

#### 4. ПРИКЛАДНЕ ПРОГРАМУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ВЗАЄМОДІЯ КОРИСТУВАЧА З НЕЮ.

Інформаційна технологія передбачає поєднання апаратного і програмного забезпечення. Під програмним забезпеченням інформаційних комп'ютерних технологій розуміють сукупність програмних і документальних засобів для створення та експлуатації систем обробки даних засобами обчислювальної техніки.

Залежно від функцій, які виконує програмне забезпечення, його можна поділити на дві групи: базове (системне) програмне забезпечення і прикладне програмне забезпечення. Базове ПЗ організує процес обробки інформації в комп'ютері і забезпечує відповідне робоче середовище для прикладних програм. Базове ПЗ тісно пов'язане з апаратними засобами, його інколи вважають частиною комп'ютера.

Прикладні системи утворюють рівень програмного забезпечення, що надається користувачеві для розв'язання своїх задач. Процедури інформаційних технологій спрямовуються на обробку інформації певного класу (даних, тексту, графіки, об'єктів реального світу) і реалізуються за допомогою програмних комплексів різного рівня, складності та призначення.

Прикладне програмне забезпечення призначене для розв'язування конкретних задач користувача й організації обчислювального процесу інформаційної системи загалом. На відміну від програмістів, користувачів прикладного ПЗ називають кінцевими користувачами, припускаючи, що саме вони і є кінцевими

Арк.  
13

Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ

користувачами тих знань, які зосереджені в пам'яті комп'ютера або можуть генеруватися під час роботи прикладних програм. Звертаючись до прикладної системи, користувачеві інколи доводиться виконувати деякі прості операції - вводити числа і тексти, Переглядати дані, виводити графіки, малюнки на екран дисплея та на зовнішні пристрої. Прикладні системи конструюються таким чином, щоб створити людині максимальний комфорт під час виконання таких дій і при цьому не вимагати від неї надзвичайно великих навичок та спеціальних знань. Прикладне ПЗ працює під управлінням базового програмного забезпечення, зокрема операційних систем.

Пакети прикладних програм (ППП) - це комплекс програм, призначений для розв'язування задач певного класу. Розрізняють кілька основних класів прикладних систем, що використовуються на персональних комп'ютерах: прикладні пакети і програми загального призначення (універсальні); пакети і програми проблемозорієнтовані; глобальних мереж; методозорієнтовані; організації (адміністрування) обчислювального процесу.

До пакетів і програм загального призначення, що їх особливо широко використовують у сфері управлінської та організаційної діяльності, належать: текстові процесори; пакети графічного подання даних; табличні процесори; системи управління базами даних; інтегровані пакети; CASE-технології; оболонки експертних систем і систем штучного інтелекту, системи підтримки комунікацій.

Для створення внутрішньомашинного інформаційного забезпечення використовуються спеціальні PPP - системи управління базами даних - СУБД. База даних - це сукупність спеціальним чином організованих наборів даних, що зберігаються на диску. СУБД забезпечує управління базою даних і передбачає введення даних, їх редагування, маніпулювання даними, тобто вилучення, поновлення тощо. Розвинуті СУБД забезпечують незалежність прикладних

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

програм, що працюють з ними, від конкретної організації інформації в базі даних.

Інтегровані системи утворюють особливу категорію програмного забезпечення. Типова інтегрована система включає текстовий процесор, системи управління базами даних, засоби роботи з таблицями, пакет ділової графіки і засоби підтримки комунікацій. Головну увагу розробники цих систем приділяють тому, щоб користувач використовував у роботі в різних середовищах інтегрованого пакета приблизно ті самі прийоми роботи і міг швидко перейти з однієї групи операцій на іншу. Ще одна суттєва вимога - простота дій користувача під час розв'язання простих задач, з якими він стикається, і звернення до складних варіантів роботи у виняткових випадках. Один із перспективних підходів - надати користувачам не готові інтегровані системи, а зручні засоби для їх створення. Такі засоби трактуються як надбудови над операційними системами, що дозволяють з'єднувати кілька прикладних пакетів у рамках зручного для користувачів операційного середовища.

CASE-технології застосовуються при створенні складних інформаційних систем, що потребують колективної реалізації проєкту, в якому беруть участь різні спеціалісти: системні аналітики, проєктувальники і програмісти. Під CASE-технологією розуміється сукупність засобів автоматизації розробки інформаційної системи, що містить методологію аналізу предметної області, проєктування, програмування й експлуатації інформаційної системи. У нинішній час CASE-технологія - одна з галузей інформатики, що найбільш динамічно розвивається й об'єднує сотні компаній. Сучасні CASE-технології успішно застосовуються для створення ІС різного класу: банки, фінансові корпорації, великі фірми. Економічний ефект застосування CASE-технологій досить значний і більшість сучасних програмних проєктів здійснюється саме за їх допомогою.

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Експертні системи (ЕС) - це системи обробки знань у вузькоспеціалізованій сфері підготовки рішень користувачів на рівні професійних експертів. Основу ЕС становить база знань, в яку закладається інформація про конкретну предметну область. Є дві основні форми представлення знань в ЕС: факти і правила. Факти фіксують кількісні та якісні показники явищ і процесів. Правила описують співвідношення між фактами у формі логічних умов, що пов'язують причини і результати. ЕС використовуються для інтерпретації і діагностики стану системи, прогнозування ситуацій у системах управління процесом та усунення порушень функціонування системи. Як засоби реалізації ЕС на ЕОМ використовують так звані оболонки експертних систем.

Проблемозорієнтовані пакети і програми, на відміну від програм загального призначення, зорієнтовані на спеціалістів певного профілю або вузького кола застосувань. Кількість таких програм для персональних комп'ютерів нині становить кілька тисяч. Практично немає жодної предметної області, для якої не існує хоча б одного ППП. Усі проблемозорієнтовані ППП можна поділити на групи, що призначені для комплексної автоматизації функцій управління у промисловій і непромисловій сферах та ППП предметних областей. Проблемозорієнтовані пакети непромислової сфери призначені для автоматизації діяльності банків, бірж, торгівлі тощо. ППП окремих предметних областей - це ППП бухгалтерського обліку, фінансового менеджменту, правових довідкових систем.

Мето дезорієнтовані ППП відзначаються тим, що в їхній алгоритмічній основі реалізовано певний економіко-математичний метод розв'язування задач. До них відносять ППП математичного програмування, математичної статистики, сіткового планування й управління тощо.

Програма, що розроблена у цій курсовій роботі, є примітивною СУБД, але простою в користуванні і такою, що забезпечує базові потреби узагальненого автопідприємства. Якби програма володіла графічним інтерфейсом, то

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

користуватися нею було б ще зручніше і звичніше. Оскільки програма не володіє графічним інтерфейсом, то вона працює в режимі консолі. Інтерфейс командного рядка — різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків (команд). Також відомий під назвою консоль. Інтерфейс командного рядка може бути протиставлений системам управління програмою на основі меню чи різних реалізацій графічного інтерфейсу. Формат виводу інформації в інтерфейсі командного рядка не регламентується; звичайно це простий текстовий вивід, але може бути й графічним, звуковим виводом тощо. У цій програмі інформація виводиться лише у вигляді тексту.

Отже, після запуску програми на виконання на екран буде виведено меню, тобто перелік можливих дій. Поряд з кожним елементом меню буде вказано символ (цифру або літеру), який йому відповідає. Щоб викликати якийсь запит на виконання користувачу необхідно ввести відповідний символ і натиснути Enter. Якщо користувач введе некоректний символ, тобто такий, якого немає в меню, на екран буде виведено повідомлення про помилку.

Реалізація деяких запитів передбачає введення ключових слів для пошуку, наприклад, якщо потрібно отримати кількість водіїв для зазначеної автомашини. У таких випадках на екрані буде написано, що саме користувач повинен ввести, щоб отримати очікуваний результат. Після введення ключового слова для пошуку треба натиснути Enter. У разі помилкового введення даних для пошуку на екран буде виведено повідомлення про помилку. Якщо дані введені коректно, то на екран буде виведено дані за пошуковим запитом у формі таблиці.

Після виконання запиту або виявлення помилки введення даних для пошуку користувач має можливість повернутися до головного меню, натиснувши клавішу, що відповідає цій команді і Enter. У такому разі результати виконання попереднього запиту або повідомлення про помилку введення даних для пошуку будуть стерті з екрану, щоб не відволікати користувача вже зайвою інформацією.

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Це суттєво полегшить користування програмою. Якщо не очищувати екран, то користувач може заплутатися через велику, і навіть надлишкову кількість інформації.

Тому для забезпечення простоти розуміння і сприйняття краще або щоразу очищувати екран, або виводити результати виконання запитів не на екран, а в окремі файли, наприклад текстові. У цій курсовій використано перший спосіб, оскільки другий, на думку автора, має суттєвий недолік – інформація, що потрібна автопідприємству зберігатиметься в різних файлах, що суперечить самій меті автоматизації бізнес-процесів автопідприємства – зробити так, щоб уся інформація не була розрізнена і не знаходилася в окремих файлах.

Отже, наша програма, хоч і є консольною, проте добре продумана, легка в користуванні і не вимагає від користувача спеціальних знань чи високого рівня володіння комп'ютером. Програма навіть не потребує інструкції, адже вказівки про те, що необхідно ввести завжди будуть на екрані.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## 5. ОСОБЛИВОСТІ РОЗРОБКИ АКТИВНИХ ПРОГРАМНИХ ЕЛЕМЕНТІВ.

У цій курсовій роботі активними програмними елементами є об'єкти класів. Щоб створити якісні класи, необхідно чітко розуміти принципи ООП (об'єктно-орієнтованого програмування). ООП – це одна з парадигм програмування, яка розглядає програму як множину об'єктів, що взаємодіють між собою. Однією з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Основу ООП складають чотири основні концепції:

- 1) інкапсуляція,
- 2) успадкування,
- 3) поліморфізм,
- 4) абстракція.

Інкапсуляція — один з трьох основних механізмів об'єктно-орієнтованого програмування. Йдеться про те, що об'єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів (методів). А також про те, що доступ до стану об'єкта напрямку заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи), що дозволяє знизити зв'язність. Таким чином контролюються звернення до полів класів та їхня правильна ініціалізація, усуваються можливі помилки пов'язані з неправильним викликом методу. Оскільки користувачі працюють лише через

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

відкриті елементи класів, то розробники класу можуть як-завгодно змінювати всі закриті елементи і навіть перейменовувати та видаляти їх, не турбуючись, що дець хтось їх використовує у своїх програмах.

Успадкування (наслідування) — механізм утворення нових класів на основі використання вже існуючих. При цьому властивості та функціональність батьківського класу переходять до класу нащадка (дочірнього).

Поліморфізм — концепція в програмуванні та теорії типів, в основі якої лежить використання єдиного інтерфейсу для різнотипних сутностей або у використанні однакового символу для маніпуляцій над даними різного типу.

Абстракція – це додання об'єкту характеристик, які відрізняють його від всіх об'єктів, чітко визначаючи його концептуальні межі. Основна ідея полягає в тому, щоб відокремити спосіб використання складових об'єктів даних від деталей їх реалізації у вигляді більш простих об'єктів, подібно до того, як функціональна абстракція розділяє спосіб використання функції і деталей її реалізації в термінах більш примітивних функцій, таким чином, дані обробляються функцією високого рівня за допомогою виклику функцій низького рівня.

Саме ці концепції й будуть використані у цій курсовій роботі, зокрема інкапсуляцію та успадкування. На думку автора, це найважливіші концепції, адже перша забезпечує надійність, безпеку даних та їх захист від неправильного використання, що може призвести до серйозних помилок роботи програми, а друга – дозволяє зекономити багато часу, адже замість створювати знову й знову класи з тими самими властивостями й методами програміст може просто утворити нові класи-нащадки. Також варто зазначити, що завдяки наслідуванню програмний код є компактнішим, більш читабельним, і його значно легше налагодити. У той же час дві інші концепції, а саме абстракція та поліморфізм, на думку автора, є корисними, звісно, проте необов'язковими. Тобто без них

Арк.  
20

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



можна обійтися, хоча звісно ж, краще все-таки чітко розуміти й застосовувати всі можливості об'єктно-орієнтованого програмування мовою C++.

Отже, у цій курсовій роботі буде створено кілька базових загальних класів, від яких будуть походити інші, більш спеціалізовані класи-нащадки. Всі змінні будуть інкапсульованими, тобто захищеними від будь-яких випадкових змін, а методи (функції) будуть відкритими і служитимуть для зв'язку з захищеними членами. Лише абстрактні класи матимуть нащадків, оскільки, на думку автора, краще робити так, щоб клас був або фінальним, або абстрактним, хоча є програмісти, які й не погоджуються з цією думкою. Це просто питання смаку, і однозначної відповіді тут не існує. Інкапсуляція здійснюється за допомогою одного з трьох специфікаторів доступу (public, protected, public), вибір яких обгрунтовано в наступних розділах.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## 6. ОСОБЛИВОСТІ РОЗРОБКИ АКТИВНИХ ПРОГРАМНИХ ШАБЛОНІВ.

Робота будь-якої інформаційної системи забезпечується наявністю базових функцій. Наприклад, функції сортування, пошуку, заповнення бази даних, виведення повідомлень про помилки, виведення на екран окремих елементів та їхню кількість, додавання нових елементів та виведення інших, тих, які вже не потрібні.

Для зберігання цих методів можна створити окремий клас в окремому файлі і назвати його, наприклад, Functions, а можна створити ці методи у відповідних їм класах. У цій курсовій роботі використано другий спосіб, тому що на думку автора, програма не є настільки складною, що потребує кількох програмних файлів. До того ж, кожен клас може мати свої специфічні методи, і якщо всі ці методи зібрати в один клас, то розробник зможе заплутатися в реалізаціях. Варто зазначити, що це лише авторське бачення.

Отже, окрім властивостей, класи мають ще й методи. Метод — підпрограма (процедура, функція), що використовується виключно разом із класом (методи класу) або з об'єктом (методи екземпляра). Залежно від впливу на стан об'єкта, методи поділяються на:

- 1) конструктори — встановлюють початковий стан об'єкта;

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

- 2) деструктори — скидають стан об'єкта;
- 3) селектори (геттери) — надають значення атрибута;
- 4) модифікатори (сеттери) — встановлюють значення атрибута;
- 5) ітератори — надають послідовний доступ до множини атрибутів.

Окрім цих методів інформаційна система має мати ще й методи обробки помилок введення даних, для можна створити окремий загальний клас Exceptions, що має похідні класи, більш спеціалізовані. Забезпечення можливості для користувача проаналізувати помилки взаємодії з програмою є одним з ключових пунктів, на думку автора. Якщо не забезпечити аналізу помилок, то взаємодія користувача з системою суттєво ускладниться.

## 7. МОДЕЛЮВАННЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.

### 7.1 Опис сучасного стану динамічних і статичних властивостей предметної області.

Перш ніж приступити до безпосереднього моделювання інформаційної системи необхідно проаналізувати предметну область. Предметна область — це частина реального світу, що розглядається в межах певного контексту. Під контекстом можна розуміти область дослідження чи область, яка є об'єктом певної діяльності. Предметна область представляється безліччю фрагментів. Усі фрагменти предметної області безліччю об'єктів і процесів, що використовують об'єкти, і безліччю користувачів, які мають різні погляди на предметну область.

Для аналізу предметної області вибрано узагальнення автопідприємства, тобто таке автопідприємство, якому притаманні всі основні ознаки і атрибути автопідприємств. Це підприємство займається пасажирськими та вантажними перевезеннями. Автопідприємство має свій автотранспорт різного призначення, свій штат водіїв та обслуговуючий персонал, а також об'єкти гаражного

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

господарства. Відділи автопідприємства можуть знаходитися в різних містах України.

Основними видами транспорту є:

❖ пасажирський транспорт:

➤ автобуси:

- міські,
- приміські,
- місцевого сполучення,
- міжміські,
- туристичні,
- екскурсійні,
- мікроавтобуси,
- тролейбуси,
- міжнародного сполучення;

➤ таксі:

- вуличні,
- маршрутні,
- вантажні,
- мототаксі;

❖ вантажний транспорт:

- вантажні автомобілі,
- автомобілі-тягачі,

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

- причеи,
- напівпричеи;
- ❖ транспорт спеціального призначення:
  - автокрани,
  - грейдери,
  - екскаватори,
  - бульдозери,
  - вантажні тролейбуси,
  - котки,
  - авто з ліцензією ADR для перевезення легкозаймистих вантажів.

З плином часу автотранспорт старіє і списується або продається, але й автопідприємство постійно поповнюється новим автотранспортом.

Підприємство має штат водіїв, закріплених за автомобілями. За одним автомобілем може бути закріплено декілька водіїв. Кожен водій має посвідчення одного або кількох категорій:

- ❖ А,
- ❖ В,
- ❖ С1,
- ❖ С,
- ❖ D1,
- ❖ D,
- ❖ BE,

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

- ❖ C1E,
- ❖ CE,
- ❖ D1E,
- ❖ DE,
- ❖ T.

З часом водії звільняються або виходять на пенсію. З іншого боку, автопідприємство постійно приймає нових водіїв у зв'язку з придбанням нових транспортних засобів.

Обслуговуючий персонал займається технічним обслуговуванням автомобільного транспорту. До обслуговуючого персоналу належать

- ❖ техніки,
- ❖ зварники,
- ❖ слюсарі,
- ❖ складальники.

Обслуговуючий персонал і водії об'єднуються в бригади, якими керують бригадири, бригадирами керують майстри, майстрами – начальники ділянок і цехів.

Автопідприємство має власні об'єкти гаражного господарства, де міститься й ремонтується автомобільна техніка. Об'єкти гаражного господарства включають в себе

- ❖ цехи,
- ❖ гаражі,
- ❖ бокси.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## 7.2 Виділення та опис об'єктів в предметній в предметній області.

Для реалізації завдання потрібно ввести спеціалізовану систему класів. Існує два способи введення системи класів: можна для кожного виду транспорту створити окремий клас, не використовуючи наслідування, або створити загальний базовий клас, що зберігатиме інформацію про різні види транспорту. У цій курсовій роботі використано другий спосіб, оскільки він більш універсальний та зрозуміліший.

Наслідування — це механізм утворення нових класів на основі використання вже існуючих. При цьому властивості та функціональність батьківського класу переходять до похідного класу. Визначення нового класу базується на визначенні вже існуючого. Новий клас отримує властивості та поведінку батьківського. Дочірній клас також може бути доповнений власними властивостями та методами. Застосування цього механізму дозволяє покращити повторне використання коду шляхом використання вже визначених властивостей та методів базового класу.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Базовим класом є клас **Vehicle** (транспортний засіб). Цей клас описує транспорт загалом, а не якийсь конкретний вид транспорту. Тому варто зробити клас Vehicle абстрактним. Абстрактний клас – це базовий клас, від якого не можна створити екземпляру. Абстрактний клас може бути реалізовано за допомогою поліморфізму. На думку автора, бажано створювати класи так, щоб вони були або абстрактними, або не мали похідних. Для того, щоб клас був абстрактним, необхідно, щоб він містив принаймні один чистий віртуальний метод, тобто такий метод, який є різним для батьківського і похідного класів. Нехай у класі Vehicle таким методом буде count().

Клас Vehicle містить поля даних для зберігання марки, кольору, року виробництва, реєстраційного номеру, інформацію про те, скільки бензину споживає цей транспортний засіб на 100 км, а також методи для отримання та встановлення значень відповідних властивостей. Для зберігання назви марки використовуємо поле brand типу string, для зберігання кольору використовуємо поле color типу string, для зберігання реєстраційного номеру використовуємо поле number типу string, для зберігання року виробництва – поле year\_of\_production типу unsigned int. Інформацію про споживання бензину зберігаємо в полі petrol типу float.

String – це стандартний клас, який представляє текстовий рядок. Серед переваг цього класу варто відзначити те, що він поміщає важку логіку управління пам'яттю на клас string, а не на програміста. Клас string реалізує типові операції з рядками. Наприклад, порівняння, конкатенація, пошук і заміна, отримання підрядків.

Тип unsigned int призначений для зберігання невід'ємних цілочислових даних, тобто представляє позитивне ціле число. Залежно від архітектури процесора, може займати 2 байти (16 біт) або 4 байти

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



(32 біти), і через це діапазон граничних значень може змінюватися: від 0 до 65535 (для 2 байт), або від 0 до 4 294 967 295 (для 4 байт).

Оскільки споживання бензину не завжди можна виразити цілим числом, використаємо тип float. Тип float представляє дійсне число з плаваючою комою в діапазоні +/- 3.4E-38 до 3.4E + 38. У пам'яті займає 4 байта (32 біта).

C++ дозволяє інкапсулювати дані. Інкапсуляція (або «приховування інформації») – це процес прихованого зберігання деталей реалізації об'єкта. Користувачі звертаються до об'єкта через відкритий інтерфейс. Абстракція даних – це механізм викриття тільки інтерфейсів і приховування деталей реалізації від користувача. інкапсуляція реалізована через специфікатор доступу. Як правило, всі змінні-члени класу є закритими, а більшість методів є відкритими (з відкритим інтерфейсом для користувача).

Інкапсуляція має багато переваг:

- 1) інкапсульовані класи простіші у використанні і зменшують складність програм;
- 2) інкапсульовані класи допомагають захистити дані і запобігають їх неправильному використанню;
- 3) інкапсульовані класи легше змінити;
- 4) інкапсульовані класи легше налагодити.

Ключовою перевагою є те, що нам не потрібно знати, як клас влаштований всередині. Клас Vehicle, який зберігає інформацію про реєстраційний номер, колір, марку, інтенсивність споживання бензину може бути реалізований за допомогою динамічних масивів, рядків C-style, array, vector тощо. Для використання цього класу нам не потрібно

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

знати деталі його реалізації. Це значно знижує складність програми, а також зменшує кількість можливих помилок. Це і є основною перевагою інкапсуляції.

Також варто пам'ятати, що глобальні змінні небезпечні, оскільки немає строгого контролю над тим, хто має до них доступ і як їх використовують. Ця сама проблема стосується класів з відкритими членами. Якби змінні класу Vehicle були відкритими, їхні значення легко могли б бути переприсвоєні в будь-якому місці програми. Якщо зробити їх закритими, то користувач буде змушений використовувати методи для взаємодії з класом. Це захищає змінні від неправильного використання.

Інкапсуляція надає можливість зміни способу реалізації класів, не порушуючи при цьому роботу всіх програм, які їх використовують. Так, якщо змінити реалізацію класу, не змінивши прототипів функцій з відкритого інтерфейсу, то програма, що використовує клас, буде й далі працювати без будь-яких змін чи проблем.

Також інкапсуляція допомагає налагодити програму в разі несправності роботи. Однією з поширених причин неправильної роботи програми є некоректне значення однієї із змінних. Якщо кожен об'єкт має прямий доступ до змінної, то відстежити частину коду, яка змінила змінну, може бути досить важко. Однак, якщо для зміни значення потрібно викликати один і той самий метод, можна просто проаналізувати цей метод і кожне місце програми, де він викликається.

Інкапсуляція здійснюється специфікаторами доступу. Специфікатор доступу визначає, хто має доступ до членів цього специфікатору. Кожен з членів має свій рівень доступу відповідно до специфікатора

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

доступу (або, якщо він не вказаний, відповідно до специфікатора доступу за замовчуванням). У мові C++ є три рівні доступу:

- 1) private,
- 2) protected,
- 3) public.

Перший робить члени закритими, другий – захищеними, тобто доступними для похідних класів, третій – відкритими. Закриті члени (або «private-члени») – це члени класу, доступ до яких мають лише інші члени цього ж класу. Оскільки функція `main()` не є членом класу `Vehicle`, то вона і не має доступу до його закритих членів. Відкриті члени (або «public-члени») – це члени класу, до яких можна отримати доступ за межами цього класу. Якщо зробити члени класу `Vehicle` публічними, то до них можна буде безпосередньо звертатися з функції `main()`. Рекомендується встановлювати специфікатор доступу `private` змінним-членам класу і специфікатор доступу `public` – методам класу (якщо немає вагомих підстав робити інакше). Оскільки для правильного функціонування програми необхідно передати поля дочірнім класам, то для цих полів, варто встановити специфікатор `protected`. Специфікатор доступу `protected` відкриває доступ до членів класу дружнім і дочірнім класами. Доступ до `protected`-члену поза тілом класу закритий.

Клас `Vehicle` – абстрактний клас, тобто він не містить членів змінних, які повинні належати лише цьому класу. Отже, усі поля класу `Vehicle` захищені (`protected`), тобто доступні лише методам цього класу, та класам, які походять від `Vehicle`. Специфікатор `protected` у цьому випадку дуже корисний, оскільки кількість класів, що походять від класу `Vehicle` є відносно невеликою, тобто якщо доведеться внести

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

зміни в реалізацію батьківського класу і відповідно похідних класів, то це не займе багато часу. На думку автора, варто спочатку перерахувати protected-члени класу Vehicle, а потім його public-члени, оскільки public-члени використовуватимуть protected-члени.

Методи доступу класу Vehicle – публічні. Їх можна використовувати за межами класу. Функція доступу – це коротка функція призначена для отримання або зміни значення закритої змінної. Функції доступу поділяються на геттери і сеттери. Геттери - це функції, які повертають значення закритих змінних-членів класу. Сеттери - це функції, які дозволяють присвоювати значення закритим змінним-членам класу. Функції доступу варто створювати лише в тому випадку, коли справді необхідно, щоб користувач міг отримувати або змінювати значення. У нашому випадку це потрібно, тому варто створити повний набір геттерів і сеттерів:

- 1) string get\_number(),
- 2) void set\_number(string nmb),
- 3) string get\_color(),
- 4) void set\_color(string clr),
- 5) unsigned int get\_year\_of\_production(),
- 6) void set\_year\_of\_production(unsigned int year),
- 7) float get\_petrol(),
- 8) void set\_petrol(float ptrl).

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Від базового класу **Vehicle** походять такі класи: **Bus** (автобус), **Taxi** (таксі), **TouristicCoach** (туристичний автобус), **Truck** (вантажівка), **AuxiliaryTransport** (транспорт допоміжного характеру).

Є три типи наслідування класів:

- 1) private,
- 2) protected,
- 3) public.

Якщо не обрати типу наслідування, то він буде визначений за замовчуванням (private), як і члени класу що за замовчуванням є приватними (private). У разі закритого наслідування всі члени базового класу успадковуються як закриті. Це означає, що private-члени залишаються недоступними, а protected- і public-члени стають private в дочірньому класі. Закрите наслідування могло б бути корисним, якби похідний клас не мав очевидного зв'язку з базовим, але використовував його у своїй реалізації. У такому випадку не бажано, щоб відкритий інтерфейс базового класу був доступний через об'єкти похідного класу. У разі protected-наслідування public- і protected-члени стають захищеними (protected), а private-члени залишаються недоступними. Найпоширенішим типом наслідування є public-наслідування. Коли відкрито успадковується батьківський клас, то успадковані public-члени залишаються public, успадковані protected-члени залишаються protected, а успадковані private-члени залишаються недоступними для дочірнього класу. Рекомендовано використовувати саме цей тип наслідування, якщо немає вагомих причин робити по-іншому. Тому в цій курсовій роботі використано тип успадкування public.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Кожен похідний клас може мати властивості та методи, які притаманні лише йому. Так, звичайний автобус (Bus) характеризується кількістю пасажирських місць і маршрутом. Оскільки місткість – це завжди ціле додатне число, то для змінної `passengerseats` варто обрати тип `unsigned int`. Маршрут (`route`) варто подати у вигляді `string`. Клас Bus є фінальним, тобто не має нащадків, тому варто зробити його поля приватними (за допомогою специфікатора `private`). Для забезпечення можливості переглядати та редагувати значення полів об'єктів класу Bus необхідно додати публічні функції доступу до його змінних. Це `unsigned int get_pessengerseats()`, `void set_pessengerseats(unsigned int seats)`, `string get_route()` і `void set_route(string rout)` відповідно.

Клас `TouristicCoach` описує туристичний автобус. Такий автобус не має маршруту, а характеризується лише місткістю, як і Bus, і вартістю годинної оренди. Місткість позначимо змінною `pessengerseats` типу `unsigned int`. Для доступу до цієї приватної змінної використано публічні методи `unsigned int get_pessengerseats()` та `void set_pessengerseats(unsigned int seats)`. Вартість оренди за годину описує приватна змінна `moneyperhour` типу `float`, для встановлення та отримання значення якої створено публічні функції `float get_moneyperhour()` та `void set_moneyperhour(float money)`.

Таксі використовується для перевезення пасажирів і вантажів, тому до класу `Taxi` варто додати властивості `passengerseats` типу `unsigned int`, що описує кількість пасажирських місць, і `kg` типу `float`, що описує вантажопідйомність. Оскільки ці змінні приватні, то для роботи з ними необхідні функції доступу `unsigned int get_pessengerseats()`, `void set_pessengerseats(unsigned int seats)`, `float get_kg()`, `void set_kg(float kh)` відповідно.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

За своєю конструкцією та обладнанням вантажівки призначені для перевезення вантажів. За вантажністю вони поділяються на

- 1) особливо малої вантажності — до 1 тони;
- 2) малої вантажності — 1-3,5 тони;
- 3) середньої вантажності — понад 3,5-15 тони;
- 4) великої вантажності — понад 15 тони;
- 5) особливо великої вантажності — понад норми, встановленої дорожніми габаритами та ваговими обмеженнями.

Тому унікальною властивістю класу `Truck` є вантажопідйомність. Вантажопідйомність вимірюється в тоннах, тому відповідну приватну змінну назвемо `tons`. Типом цієї змінної доцільно обрати `float`, оскільки вантажопідйомність не завжди можливо точно описати цілим числом. Змінна `tons` – закрита, тому створимо функції доступу `float get_tons()`, `void set_tons(float ton)` для отримання та встановлення значення змінної `tons` відповідно.

До транспорту допоміжного характеру належать евакуатори. Евакуатори характеризуються вантажопідйомністю, а також довжиною та шириною платформи. Їх описано в класі `AuxiliaryTransport`. Вантажопідйомність вимірюється в тоннах, тому вона буде позначатися змінною `tons` типу `float` і матиме функції доступу `float get_tons()`, `void set_tons(float ton)`. Перша призначена для присвоєння змінній `tons` конкретного значення, а друга для отримання цього значення. Довжина та ширина платформи вимірюються в метрах. Довжину позначимо `length`, а ширину – `width`. Обидві змінні приватні, тому для обидвох потрібно створити методи доступу. Для змінної `length` це `float get_lenght()` і `void set_lenght(float m)`. Для змінної `width` це `float get_width()` і `void set_width(float m)`.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Ще одним базовим класом є клас **Worker**. Він описує всіх працівників автопідприємства загалом, тобто водіїв та обслуговуючий персонал. Цей клас матиме кілька похідних класів і є абстрактним, тому всі його властивості є захищеними (protected), а методи – публічними (public). Серед властивостей класу Worker є такі fullname (прізвище та ім'я) типу string, age (вік) типу unsigned int, experience (досвід роботи) типу float, а також методи доступу до цих значень: string get\_fullname() і void set\_fullname(string name), unsigned int get\_age() і void set\_age(unsigned int years), float get\_experience() і void set\_experience(float expr). Всі елементи і методи цього класу будуть взяті за основу під час створення похідних класів. Завдяки цьому можна зекономити багато часу на написання та налагодження коду програми. Об'єкти похідного класу будуть використовувати все, що створено і налагоджено в базовому класі, але й міститимуть власні унікальні елементи та методи. Як зазначалося вище, всі властивості класу інкапсульовані за допомогою специфікатора доступу protected. Він відрізняється від private тим, що дозволяє доступ до елементів базового класу з похідних класів. Якби елементи класу Worker перебували у полі private, то доступ до них був би закритий і було б неможливо змінити їхнє значення через об'єкти похідних класів. Тому під час створення класу, який надалі буде використано як базовий, то його поля треба оголошувати як protected, а не private. Інакше об'єкти похідного класу не зможуть звертатися до елементів базового.

Похідним від класу Worker є клас **Driver**, що описує водія. Окрім властивостей та методів, успадкованих від класу Worker, цей клас містить такі приватні елементи: licensenumber типу string для зберігання номеру та серії водійського посвідчення, category типу string для зберігання категорії, а також публічні методи доступу string

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



get\_licencenumber() і void set\_licencenumber(string licence), string get\_category() і void set\_category(string cat). За кожним водієм закріплено автомобіль, тому клас Driver повинен взаємодіяти з класом Vehicle.

В об'єктно-орієнтованому програмуванні класи можуть взаємодіяти між собою за допомогою наслідування, композиції та агрегації. Клас Driver не може наслідувати клас Vehicle, інакше це б суперечило логіці, тому для забезпечення взаємодії варто обирати між агрегацією і композицією. Однак, на ранніх етапах об'єктно-орієнтованого аналізу та проектування часто задаються відносини асоціацій, а свою конкретизацію у вигляді агрегацій і композицій вони отримують пізніше.

Композиція – це коли один клас не існує окремо від іншого. Залежний клас створюється під час створення незалежного, а незалежний клас повністю контролює залежний. У випадку композиції життєвий цикл об'єкта-"частини" залежить від об'єкта-"цілого", тобто якщо знищити незалежний клас, то залежний теж припинить своє існування. Агрегація – це коли екземпляр створюється в іншому місці коду, але передається в другий клас як параметр. У цьому разі життєвий цикл об'єкта-"частини" не залежить від об'єкта-"цілого", тобто якщо знищити незалежний клас, то залежний продовжить своє існування.

Хоча ведуться дискусії про переваги того чи іншого способу організації взаємодії між класами, будь-якого абстрактного правила не існує. Розробник обирає той чи інший шлях, спираючись на бізнес-логіку, а також бере до уваги можливості і обмеження, які дають і накладають ці способи. Отже, різниця між агрегацією і композицією

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

полягає в тому, що у зв'язку композиція життя об'єкта контролюється його "володарем", а під час агрегації – не контролюється.

У цій курсовій водії будуть агрегованими об'єктами для `Vehicle`, тому їх не буде створено там, а буде передано вже створений об'єкт, який "житиме" і після видалення з пам'яті поточного об'єкту. На думку автора, вибір агрегації тут логічніший, ніж вибір композиції, оскільки у разі списання поточного автомобіля водій працюватиме в автопідприємстві й далі. Агрегацію здійснено за допомогою використання вказівників, хоча варто зазначити, що це не є єдиним способом реалізації агрегації. Тому кожному класу, що походить від `Vehicle` надають вказівник на клас `Driver`. Таким чином можна встановити відповідність між автомобілями і водіями, а також отримувати чи встановлювати значення змінних об'єкта `Driver` через об'єкти класів, що описують автомобілі.

Ще одним класом, який успадковує елементи класу `Worker` є **`Staff`**, який описує обслуговуючий персонал, а саме слюсарів, техніків, зварників та складальників. Кожній цій професії притаманні розряди. Так, кожен спеціаліст може мати від 1 до 6 розряду. Тому в класі `Staff` створено змінну `category` типу `int`, що описує розряд. Сама змінна `category` захищена, а методи доступу до неї `int get_category()` і `void set_category(int cat)` – публічні. Окрім розряду, зварникам ще притаманні 4 рівні кваліфікації. Тому від класу `Staff` походить клас `Welder`, який має одну приватну змінну `level`. Тип цієї змінної – `int`. Методи доступу `int get_level()` і `void set_level(int lev)` – публічні.

Водії та обслуговуючий персонал об'єднуються в бригади, якими керують бригадири. Бригадами керують майстри, а майстрами – начальники ділянок і цехів. Для того, щоб реалізувати це програмно, на

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

думку автора, варто використати композицію, адже підлеглі повністю підпорядковуються своїм начальникам, а якщо знищити клас об'єкт начальника, то варто знищити й об'єкти підлеглих, бо без начальника працівники вже не будуть підлеглими. Потрібно створити класи **Brigadier**, **Master**, та **Head**, які описують бригадира, майстра та начальника і є похідними від **Worker**.

Для об'єктів гаражного господарства створено клас **Garage** з єдиним полем `area` типу `float`, що зберігає значення площі гаража у метрах квадратних. Змінна приватна, тому для роботи з нею необхідні методи доступу `float get_area()` і `void set_area(float m2)`.

Отже, існує загальний клас **Vehicle**, атрибути якого притаманні всім автомобілям автопідприємства. Від нього походять класи, що описують специфічні особливості кожного виду автомобільного транспорту. Також створено базовий клас **Worker**, який наслідують класи, що описують професії більш детально. Класи **Vehicle** та **Driver** взаємодіють за допомогою асоціації (агрегації). Взаємодію між класами візуалізовано за допомогою діаграми класів (рис. 7.2.1).

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

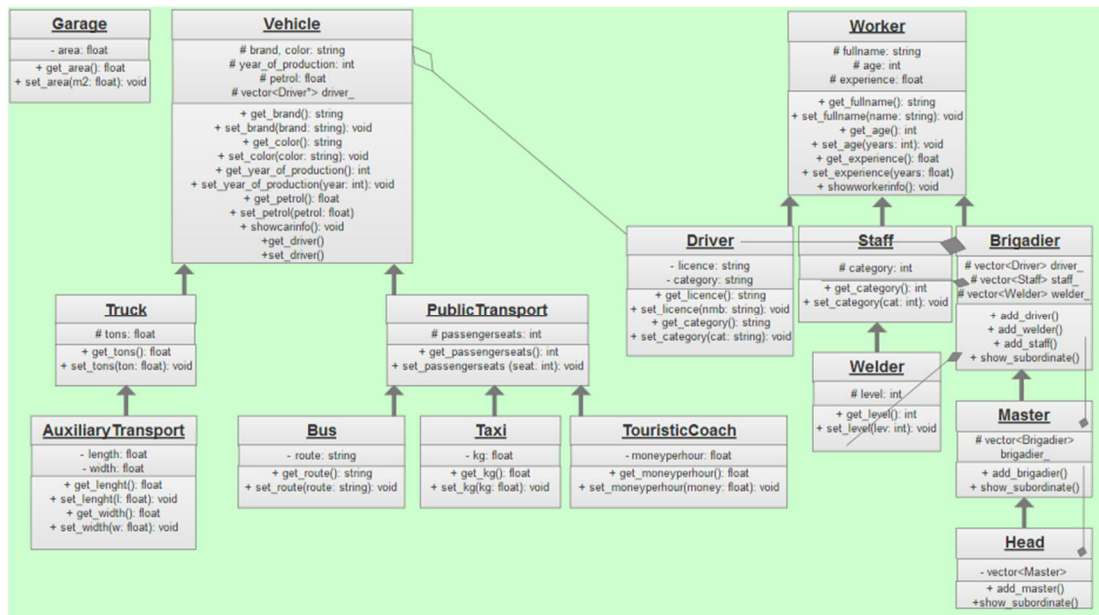


Рис. 7.2.1 – Діаграма класів

## ВИСНОВКИ

Для того, щоб виконати цю курсову роботу спочатку необхідно було проаналізувати предметну область – автопідприємство міста. Для аналізу предметної області було обрано типове автопідприємство, яке потребує автоматизації бізнес-процесів. Після аналізу предметної області було виділено і описано об'єкти в предметній області. Для кращого розуміння та унаочнення зв'язків між різними класами було створено діаграму класів. Після виділення об'єктів було створено базу даних у різних текстових файлах. Потім було розроблено програмний код в інтерпретаторі Dev C++, а саме створено класи, зчитано дані з файлів, налаштовано головне меню, розроблено запити, протестовано роботу програми.

Програма є захищеною та надійною, оскільки під час її розробки багато уваги було приділено інкапсуляції даних. Інкапсуляція являє собою захист членів чи методів об'єкта від доступу з боку іншого коду, що не відноситься до даного класу. Інкапсуляцію реалізовано за допомогою специфікаторів доступу. Так, усі змінні фінальних класів є приватними (використано специфікатор доступу `private`), а змінні класів, що мають нащадків, - захищеними (використано специфікатор доступу `protected`). Для роботи з цими змінними було розроблено методи доступу (використано специфікатор доступу `public`). Тобто безпосередньо звернутися до атрибуту будь-якого класу неможливо. Це можна зробити тільки за допомогою методів доступу. Таким чином усі поля захищено від будь-яких випадкових змін чи неправильного використання.

Оскільки для аналізу предметної області було обрано узагальнення автопідприємства, тобто таке автопідприємство, якому притаманні всі основні ознаки і атрибути автопідприємств, то програма також підходить для будь-якого автопідприємства, зокрема малого та середнього, адже у програмі реалізовано лише базовий функціонал. Для того, щоб програма підходила і для великих автокомпаній треба створити більше класів, оскільки у великих компаніях працюють ще й інші спеціалісти, окрім тих, для яких уже створено класи. Після цього доведеться розширити функціонал, наприклад, додати більше видів запитів в інформаційній системі, щоб якимось взаємодіяти з об'єктами нових класів. Також доцільно було б розробити графічний інтерфейс програми, адже саме такий інтерфейс зараз найбільш звичний для користувачів програмного забезпечення.

Отже, під час виконання курсової роботи ми навчилися аналізувати предметну область, моделювати взаємодію між об'єктами класів мовою UML, використовувати динамічні структури для раціонального використання пам'яті,

Арк.  
41

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

створювати та використовувати власні класи, розробляти прості та зрозумілі інтерфейси, краще зрозуміти наслідування, інкапсуляцію, абстрактні класи та інші підходи об'єктно-орієнтованого програмування.

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## ЛІТЕРАТУРА

- 1) <https://www.wikipedia.org/>
- 2) <http://cplusplus.com/doc/tutorial/>
- 3) <https://unetway.com/>
- 4) <https://ravesli.com/>
- 5) <https://metanit.com/>
- 6) <http://cppstudio.com/uk/>
- 7) <https://habr.com/ru/>
- 8) <https://pidru4niki.com/>
- 9) <http://eprints.cdu.edu.ua/>

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Міністерство освіти і науки України  
Івано-Франківський національний технічний університет нафти та газу  
Інститут інформаційних технологій

ЗАТВЕРДЖУЮ

зав. кафедри ІІЗ, проф., д.т.н.

\_\_\_\_\_ В.І. Шекета

«\_\_\_\_\_» \_\_\_\_\_ 2021р.

ТЕХНІЧНЕ ЗАВДАННЯ

На розробку прикладного програмного забезпечення з моделювання та аналізу програмного забезпечення»

1. Область застосування — проєктування інформаційних систем .
2. Основа розробки — робочий навчальний план дисципліни.
3. Мета та експлуатаційне призначення:
  - а. мета — отримання практичних навичок проєктування та конфігурування інформаційних систем ;
  - б. призначення розробки — навчальний курсовий проєкт із дисципліни «Проектування інформаційних систем»;
4. Джерела розробки — індивідуальне завдання на курсовий проєкт із дисципліни, технічні рекомендації щодо проєктування інформаційних систем та інші технічні матеріали для налаштування окремих компонентів програмної системи.
5. Технічні вимоги

Кінцевий термін виконання курсового проєкту «\_\_\_\_\_» 2020 р

Початок розробки «\_\_\_\_\_» 2020 р

Порядок контролю та прийняття.

6.1. Виконання етапів технічної та розрахункової документації курсового проєкту, а також моделювання роботи інформаційної системи контролюється викладачем згідно з графіком виконання проєкту;

6.2. Прийняття проєкту здійснюється комісією, затвердженою зав. кафедри згідно графіку захисту.

Арк.  
44

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ІІЗ	



6.3. Коригування технічного завдання допускається з дозволу керівника проекту.  
Розробив студент групи ІП-20-2 Кашуба Наталія

Програмний код.

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <vector>
```

```
#include <string>
```

```
using namespace std;
```

```
class Worker {
```

```
    protected:
```

```
        string fullname;
```

```
        unsigned int age;
```

```
        float experience;
```

```
    public:
```

```
        void set_fullname(string name){
```

```
            fullname=name;
```

```
        }
```

```
        string get_fullname() const {
```

```
            return fullname;
```

```
        }
```

Арк.  
45

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

void set_age(unsigned int years){

    age=years;

}

unsigned int get_age() const {

    return age;

}

void set_experience(float expr){

    experience=expr;

}

float get_experience() const {

    return experience;

}

virtual void showworkerinfo() const =0;

};

class Driver: public Worker{

    private:

        string licencenumber, category;

    public:

        void set_licencenumber(string licence){

            licencenumber=licence;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

    string get_licencenumber() const {

        return licencenumber;

    }

    void set_category(string cat){

        category=cat;

    }

    string get_category() const {

        return category;

    }

    void showworkerinfo() const override{

        cout<<"| "<<"\t"<<fullname<<"\t"<<"| "<<"\t"<<age<<"\t"<<"| "<<"\t"<<experience<
<"\t"<<"| "<<"\t"<<licencenumber<<"\t"<<"| "<<"\t"<<category<<"\t"<<"| "<<endl;

    }

};

class Staff: public Worker{

    protected:

        int category;

    public:

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

void set_category(int cat){

    category=cat;

}

int get_category() const {

    return category;

}

void showworkerinfo() const override{

    cout<<"|"<<"\t"<<fullname<<"\t"<<"|"<<"\t"<<age<<"\t"<<"|"<<"\t"<<experience<
<"\t"<<"|"<<"\t"<<category<<"\t"<<"|"<<endl;

}

};

class Welder: public Staff{

    private:

        int level;

    public:

        void set_level(int lev){

            level=lev;

        }

        int get_level() const {

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        return level;

    }

    void showworkerinfo() const override{

        cout<<"|"<<"\t"<<fullname<<"\t"<<"|"<<"\t"<<age<<"\t"<<"|"<<"\t"<<experience<
<"\t"<<"|"<<"\t"<<category<<"\t"<<"|"<<"\t"<<level<<"\t"<<"|"<<endl;

    }

};

class Brigadier: public Worker{

    protected:

    vector<Driver> driver_;

    vector<Staff> staff_;

    vector<Welder> welder_;

    public:

        void showworkerinfo() const override{

            cout<<"|"<<"\t"<<fullname<<"\t"<<"|"<<"\t"<<age<<"\t"<<"|"<<"\t"<<experience<
<"\t"<<"|"<<endl;

        }

        void add_driver(Driver driver) { //додавання водія-підлеглого

            driver_.push_back(driver);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

    void add_staff(Staff staff) { //додавання техніка-підлеглого (обслуговуючий
персонал)

    staff_.push_back(staff);

    }

    void add_welder(Welder welder) { //додавання зварника-підлеглого

welder_.push_back(welder);

    }

    void show_subordinate() const { //вивід даних про підлеглих

        cout<<" Прізвище та ім'я бригадира "<<get_fullname()<<endl;

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

            cout<<driver_[i].get_fullname()<<endl;

            }

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

                cout<<welder_[i].get_fullname()<<endl;

                }

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

                cout<<staff_[i].get_fullname()<<endl;

            }

        }

    }

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

};

class Master: public Brigadier{

protected:

    vector<Brigadier> brigadier_;

public:

    void add_brigadier(Brigadier brigadier) {//додавання бригадира-
підлеглого

    brigadier_.push_back(brigadier);

    }

    void show_subordinate() const{//вивід даних про підлеглих

    cout<<"    Прізвище та ім'я майстра "<<get_fullname()<<endl;

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

        brigadier_[i].show_subordinate();

    }

}

};

class Head: public Master{

private:

    vector<Master> master_;

public:

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

void add_master(Master master) {//додавання майстра-підлеглого

    master_.push_back(master);

}

void show_subordinate()const{//вивід даних про підлеглих

    cout<<"          Прізвище та ім'я голови цеху
"<<get_fullname()<<endl;

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

        master_[i].show_subordinate();

    }

}

};

class Garage{

    private:

        float area;

    public:

        void set_area(float m2){

            area=m2;

        }

        float get_area() const {

            return area;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



```

    }

};

class Runtime{//для збереження даних про пробіг

private:

    int Day;

    int Month;

    int Year;

    float Run;

public:

    void get_runtime(int& d, int& m, int& y, float& km){

        d=Day;

        m=Month;

        y=Year;

        km=Run;

    }

    void set_runtime(int d, int m, int y, float km){

        Day=d;

        Month=m;

        Year=y;

        Run=km;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

    int get_day(){return Day;}

    int get_month(){return Month;}

    int get_year(){return Year;}

    float get_run(){return Run;}

};

class Vehicle {

protected:

    string brand;

    string color;

    string number;

    unsigned int year_of_production;

    float petrol;

    vector<Driver*> driver_;

    vector<Runtime> runtimes;

public:

    void set_brand(string brandname){

        brand=brandname;

    }

    string get_brand() const {

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        return brand;

    }

    void set_color(string colour){

        color=colour;

    }

    string get_color() const {

        return color;

    }

    void set_number(string nmb){

        number=nmb;

    }

    string get_number()const{

        return number;

    }

    void set_year_of_production(unsigned int year){

        year_of_production=year;

    }

    unsigned int get_year_of_production()const{

        return year_of_production;

    }

```

Арк.  
55

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

void set_petrol(float ptrl){

    petrol=ptrl;

}

float get_petrol()const{

    return petrol;

}

void add_driver(Driver* driver) {//закріплення водія за автомобілем

    driver_.push_back(driver);

}

void show_driver(){//вивід даних про водіїв

    cout<<"_____
_____ "<<endl;

    cout<<"\tПізвище та ім'я\t\tНомер посвідчення\t"<<endl;

    cout<<"_____
_____ "<<endl;

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

        cout<<"\t"<<driver_[i]->get_fullname()<<"\t\t"<<driver_[i]-
>get_licencenumber()<<"\t\t\n";

        cout<<"_____
_____ "<<endl;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

    cout<<"За автомашиною закріплено "<<driver_.size()<<" водіїв."<<endl;

    }

    void add_runtime(Runtime runtime) {

    runtimes.push_back(runtime);

    }

    void show_runtime(){

        cout<<number<<endl;

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

            cout<<runtimes[i].get_day()<<". "<<runtimes[i].get_month()<<". "<<runtimes[i]
            ].get_year()<<" - "<<runtimes[i].get_run()<<endl;

            }

        }

    virtual void showcarinfo()const=0;

};

class PublicTransport: public Vehicle{//абстрактний клас, бо віртуальний метод
не перевантажено

protected:

    unsigned int passengerseats;

public:

```

Арк.  
57

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

void set_passengersseats(unsigned int seats){

    passengerseats=seats;

}

unsigned int get_passengerseats()const{

    return passengerseats;

}

};

class Bus: public PublicTransport{

    private:

        string route;

    public:

        void set_route(string rout){

            route=rout;

        }

        string get_route()const{

            return route;

        }

        void showcarinfo()const override{

            cout<<"|"<<"\t"<<number<<"\t"<<"|"<<"\t"<<color<<"\t"<<"|"<<"\t"<<brand<<"\t"<

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



```

class TouristicCoach: public PublicTransport{

    private:

        float moneyperhour;

    public:

        void set_moneyperhour(float money){

            moneyperhour=money;

        }

        float get_moneyperhour()const{

            return moneyperhour;

        }

        void showcarinfo()const override{

            cout<<"|"<<"\t"<<number<<"\t"<<"|"<<"\t"<<color<<"\t"<<"|"<<"\t"<<brand<<"\t"<
            <"|"<<"\t"<<year_of_production<<"\t"<<"|"<<"\t"<<petrol<<"\t"<<"|"<<"\t"<<
            passengerseats<<"\t"<<"|"<<"\t"<<moneyperhour<<"\t"<<"|"<<endl;

        }

};

class Truck: public Vehicle{

    protected:

        float tons;

    public:

```

Арк.  
60

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



```

void set_tons(float ton){

    tons=ton;

}

float get_tons()const{

    return tons;

}

void showcarinfo()const override{

    cout<<"|"<<"\t"<<number<<"\t"<<"|"<<"\t"<<color<<"\t"<<"|"<<"\t"<<brand<<"\t"<
<"|"<<"\t"<<year_of_production<<"\t"<<"|"<<"\t"<<petrol<<"\t"<<"|"<<"\t"<<tons<<"\t"<
<"|"<<endl;

}

};

class AuxiliaryTransport: public Truck{

    private:

        float length, width;

    public:

        void set_length(float m){

            length=m;

        }

        float get_length()const{

```

Арк.  
61

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        return length;

    }

    void set_width(float m){

        width=m;

    }

    float get_width()const{

        return width;

    }

    void showcarinfo()const override{

        cout<<"|"<<"\t"<<number<<"\t"<<"|"<<"\t"<<color<<"\t"<<"|"<<"\t"<<brand<<"\t"<
        <"|"<<"\t"<<year_of_production<<"\t"<<"|"<<"\t"<<petrol<<"\t"<<"|"<<"\t"<<tons<<"\t"<
        <"|"<<"\t"<<length<<"\t"<<"|"<<"\t"<<width<<"\t"<<"|"<<endl;

    }

};

//методи для зчитування даних з текстових файлів

void cin_string(string& str,ifstream& file,int size=256,char delim='|'){

    str="";

    getline(file,str,delim);

}

void cin_float(double& f,ifstream& file,char delim='|',int size=256){

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

char* cstr= new char[size];

file.getline(cstr,size,delim);

f=atof(cstr);

delete[] cstr;

}

void cin_int(int& f,ifstream& file,char delim='|',int size=256){

char* cstr= new char[size];

file.getline(cstr,size,delim);

f=atoi(cstr);

delete[] cstr;

}

int main()

{

SetConsoleCP(1251);//для введення українських літер

SetConsoleOutputCP(1251);//для виведення українських літер

SetConsoleTitle("Курсова робота студентки групи ІП-20-2 Кашуби Наталії");

cout << "

_____

_____ "<<endl;

cout << "| База даних автопідприємства

|"<<endl;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
cout << "
```

```
_____"<<endl;
```

```
cout << "| Види запитів: _____|"<<endl;
```

```
cout << "| [1]Перегляд даних про автопарк  
|"<<endl;
```

```
cout << "| [2]Перегляд розподілу автобусів по маршрутах  
|"<<endl;
```

```
cout << "| [3]Перегляд списку і кількості водіїв  
|"<<endl;
```

```
cout << "| [4]Перегляд даних про об'єкти гаражного господарства  
|"<<endl;
```

```
cout << "| [5]Перегляд списку і кількості водіїв зазначеної автомашини  
|"<<endl;
```

```
cout << "| [6]Перегляд розподілу водіїв по автомобілях  
|"<<endl;
```

```
cout << "| [7]Перегляд підпорядкованості персоналу  
|"<<endl;
```

```
cout << "| [8]Перегляд складу підлеглого зазначеного керівника  
|"<<endl;
```

```
cout << "| [9]Перегляд пробігу автотранспорту певної категорії за  
вказану дату|"<<endl;
```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
cout << "
```

```
_____"<<endl;
```

```
vector<Bus> buses;
```

```
vector<Taxi> taxis;
```

```
vector<TouristicCoach> coaches;
```

```
vector<Truck> trucks;
```

```
vector<AuxiliaryTransport> auxiliaries;
```

```
vector<Garage> garages;
```

```
vector<Driver> drivers;
```

```
vector<Staff> staff;
```

```
vector<Welder> welders;
```

```
vector<Brigadier> brigadiers;
```

```
vector<Master> masters;
```

```
vector<Head> heads;
```

```
vector<Runtime> runt;
```

```
vector<Runtime> runb;
```

```
vector<Runtime> runa;
```

```
string carnumber, carcolor, carbrand, carroute, personname, personlicence,  
personcategory;//для тимчасового зберігання зчитаних даних
```

Арк.  
65

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
int caryear, carseat, personage, npersoncategory, personlevel, runday, runmonth,
runyear;
```

```
double carpetrol, carkg, carmoney, carton, carlength, carwidth, cararea, personexp,
runkm;
```

```
//послідовне зчитування даних з файлів і додавання об'єктів до векторів
```

```
ifstream fbus("buses.txt");
```

```
if(!fbus.is_open()){
```

```
cout<<"\n"<<"Файл, що містить дані про маршрутні автобуси, не
знайдено."<<endl;
```

```
}
```

```
while(!fbus.eof()){
```

```
    cin_string(carnumber,fbus);
```

```
    cin_string(carcolor,fbus);
```

```
    cin_string(carbrand,fbus);
```

```
    cin_int(caryear,fbus);
```

```
    cin_float(carpetrol,fbus);
```

```
    cin_int(carseat,fbus);
```

```
    cin_string(carroute,fbus);
```

```
    Bus bus;
```

```
    bus.set_number(carnumber);
```

```
    bus.set_color(carcolor);
```

Арк.  
66

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        bus.set_brand(carbrand);

        bus.set_year_of_production(caryear);

        bus.set_petrol(carpetrol);

        bus.set_passengersseats(carseat);

        bus.set_route(carroute);

        buses.push_back(bus);

    }

    fbus.close();

    ifstream ftaxi("taxi.txt");

    if(!ftaxi.is_open()){

        cout<<"\n"<<"Файл, що містить дані про таксі, не знайдено."<<endl;

    }

    while(!ftaxi.eof()){

        cin_string(carnumber,ftaxi);

        cin_string(carcolor,ftaxi);

        cin_string(carbrand,ftaxi);

        cin_int(caryear,ftaxi);

        cin_float(carpetrol,ftaxi);

        cin_int(carseat,ftaxi);

        cin_float(carkg,ftaxi);

```

Арк.  
67

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

Taxi taxi;

taxi.set_number(carnumber);

    taxi.set_color(carcolor);

    taxi.set_brand(carbrand);

    taxi.set_year_of_production(caryear);

    taxi.set_petrol(carpetrol);

    taxi.set_passengersseats(carseat);

    taxi.set_kg(carkg);

    taxis.push_back(taxi);

}

ftaxi.close();

ifstream fcoach("coaches.txt");

if(!fcoach.is_open()){

    cout<<"\n"<<"Файл, що містить дані про туристичні автобуси, не
знайдено."<<endl;

}

while(!fcoach.eof()){

    cin_string(carnumber,fcoach);

    cin_string(carcolor,fcoach);

    cin_string(carbrand,fcoach);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



```

cin_int(caryear,fcoach);

cin_float(carpetrol,fcoach);

cin_int(carseat,fcoach);

cin_float(carmoney,fcoach);

TouristicCoach coach;

coach.set_number(carnumber);

    coach.set_color(carcolor);

    coach.set_brand(carbrand);

    coach.set_year_of_production(caryear);

    coach.set_petrol(carpetrol);

    coach.set_passengersseats(carseat);

    coach.set_moneyperhour(carmoney);

    coaches.push_back(coach);

}

fcoach.close();

ifstream ftruck("trucks.txt");

if(!ftruck.is_open()){

cout<<"\n"<<"Файл, що містить дані про вантажівки, не знайдено."<<endl;

}

while(!ftruck.eof()){

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        cin_string(carnumber,ftruck);

        cin_string(carcolor,ftruck);

        cin_string(carbrand,ftruck);

        cin_int(caryear,ftruck);

        cin_float(carpetrol,ftruck);

        cin_float(carton,ftruck);

        Truck truck;

        truck.set_number(carnumber);

        truck.set_color(carcolor);

        truck.set_brand(carbrand);

        truck.set_year_of_production(caryear);

        truck.set_petrol(carpetrol);

        truck.set_tons(carton);

        trucks.push_back(truck);

    }

    ftruck.close();

    ifstream fauxiliary("auxiliary.txt");

    if(!fauxiliary.is_open()){

        cout<<"\n"<<"Файл, що містить дані про допоміжний транспорт, не  

        знайдено."<<endl;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

}

while(!fauxiliary.eof()){

    cin_string(carnumber,fauxiliary);

    cin_string(carcolor,fauxiliary);

    cin_string(carbrand,fauxiliary);

    cin_int(caryear,fauxiliary);

    cin_float(carpetrol,fauxiliary);

    cin_float(carton,fauxiliary);

    cin_float(carlength,fauxiliary);

    cin_float(carwidth,fauxiliary);

    AuxiliaryTransport auxiliary;

    auxiliary.set_number(carnumber);

    auxiliary.set_color(carcolor);

    auxiliary.set_brand(carbrand);

    auxiliary.set_year_of_production(caryear);

    auxiliary.set_petrol(carpetrol);

    auxiliary.set_tons(carton);

    auxiliary.set_length(carlength);

    auxiliary.set_width(carwidth);

    auxiliaries.push_back(auxiliary);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

}

fauxiliary.close();

ifstream fgarage("garages.txt");

if(!fgarage.is_open()){

    cout<<"\n"<<"Файл, що містить дані про об'єкти гаражного господарства, не
знайдено."<<endl;

}

while(!fgarage.eof()){

    cin_float(cararea,fgarage);

    Garage garage;

    garage.set_area(cararea);

    garages.push_back(garage);

}

fgarage.close();

ifstream fdriver("drivers.txt");

if(!fdriver.is_open()){

    cout<<"\n"<<"Файл, що містить дані про водіїв, не знайдено."<<endl;

}

while(!fdriver.eof()){

    cin_string(personname,fdriver);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        cin_int(personage,fdriver);

    cin_float(personexp,fdriver);

    cin_string(personlicence,fdriver);

    cin_string(personcategory,fdriver);

    Driver driver;

    driver.set_fullname(personname);

    driver.set_age(personage);

    driver.set_experience(personexp);

    driver.set_licencenumber(personlicence);

    driver.set_category(personcategory);

    drivers.push_back(driver);

}

fdriver.close();

ifstream fstaff("staff.txt");

if(!fstaff.is_open()){

    cout<<"\n"<<"Файл, що містить дані про обслуговуючий персонал, не
знайдено."<<endl;

}

while(!fstaff.eof()){

    cin_string(personname,fstaff);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        cin_int(personage,fstaff);

    cin_float(personexp,fstaff);

    cin_int(npersoncategory,fstaff);

    Staff staf;

    staf.set_fullname(personname);

    staf.set_age(personage);

    staf.set_experience(personexp);

    staf.set_category(npersoncategory);

    staff.push_back(staf);

}

fstaff.close();

ifstream fwelder("welders.txt");

if(!fwelder.is_open()){

    cout<<"\n"<<"Файл, що містить дані про слюсарів, не знайдено."<<endl;

}

while(!fwelder.eof()){

    cin_string(personname,fwelder);

    cin_int(personage,fwelder);

    cin_float(personexp,fwelder);

    cin_int(npersoncategory,fwelder);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

cin_int(personlevel,fwelder);

Welder welder;

welder.set_fullname(personname);

welder.set_age(personage);

welder.set_experience(personexp);

welder.set_category(npersoncategory);

welder.set_level(personlevel);

welders.push_back(welder);

}

fwelder.close();

ifstream fbrigadier("brigadiers.txt");

if(!fbrigadier.is_open()){

cout<<"\n"<<"Файл, що містить дані про бригадирів, не знайдено."<<endl;

}

while(!fbrigadier.eof()){

    cin_string(personname,fbrigadier);

    cin_int(personage,fbrigadier);

    cin_float(personexp,fbrigadier);

    Brigadier brigadier;

    brigadier.set_fullname(personname);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

brigadier.set_age(personage);

brigadier.set_experience(personexp);

brigadiers.push_back(brigadier);

}

fbrigadier.close();

ifstream fmaster("masters.txt");

if(!fmaster.is_open()){

cout<<"\n"<<"Файл, що містить дані про майстрів, не знайдено."<<endl;

}

while(!fmaster.eof()){

    cin_string(personname,fmaster);

    cin_int(personage,fmaster);

    cin_float(personexp,fmaster);

    Master master;

    master.set_fullname(personname);

    master.set_age(personage);

    master.set_experience(personexp);

    masters.push_back(master);

}

fmaster.close();

```

Арк.  
76

	Зм.	Арк.№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



```

ifstream fhead("heads.txt");

if(!fhead.is_open()){

    cout<<"\n"<<"Файл, що містить дані про начальників ділянок та цехів, не
знайдено."<<endl;

}

while(!fhead.eof()){

    cin_string(personname,fhead);

    cin_int(personage,fhead);

    cin_float(personexp,fhead);

    Head head;

    head.set_fullname(personname);

    head.set_age(personage);

    head.set_experience(personexp);

    heads.push_back(head);

}

fhead.close();

//закріплення автомобілів за водіями

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

    for(int j=0;j<3;j++){

        buses[i].add_driver(&drivers[j]);

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

}

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

    for(int j=3;j<5;j++){

        coaches[i].add_driver(&drivers[j]);

    }

}

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

    for(int j=5;j<10;j++){

        taxis[i].add_driver(&drivers[j]);

    }

}

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

    for(int j=10;j<13;j++){

        trucks[i].add_driver(&drivers[j]);

    }

}

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

    for(int j=13;j<15;j++){

        auxiliaries[i].add_driver(&drivers[j]);

```

Арк.  
78

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

}

//додавання підлеглих до водіїв

for(int i=0;i<4;i++){

    brigadiers[0].add_welder(welders[i]);

}

brigadiers[1].add_welder(welders[1]);

for(int i=0;i<2;i++){

    brigadiers[1].add_staff(staff[i]);

}

for(int i=2;i<4;i++){

    brigadiers[2].add_staff(staff[i]);

}

for(int i=4;i<6;i++){

    brigadiers[3].add_staff(staff[i]);

}

for(int i=6;i<staff.size();i++){

    brigadiers[4].add_staff(staff[i]);

}

for(int i=0;i<5;i++){

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        brigadiers[5].add_driver(drivers[i]);

    }

    for(int i=5;i<10;i++){

        brigadiers[6].add_driver(drivers[i]);

    }

    for(int i=10;i<drivers.size();i++){

        brigadiers[7].add_driver(drivers[i]);

    }

    for(int i=0;i<2;i++){

        masters[0].add_brigadier(brigadiers[i]);

    }

    for(int i=2;i<4;i++){

        masters[1].add_brigadier(brigadiers[i]);

    }

    for(int i=4;i<6;i++){

        masters[2].add_brigadier(brigadiers[i]);

    }

    for(int i=6;i<brigadiers.size();i++){

        masters[3].add_brigadier(brigadiers[i]);

    }

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

for(int i=0;i<2;i++){

    heads[0].add_master(masters[i]);

}

for(int i=2;i<masters.size();i++){

    heads[1].add_master(masters[i]);

}

ifstream frunbus("runbus.txt");

if(!frunbus.is_open()){

    cout<<"\n"<<"Файл, що містить дані про пробіг пасажирського транспорту, не
знайдено."<<endl;

}

while(!frunbus.eof()){

    cin_int(runday,frunbus);

    cin_int(runmonth,frunbus);

    cin_int(runyear,frunbus);

    cin_float(runkm,frunbus);

    Runtime rb;

    rb.set_runtime(runday,runmonth,runyear,runkm);

    runb.push_back(rb);

}

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

frunbus.close();

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

    for(int j=0;j<runb.size();j++){

        buses[i].add_runtime(runb[j]);

    }

}

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

    for(int j=0;j<runb.size();j++){

        taxis[i].add_runtime(runb[j]);

    }

}

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

    for(int j=0;j<runb.size();j++){

        coaches[i].add_runtime(runb[j]);

    }

}

ifstream fruntruck("runtruck.txt");

if(!fruntruck.is_open()){

    cout<<"\n"<<"Файл, що містить дані про пробіг вантажного транспорту, не
знайдено."<<endl;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

}

while(!fruntruck.eof()){

    cin_int(runday,fruntruck);

    cin_int(runmonth,fruntruck);

    cin_int(runyear,fruntruck);

    cin_float(runkm,fruntruck);

    Runtime rt;

    rt.set_runtime(runday,runmonth,runyear,runkm);

    runt.push_back(rt);

}

fruntruck.close();

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

    for(int j=0;j<runt.size();j++){

        trucks[i].add_runtime(runt[j]);

    }

}

ifstream frunaux("runaux.txt");

if(!frunaux.is_open ()){

    cout<<"\n"<<"Файл, що містить дані про пробіг допоміжного транспорту, не знайдено."<<endl;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

}

while(!frunaux.eof()){

    cin_int(runday,frunaux);

    cin_int(runmonth,frunaux);

    cin_int(runyear,frunaux);

    cin_float(runkm,frunaux);

    Runtime ra;

    ra.set_runtime(runday,runmonth,runyear,runkm);

    runa.push_back(ra);

}

frunaux.close();

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

    for(int j=0;j<runa.size();j++){

        auxiliaries[i].add_runtime(runa[j]);

    }

}

cout<<"Оберіть дію ";

int action;

cin>>action;

switch(action){

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



case 1:

```
cout<<"Маршрутні автобуси"<<endl;
```

```
cout << "
```

```
_____  
_____  
_____ "<<endl;
```

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Колір"<<"\t"<<"|"<<"\t"<<"Марка"<<"\t"  
<<"|"<<"\t"<<"Рік виробництва "<<"\t"<<"|"<<"\t"<<"л бензину/100  
м"<<"\t"<<"|"<<"\t"<<"Кількість  
місць"<<"\t"<<"|"<<"\t"<<"Маршрут"<<"\t"<<"|"<<"\n";
```

```
cout << "
```

```
_____  
_____  
_____ "<<endl;
```

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

```
    buses[i].showcarinfo();
```

```
    cout << "
```

```
_____  
_____  
_____ "<<endl;
```

```
}
```

```
cout<<"Таксі"<<endl;
```

Арк.  
85

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
cout << "
```

```
_____"<<endl;
```

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Колір"<<"\t"<<"|"<<"\t"<<"Марка"<<"\t"
<<"|"<<"\t"<<"Рік виробництва"<<"\t"<<"|"<<"\t"<<"л бензину/100
м"<<"\t"<<"|"<<"\t"<<"Кількість місць"<<"\t"<<"|"<<"\t"<<"Вантажопідйомність
(кг)"<<"\t"<<"|"<<endl;
```

```
cout << "
```

```
_____"<<endl;
```

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

```
    taxis[i].showcarinfo();
```

```
    cout << "
```

```
_____"<<endl;
```

```
}
```

```
cout<<"Туристичні автобуси"<<endl;
```

```
cout << "
```

```
_____"<<endl;
```

Арк.  
86

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Колір"<<"\t"<<"|"<<"\t"<<"Марка"<<"\t"
<<"|"<<"\t"<<"Рік виробництва"<<"\t"<<"|"<<"\t"<<"л бензину/100
м"<<"\t"<<"|"<<"\t"<<"Кількість місць"<<"\t"<<"|"<<"\t"<<"Вартість оренди за 1 год
(грн)"<<"\t"<<"|"<<endl;
```

```
cout << "
```

```
_____  
_____  
_____"<<endl;
```

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

```
    coaches[i].showcarinfo();
```

```
    cout << "
```

```
_____  
_____  
_____"<<endl;
```

```
}
```

```
cout<<"Вантажівки"<<endl;
```

```
cout << "
```

```
_____  
_____ "<<endl;
```

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Колір"<<"\t"<<"|"<<"\t"<<"Марка"<<"\t"
<<"|"<<"\t"<<"Рік виробництва"<<"\t"<<"|"<<"\t"<<"л бензину/100
м"<<"\t"<<"|"<<"\t"<<"Вантажопідйомність (т)"<<"\t"<<"|"<<endl;
```

```
cout << "
```

```
"<<endl;
```

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

```
    trucks[i].showcarinfo();
```

```
    cout << "
```

```
"<<endl;
```

```
}
```

```
cout<<"Допоміжний транспорт"<<endl;
```

```
cout << "
```

```
"<<endl;
```

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Колір"<<"\t"<<"|"<<"\t"<<"Марка"<<"\t"
<<"|"<<"\t"<<"Рік виробництва"<<"\t"<<"|"<<"\t"<<"л бензину/100
м"<<"\t"<<"|"<<"\t"<<"Вантажопідйомність (т)"<<"\t"<<"|"<<"\t"<<"Довжина
"<<"|"<<"Ширина"<<"\t"<<"|"<<endl;
```

```
cout << "
```

```
"<<endl;
```

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

```

        auxiliaries[i].showcarinfo();

        cout << "
____
____
____" << endl;

    }

    break;

case 2:

    cout << " _____" << endl;

    cout << "|" << "\t" << "Homep" << "\t" << "|" << "\t" << "Маршрут" << "\t" << "|" << "\n";

    cout << " _____" << endl;

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

        cout << "|" << "\t" << buses[i].get_number() << "\t" << "|" << "\t" << buses[i].get_route() << "\t" <<
        "|" << endl;

        cout << "
_____ " << endl;

    }

    break;

case 3:

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
cout << "
```

```
"<<endl;
```

```
cout<<"|"<<"\t"<<"Прізвище та  
ім'я"<<"\t"<<"|"<<"\t"<<"Вік"<<"\t"<<"|"<<"\t"<<"Досвід  
роботи"<<"\t"<<"|"<<"\t"<<"Номер  
ліцензії"<<"\t"<<"|"<<"\t"<<"Категорія"<<"\t"<<"|"<<endl;
```

```
cout << "
```

```
"<<endl;
```

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

```
drivers[i].showworkerinfo();
```

```
cout << "
```

```
"<<endl;
```

```
}
```

```
cout<<"Загальне число водіїв автопідприємства:  
"<<drivers.size()<<". "<<endl;
```

```
break;
```

```
case 4:
```

```
cout<<"Наявність об'єктів гаражного господарства в цілому"<<endl;
```

```
cout << " _____ "<<endl;
```

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Площа"<<"\t"<<"|"<<endl;
```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

cout << " _____" << endl;

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

cout<<"|" << "\t" << i+1 << "\t" << "|" << "\t" << garages[i].get_area() << "\t" << "|" << endl;

        cout << " _____" << endl;

    }

    cout<<"Наявність об'єктів гаражного господарства для легкових
автомобілів" << endl;

    cout << " _____" << endl;

    cout<<"|" << "\t" << "Номер" << "\t" << "|" << "\t" << "Площа" << "\t" << "|" << endl;

    cout << " _____" << endl;

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

        if(garages[i].get_area()<35){

            cout<<"|" << "\t" << i+1 << "\t" << "|" << "\t" << garages[i].get_area() << "\t" << "|" << endl;

                cout << " _____" << endl;

            }

        }

        cout<<"Наявність об'єктів гаражного господарства для
автобусів" << endl;

        cout << " _____" << endl;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Площа"<<"\t"<<"|"<<endl;

cout << " _____" << endl;

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

    if(garages[i].get_area()>=35&&garages[i].get_area()<80){

cout<<"|"<<"\t"<<i+1<<"\t"<<"|"<<"\t"<<garages[i].get_area()<<"\t"<<"|"<<endl;

        cout << " _____" << endl;

    }

}

cout<<"Наявність об'єктів гаражного господарства для
вантажівок"<<endl;

cout << " _____" << endl;

cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Площа"<<"\t"<<"|"<<endl;

cout << " _____" << endl;

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

    if(garages[i].get_area()>=80&&garages[i].get_area()<115){

cout<<"|"<<"\t"<<i+1<<"\t"<<"|"<<"\t"<<garages[i].get_area()<<"\t"<<"|"<<endl;

        cout << " _____" << endl;

    }

}

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



```
cout<<"Наявність об'єктів гаражного господарства для допоміжного  
транспорту"<<endl;
```

```
cout << " _____" << endl;
```

```
cout<<"|"<<"\t"<<"Номер"<<"\t"<<"|"<<"\t"<<"Площа"<<"\t"<<"|"<<endl;
```

```
cout << " _____" << endl;
```

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

```
    if(garages[i].get_area()>=115){
```

```
        cout<<"|"<<"\t"<<i+1<<"\t"<<"|"<<"\t"<<garages[i].get_area()<<"\t"<<"|"<<endl;
```

```
        cout << " _____" << endl;
```

```
    }
```

```
}
```

```
break;
```

```
case 5:
```

```
    cout<<"Введіть номер автомашини ";
```

```
    cin>>carnumber;
```

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

```
        if(carnumber==buses[i].get_number()){
```

```
            buses[i].show_driver();
```

```
        }
```

Арк.  
93

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

    }

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

        if(carnumber==coaches[i].get_number()){

            coaches[i].show_driver();

        }

    }

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

        if(carnumber==taxis[i].get_number()){

            taxis[i].show_driver();

        }

    }

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

        if(carnumber==trucks[i].get_number()){

            trucks[i].show_driver();

        }

    }

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

        if(carnumber==auxiliaries[i].get_number()){

            auxiliaries[i].show_driver();

        }

    }

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        }

        break;

    case 6:

        cout<<"Маршрутні автобуси"<<endl;

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

            cout<<"Номер автомашини "<<buses[i].get_number()<<endl;

            buses[i].show_driver();

        }

        cout<<"Туристичні автобуси"<<endl;

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

            cout<<"Номер автомашини

"<<coaches[i].get_number()<<endl;

            coaches[i].show_driver();

        }

        cout<<"Таксі"<<endl;

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

            cout<<"Номер автомашини

"<<taxis[i].get_number()<<endl;

            taxis[i].show_driver();

        }

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

cout<<"Вантажівки"<<endl;

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

    cout<<"Номер автомашини "<<trucks[i].get_number()<<endl;

    trucks[i].show_driver();

}

cout<<"Допоміжний транспорт"<<endl;

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

    cout<<"Номер автомашини
"<<auxiliaries[i].get_number()<<endl;

    auxiliaries[i].show_driver();

}

break;

case 7:

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

heads[i].show_subordinate();

    }

    break;

case 8:

    cout<<"Введіть прізвище та ім'я керівника через нижнє
підкреслення _ ";

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        cin>>personname;

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

            if(brigadiers[i].get_fullname()==personname){

brigadiers[i].show_subordinate();

                }

            }

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

            if(masters[i].get_fullname()==personname){

                masters[i].show_subordinate();

                }

            }

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

            if(heads[i].get_fullname()==personname){

                heads[i].show_subordinate();

                }

            }

        break;

    case 9:

        cout<<"Введіть категрію через нижнє підкреслення _ ";

        cin>>carcolor;

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```
cout<<"Введіть день ";
```

```
cin>>runday;
```

```
cout<<"Введіть місяць ";
```

```
cin>>runmonth;
```

```
cout<<"Введіть рік ";
```

```
cin>>runyear;
```

```
if(carcolor=="Маршрутні_автобуси"||carcolor=="Туристичні_автобуси"||car  
color=="Таксі"){
```

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

```
for(int j=0;j<runb.size();j++){
```

```
if(runb[j].get_day()==runday&&runb[j].get_month()==runmonth&&runb[j].ge  
t_year()==runyear){
```

```
cout<<buses[i].get_number()<<"\t"<<runb[j].get_run()<<" км"<<endl;
```

```
}
```

```
}
```

```
}
```

```
}
```

```
if(carcolor=="Допоміжний_транспорт"){
```

```

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

            for(int j=0;j<runa.size();j++){

                if(runa[j].get_day()==runday&&runa[j].get_month()==runmonth&&runa[j].get_
_year()==runyear){

                    cout<<buses[i].get_number()<<"\t"<<runa[j].get_run()<<" км"<<endl;

                        }

                    }

                }

            }

            if(carcolor=="Вантажний_транспорт"){

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

                    for(int j=0;j<runt.size();j++){

                        if(runt[j].get_day()==runday&&runt[j].get_month()==runmonth&&runt[j].get_
year()==runyear){

                            cout<<trucks[i].get_number()<<"\t"<<runt[j].get_run()<<" км"<<endl;

                                }

                            }

                        }

                    }

```

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

```

        }

        break;

default:

        cout<<"Ви ввели неправильне значення.";

    }

}

```

Вміст текстових файлів, з яких було сформовано базу даних.

Вміст файлу **buses.txt**:

```

0098|червоний|Ikarus|1999|32.6|38|Івано-Франківськ-Львів|
4400|зелений|Hyundai|2020|16.3|56|Івано-Франківськ-Одеса|
8967|оранжевий|Daewoo|2017|12.3|24|Івано-Франківськ-Тернопіль|
7344|чорний|Mercedes-Benz|2001|36.5|84|Івано-Франківськ-Київ|
3434|білий|MAN|2003|23|40|Івано-Франківськ-Ужгород|
6834|рожевий|Volvo|2009|34.2|48|Івано-Франківськ-Харків|
4814|коричневий|Scania|2005|40.4|80|Івано-Франківськ-Чернігів|

```

Арк.  
100

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



7561|фіолетовий|VolgaBus|2013|40.1|76|Івано-Франківськ-Чернівці|

6522|синій|Ikarus|2007|11.5|20|Івано-Франківськ-Тернопіль|

9274|сірий|Mercedes-Benz|2017|46.4|54|Івано-Франківськ-Чернівці|

8887|чорний|Hyundai|2005|58.1|88|Івано-Франківськ-Луцьк|

5782|білий|Ikarus|2015|12.2|24|Івано-Франківськ-Дніпро|

3294|зелений|Daewoo|1995|66.3|54|Івано-Франківськ-Київ|

4323|червоний|Scania|1994|32.8|42|Івано-Франківськ-Київ|

4314|фіолетовий|Ikarus|2000|16.2|30|Івано-Франківськ-Чернівці|

Вміст файлу **taxi.txt**:

1198|червоний|Toyota|1999|10|5|1500|

1400|зелений|Renault|2020|6|4|1400|

1967|оранжевий|Volkswagen|2017|4|8|2400|

1344|чорний|BMW|2001|7.5|4|1400|

1434|білий|Ford|2003|2.3|10|4000|

1834|рожевий|Kia|2009|3.2|9|4800|

1814|коричневий|Nissan|2005|4.4|8|8000|

1561|фіолетовий|Skoda|2013|4.1|18|7600|

1122|синій|Toyota|2020|1.5|8|2000|

1174|сірий|Toyota|2017|4.4|6|5400|

1187|чорний|Mercedes|2005|5.1|10|1800|

Арк.  
101

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП – 12.00.00.000 ПЗ	

1182|білий|Toyota|2015|2.2|6|2400|

1194|зелений|Renault|1995|6.3|4|1400|

1123|червоний|Renault|1994|5.8|7|1200|

1114|фіолетовий|Volkswagen|2000|6.2|4|1300|

Вміст файлу **coaches.txt**:

0198|чорний|Ikarus|1999|32.6|38|400|

0401|синій|Hyundai|2020|16.3|56|600|

0067|червоний|Daewoo|2017|12.3|24|350|

0304|жовтий|Mercedes-Benz|2001|36.5|84|820|

0984|рожевий|MAN|2003|23|40|500|

0804|білий|Volvo|2009|34.2|48|550|

0814|чорний|Scania|2005|40.4|80|790|

0561|жовтий|VolgaBus|2013|40.1|76|750|

0522|фіолетовий|Ikarus|2007|11.5|20|210|

0274|червоний|Mercedes-Benz|2017|46.4|54|590|

0887|білий|Hyundai|2005|58.1|88|900|

0782|зелений|Ikarus|2015|12.2|24|350|

0294|чорний|Daewoo|1995|66.3|54|590|

0323|оранжевий|Scania|1994|32.8|42|510|

0314|білий|Ikarus|2000|16.2|30|380|

Арк.  
102

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП – 12.00.00.000 ПЗ	

Вміст файлу **trucks.txt**:

2298|червоний|Mercedes-Benz|1999|1500|7.6|

2400|зелений|Freightliner Trucks|2020|1400|15.2|

2967|оранжевий|Volkswagen Commercial Vehicles|2017|2400|45.4|

2344|чорний|Sterling Trucks|2001|1400|33.7|

2434|білий|Ford|2003|4000|1.5|

2834|рожевий|Iveco|2009|4800|2.8|

2814|коричневий|UD Nissan Diese|2005|8000|88.1|

2561|фіолетовий|Isuzu|2013|7600|14.5|

2122|синій|Mack|2020|2000|7.6|

2174|сірий|Magirus|2017|5400|6.7|

2187|чорний|Mercedes-Benz|2005|1800|4.5|

2182|білий|Astra|2015|2400|5.6|

2194|зелений|Renault|1995|1400|6.5|

2123|червоний|Renault|1994|1200|5.5|

2114|фіолетовий|Volkswagen|2000|1300|8.9

Вміст файлу **auxiliary.txt**:

3298|червоний|Polycar|1999|3000|15.2|5.65|2.5|

3400|зелений|Ford|2020|2800|30.4|6.65|3.5|

3967|оранжевий|Polycar|2017|4800|90.8|7.65|4.5|

Арк.  
103

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

3344|чорний|Ford|2001|2800|66.7|9.65|3.5|

3434|білий|Ford|2003|6000|15.1|7.65|2.5|

3834|рожевий|Polycar|2009|6000|12.8|8.65|2.0|

3814|коричневий|Ford|2005|9000|90|4.65|2.1|

3561|фіолетовий|Polycar|2013|8600|18.5|3.65|1.5|

3122|синій|Polycar|2020|3000|17.6|4.65|1.5|

3174|сірий|Ford|2017|5500|16.7|8.65|3.5|

3187|чорний|Polycar|2005|2800|14.5|4.65|0.5|

3182|білий|Ford|2015|3400|15.6|7.65|1.5|

3194|зелений|Polycar|1995|2400|16.5|5.6|3.5|

3123|червоний|Ford|1994|2200|15.5|5.5|1.5|

3114|фіолетовий|Polycar|2000|2300|18.9|4.6|1.5

Вміст файлу **garages.txt**:

270|164|24|78.4|133|32.9|37.9|29.6|76.7|28.4|211.2|85.1|43.7|99.2|110.8

Вміст файлу **drivers.txt**:

Петренко Олексій|27|4.5|397530|D|

Давиденко Денис|34|10.5|397531|D|

Легін Олег|41|12.5|397532|D|

Сегін Андрій|49|19.5|397533|D|

Борбулевич Антон|29|5.5|397534|D|

Арк.  
104

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Винник Арсен|38|8.5|397535|В|

Литвиненко Віктор|59|29.5|397536|В|

Карпенко Андрій|45|25.5|397537|В|

Глінський Борис|47|27.5|397538|В|

Тарасенко Гліб|50|30.5|397539|В|

Варварич Данило|58|35.5|397510|С|

Шевчук Михайло|43|13.5|397511|С|

Бойко Ярослав|33|13.5|397512|С|

Якименко Віталій|56|16.5|397513|С|

Гаврилюк Павло|39|19.5|397514|С|

Вміст файлу **staff.txt**:

Авраменко Олександр|56|36.5|6|

Травненко Кирило|43|23.5|5|

Давиденко Віталій|42|22.5|4|

Кісільовський Антон|34|14.5|2|

Тарновський Степан|35|15.5|3|

Федорчук Пилип|36|16.5|3|

Антонюк Михайло|20|1.5|1|

Мусякувич Ігор|21|2.5|2|

Арк.  
105

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Федорчук Микола|26|4.5|3|

Павленко Петро|24|3.5|2|

Вміст файлу **welders.txt**:

Многогрішний Григорій|39|18.5|4|3|

Тютюнник Григiр|41|21.5|5|4|

Квітковий Любомир|45|25.5|5|4|

Рожко Максим|35|13.5|4|2|

Чорний Вадим|23|2.5|1|1|

Вміст файлу **brigadiers.txt**:

Максименко\_Микола|34|14|

Винник\_Тарас|33|13|

Карий\_Олег|54|24|

Якименко\_Богдан|45|25|

Баран\_Юрій|47|27|

Ревенко\_Віктор|43|23|

Рева\_Віталій|39|19|

Авраменко\_Олександр|38|18|

Вміст файлу **masters.txt**:

Сметана\_Віктор|50|30|

Білий\_Роман|55|35|

Арк.  
106

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Палагнюк\_Денис|54|34|

Деркевич\_Тарас|43|23|

Вміст файлу **heads.txt**:

Бойко\_Богдан|47|20.5|

Антонюк\_Віталій|38|17.5|

Вміст файлу **runbus.txt**

1|6|2021|100|2|6|2021|200|3|6|2021|300|4|6|2021|400|5|6|2021|100|6|6|2021|600|7|6|2021|300|8|6|2021|200|9|6|2021|100|10|6|2021|200|11|6|2021|300|12|6|2021|200|13|6|2021|540|14|6|2021|490|15|6|2021|380|16|6|2021|345|17|6|2021|320|18|6|2021|120|19|6|2021|300|20|6|2021|340|21|6|2021|200|22|6|2021|300|23|6|2021|390|24|6|2021|347|25|6|2021|456|26|6|2021|500|27|6|2021|450|28|6|2021|390|29|6|2021|276|30|6|2021|300|31|6|2021|100

Вміст файлу **runtruck.txt**

1|6|2021|10|2|6|2021|20|3|6|2021|30|4|6|2021|40|5|6|2021|10|6|6|2021|60|7|6|2021|30|8|6|2021|20|9|6|2021|10|10|6|2021|20|11|6|2021|30|12|6|2021|20|13|6|2021|50|14|6|2021|40|15|6|2021|80|16|6|2021|45|17|6|2021|30|18|6|2021|10|19|6|2021|30|20|6|2021|34|21|6|2021|20|22|6|2021|30|23|6|2021|30|24|6|2021|37|25|6|2021|46|26|6|2021|50|27|6|2021|45|28|6|2021|39|29|6|2021|27|30|6|2021|30|31|6|2021|10

Вміст файлу **runaux.txt**


1|6|2021|1|2|6|2021|2|3|6|2021|3|4|6|2021|4|5|6|2021|1|6|6|2021|6|7|6|2021|3|8|6|2021|2|9|6|2021|1|10|6|2021|20|11|6|2021|3|12|6|2021|2|13|6|2021|5|14|6|2021|4|15|6|2021|8|16|6|2021|4|17|6|2021|3|18|6|2021|1|19|6|2021|3|20|6|2021|3|21|6|2021|2|22|6|2021|3|23|6|2021|3|24|6|2021|3|25|6|2021|4|26|6|2021|5|27|6|2021|4|28|6|2021|3|29|6|2021|2|30|6|2021|3|31|6|2021|1

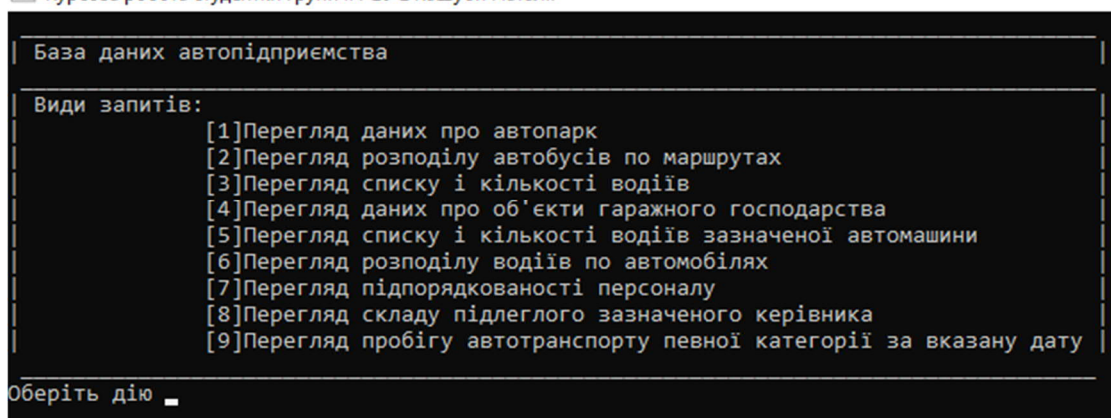
Арк.  
107

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Скріншоти етапів виконання.

Головне меню.

 Курсова робота студентки групи ІП-20-2 Кашуби Наталії



```
База даних автопідприємства

Види запитів:
[1]Перегляд даних про автопарк
[2]Перегляд розподілу автобусів по маршрутах
[3]Перегляд списку і кількості водіїв
[4]Перегляд даних про об'єкти гаражного господарства
[5]Перегляд списку і кількості водіїв зазначеної автомашини
[6]Перегляд розподілу водіїв по автомобілях
[7]Перегляд підпорядкованості персоналу
[8]Перегляд складу підлеглого зазначеного керівника
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію _
```

Арк.  
108

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



## Перегляд автопарку.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

База даних автопідприємства								
Види запитів:								
[1]Перегляд даних про автопарк								
[2]Перегляд розподілу автобусів по маршрутах								
[3]Перегляд списку і кількості водіїв								
[4]Перегляд даних про об'єкти гаражного господарства								
[5]Перегляд списку і кількості водіїв зазначеної автомашини								
[6]Перегляд розподілу водіїв по автомобілях								
[7]Перегляд підпорядкованості персоналу								
[8]Перегляд складу підлеглого зазначеного керівника								
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату								
Оберіть дію 1								
Маршрутні автобуси								
Номер	Колір	Марка	Рік виробництва	л бензину/100 м	Кількість місць	Маршрут		
0098	червоний	Ikarus	1999	32.6	38	Івано-Франківськ-Львів		
4400	зелений	Hyundai	2020	16.3	56	Івано-Франківськ-Одеса		
8967	оранжевий	Daewoo	2017	12.3	24	Івано-Франківськ-Тернопіль		
7344	чорний	Mercedes-Benz	2001	36.5	84	Івано-Франківськ-Київ		
3434	білий	MAN	2003	23	40	Івано-Франківськ-Ужгород		
6834	рожевий	Volvo	2009	34.2	48	Івано-Франківськ-Харків		
4814	коричневий	Scania	2005	40.4	80	Івано-Франківськ-Чернігів		
7561	фіолетовий	VolgaBus	2013	40.1	76	Івано-Франківськ-Чернівці		
6522	синій	Ikarus	2007	11.5	20	Івано-Франківськ-Тернопіль		
9274	сірий	Mercedes-Benz	2017	46.4	54	Івано-Франківськ-Чернівці		
8887	чорний	Hyundai	2005	58.1	88	Івано-Франківськ-Луцьк		
5782	білий	Ikarus	2015	12.2	24	Івано-Франківськ-Дніпро		
3294	зелений	Daewoo	1995	66.3	54	Івано-Франківськ-Київ		

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

1314	фіолетовий	Ikarus	2000	16.2	30	Івано-Франківськ-Чернівці
Таксі						
Номер	Колір	Марка	Рік виробництва	л бензину/100 м	Кількість місць	Вантажопідйомність (кг)
1198	червоний	Toyota	1999	10	5	1500
1400	зелений	Renault	2020	6	4	1400
1967	оранжевий	Volkswagen	2017	4	8	2400
1344	чорний	BMW	2001	7.5	4	1400
1434	білий	Ford	2003	2.3	10	4000
1834	рожевий	Kia	2009	3.2	9	4800
1814	коричневий	Nissan	2005	4.4	8	8000
1561	фіолетовий	Skoda	2013	4.1	18	7600
1122	синій	Toyota	2020	1.5	8	2000
1174	сірий	Toyota	2017	4.4	6	5400
1187	чорний	Mercedes	2005	5.1	10	1800
1182	білий	Toyota	2015	2.2	6	2400
1194	зелений	Renault	1995	6.3	4	1400

Арк.  
109

Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

0114	фіолетовий	Volkswagen	2000	6.2	4	1300	
Туристичні автобуси							
Номер	Колір	Марка	Рік виробництва	л бензину/100 м	Кількість місць	Вартість оренди за 1 год (грн)	
0198	чорний	Ikarus	1999	32.6	38	400	
0401	синій	Hyundai	2020	16.3	56	600	
0067	червоний	Daewoo	2017	12.3	24	350	
0304	жовтий	Mercedes-Benz	2001	36.5	84	820	
0984	рожевий	MAN	2003	23	40	500	
0804	білий	Volvo	2009	34.2	48	550	
0814	чорний	Scania	2005	40.4	80	790	
0561	жовтий	VolgaBus	2013	40.1	76	750	
0522	фіолетовий	Ikarus	2007	11.5	20	210	
0274	червоний	Mercedes-Benz	2017	46.4	54	590	
0887	білий	Hyundai	2005	58.1	88	900	
0782	зелений	Ikarus	2015	12.2	24	350	
0294	чорний	Daewoo	1995	66.3	54	590	

0314	білий	Ikarus	2000	16.2	30	380	
Вантажівки							
Номер	Колір	Марка	Рік виробництва	л бензину/100 м	Вантажопідйомність (т)		
2298	червоний	Mercedes-Benz	1999	1500	7.6		
2400	зелений	Freightliner Trucks	2020	1400	15.2		
2967	оранжевий	Volkswagen Commercial Vehicles	2017	2400	45.4		
2344	чорний	Sterling Trucks	2001	1400	33.7		
2434	білий	Ford	2003	4000	1.5		
2834	рожевий	Iveco	2009	4800	2.8		
2814	коричневий	UD Nissan Diese	2005	8000	88.1		
2561	фіолетовий	Isuzu	2013	7600	14.5		
2122	синій	Mack	2020	2000	7.6		
2174	сірий	Magirus	2017	5400	6.7		
2187	чорний	Mercedes-Benz	2005	1800	4.5		
2182	білий	Astra	2015	2400	5.6		
2194	зелений	Renault	1995	1400	6.5		

## Допоміжний транспорт

	Номер	Колір	Марка	Рік виробництва	л бензину/100 м	Вантажопідйомність (т)	Довжина	Ширина
	3298	червоний	Polycar	1999	3000	15.2	5.65	2.5
3400	зелений	Ford	2020	2800	30.4	6.65	3.5	
3967	оранжевий	Polycar	2017	4800	90.8	7.65	4.5	
3344	чорний	Ford	2001	2800	66.7	9.65	3.5	
3434	білий	Ford	2003	6000	15.1	7.65	2.5	
3834	рожевий	Polycar	2009	6000	12.8	8.65	2	
3814	коричневий	Ford	2005	9000	90	4.65	2.1	
3561	фіолетовий	Polycar	2013	8600	18.5	3.65	1.5	
3122	синій	Polycar	2020	3000	17.6	4.65	1.5	
3174	сірий	Ford	2017	5500	16.7	8.65	3.5	
3187	чорний	Polycar	2005	2800	14.5	4.65	0.5	
3182	білий	Ford	2015	3400	15.6	7.65	1.5	
3194	зелений	Polycar	1995	2400	16.5	5.6	3.5	

# Перегляд водіїв автопідприємства.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

- [3]Перегляд списку і кількості водіїв
- [4]Перегляд даних про об'єкти гаражного господарства
- [5]Перегляд списку і кількості водіїв зазначеної автомашини
- [6]Перегляд розподілу водіїв по автомобілях
- [7]Перегляд підпорядкованості персоналу
- [8]Перегляд складу підлеглого зазначеного керівника
- [9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 3

Прізвище та ім'я	Вік	Досвід роботи	Номер ліцензії	Категорія
Петренко Олексій	27	4.5	397530	D
Давиденко Денис	34	10.5	397531	D
Легін Олег	41	12.5	397532	D
Сегін Андрій	49	19.5	397533	D
Борбулевич Антон	29	5.5	397534	D
Винник Арсен	38	8.5	397535	B
Литвиненко Віктор	59	29.5	397536	B
Карпенко Андрій	45	25.5	397537	B
Глінський Борис	47	27.5	397538	B
Тарасенко Гліб	50	30.5	397539	B
Варварич Данило	58	35.5	397510	C
Шевчук Михайло	43	13.5	397511	C
Бойко Ярослав	33	13.5	397512	C
Якименко Віталій	56	16.5	397513	C
Гаврилюк Павло	39	19.5	397514	C

Загальне число водіїв автопідприємства: 15.

Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ

## Перегляд об'єктів гаражного господарства.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

```
[4]Перегляд даних про об'єкти гаражного господарства
[5]Перегляд списку і кількості водіїв зазначеної автомашини
[6]Перегляд розподілу водіїв по автомобілях
[7]Перегляд підпорядкованості персоналу
[8]Перегляд складу підлеглого зазначеного керівника
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату
```

Оберіть дію 4

Наявність об'єктів гаражного господарства в цілому

Номер	Площа
1	270
2	164
3	24
4	78.4
5	133
6	32.9
7	37.9
8	29.6
9	76.7
10	28.4
11	211.2
12	85.1
13	43.7
14	99.2
15	110.8

Арк.  
113

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

### Наявність об'єктів гаражного господарства для легкових автомобілів

Номер	Площа
3	24
6	32.9
8	29.6
10	28.4

### Наявність об'єктів гаражного господарства для автобусів

Номер	Площа
4	78.4
7	37.9
9	76.7
13	43.7

### Наявність об'єктів гаражного господарства для вантажівок

Номер	Площа
12	85.1
14	99.2
15	110.8

### Наявність об'єктів гаражного господарства для допоміжного транспорту

Номер	Площа
1	270
2	164
5	133



## Перегляд водіїв зазначеної автомашини.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

База даних автопідприємства

Види запитів:

- [1]Перегляд даних про автопарк
- [2]Перегляд розподілу автобусів по маршрутах
- [3]Перегляд списку і кількості водіїв
- [4]Перегляд даних про об'єкти гаражного господарства
- [5]Перегляд списку і кількості водіїв зазначеної автомашини
- [6]Перегляд розподілу водіїв по автомобілях
- [7]Перегляд підпорядкованості персоналу
- [8]Перегляд складу підлеглого зазначеного керівника
- [9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 5

Введіть номер автомашини 0098

Прізвище та ім'я	Номер посвідчення
Петренко Олексій	397530
Давиденко Денис	397531
Легін Олег	397532

За машиною закріплено 3 водіїв.

Арк.  
115

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## Перегляд розподілу автобусів по маршрутах.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

База даних автопідприємства

Види запитів:  
[1]Перегляд даних про автопарк  
[2]Перегляд розподілу автобусів по маршрутах  
[3]Перегляд списку і кількості водіїв  
[4]Перегляд даних про об'єкти гаражного господарства  
[5]Перегляд списку і кількості водіїв зазначеної автомашини  
[6]Перегляд розподілу водіїв по автомобілях  
[7]Перегляд підпорядкованості персоналу  
[8]Перегляд складу підлеглого зазначеного керівника  
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 2

Номер	Маршрут
0098	Івано-Франківськ-Львів
4400	Івано-Франківськ-Одеса
8967	Івано-Франківськ-Тернопіль
7344	Івано-Франківськ-Київ
3434	Івано-Франківськ-Ужгород
6834	Івано-Франківськ-Харків
4814	Івано-Франківськ-Чернігів
7561	Івано-Франківськ-Чернівці
6522	Івано-Франківськ-Тернопіль
9274	Івано-Франківськ-Чернівці
8887	Івано-Франківськ-Луцьк
5782	Івано-Франківськ-Дніпро
3294	Івано-Франківськ-Київ

Арк.  
116

Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



## Перегляд розподілу водіїв по автомобілях.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

База даних автопідприємства

Види запитів:

[1]Перегляд даних про автопарк

[2]Перегляд розподілу автобусів по маршрутах

[3]Перегляд списку і кількості водіїв

[4]Перегляд даних про об'єкти гаражного господарства

[5]Перегляд списку і кількості водіїв зазначеної автомашини

[6]Перегляд розподілу водіїв по автомобілях

[7]Перегляд підпорядкованості персоналу

[8]Перегляд складу підлеглого зазначеного керівника

[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 6

Маршрутні автобуси

Номер автомашини 0098

Прізвище та ім'я	Номер посвідчення
Петренко Олексій	397530
Давиденко Денис	397531
Легін Олег	397532

За машиною закріплено 3 водіїв.

Номер автомашини 4400

Прізвище та ім'я	Номер посвідчення
Петренко Олексій	397530
Давиденко Денис	397531
Легін Олег	397532

За машиною закріплено 3 водіїв.

Номер автомашини 8967

Прізвище та ім'я	Номер посвідчення
Петренко Олексій	397530

Арк.  
117

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Туристичні автобуси  
Номер автомашини 0198

Прізвище та ім'я	Номер посвідчення
Сегін Андрій	397533
Борбулевич Антон	397534

За машиною закріплено 2 водіїв.  
Номер автомашини 0401

Прізвище та ім'я	Номер посвідчення
Сегін Андрій	397533
Борбулевич Антон	397534

За машиною закріплено 2 водіїв.  
Номер автомашини 0067

Прізвище та ім'я	Номер посвідчення
Сегін Андрій	397533
Борбулевич Антон	397534

За машиною закріплено 2 водіїв.  
Номер автомашини 0304

Прізвище та ім'я	Номер посвідчення
Сегін Андрій	397533
Борбулевич Антон	397534

За машиною закріплено 2 водіїв.  
Номер автомашини 0984

Прізвище та ім'я	Номер посвідчення
Сегін Андрій	397533

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Таксі

Номер автомашини 1198

Прізвище та ім'я	Номер посвідчення
Винник Арсен	397535
Литвиненко Віктор	397536
Карпенко Андрій	397537
Глінський Борис	397538
Тарасенко Гліб	397539

За машиною закріплено 5 водіїв.

Номер автомашини 1400

Прізвище та ім'я	Номер посвідчення
Винник Арсен	397535
Литвиненко Віктор	397536
Карпенко Андрій	397537
Глінський Борис	397538
Тарасенко Гліб	397539

За машиною закріплено 5 водіїв.

Номер автомашини 1967

Прізвище та ім'я	Номер посвідчення
Винник Арсен	397535
Литвиненко Віктор	397536
Карпенко Андрій	397537
Глінський Борис	397538
Тарасенко Гліб	397539

Арк.  
119

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Вантажівки

Номер автомашини 2298

Прізвище та ім'я	Номер посвідчення
Варварич Данило	397510
Шевчук Михайло	397511
Бойко Ярослав	397512

За машиною закріплено 3 водіїв.

Номер автомашини 2400

Прізвище та ім'я	Номер посвідчення
Варварич Данило	397510
Шевчук Михайло	397511
Бойко Ярослав	397512

За машиною закріплено 3 водіїв.

Номер автомашини 2967

Прізвище та ім'я	Номер посвідчення
Варварич Данило	397510
Шевчук Михайло	397511
Бойко Ярослав	397512

За машиною закріплено 3 водіїв.

Номер автомашини 2344

Прізвище та ім'я	Номер посвідчення
Варварич Данило	397510
Шевчук Михайло	397511
Бойко Ярослав	397512

Арк.  
120

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

Допоміжний транспорт  
Номер автомашини 3298

Прізвище та ім'я	Номер посвідчення
Якименко Віталій	397513
Гаврилюк Павло	397514

За машиною закріплено 2 водіїв.  
Номер автомашини 3400

Прізвище та ім'я	Номер посвідчення
Якименко Віталій	397513
Гаврилюк Павло	397514

За машиною закріплено 2 водіїв.  
Номер автомашини 3967

Прізвище та ім'я	Номер посвідчення
Якименко Віталій	397513
Гаврилюк Павло	397514

За машиною закріплено 2 водіїв.  
Номер автомашини 3344

Прізвище та ім'я	Номер посвідчення
Якименко Віталій	397513
Гаврилюк Павло	397514

За машиною закріплено 2 водіїв.  
Номер автомашини 3434

Прізвище та ім'я	Номер посвідчення
Якименко Віталій	397513

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



## Перегляд підпорядкованості персоналу.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

```
База даних автопідприємства

Види запитів:
[1]Перегляд даних про автопарк
[2]Перегляд розподілу автобусів по маршрутах
[3]Перегляд списку і кількості водіїв
[4]Перегляд даних про об'єкти гаражного господарства
[5]Перегляд списку і кількості водіїв зазначеної автомашини
[6]Перегляд розподілу водіїв по автомобілях
[7]Перегляд підпорядкованості персоналу
[8]Перегляд складу підлеглого зазначеного керівника
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 7
      Прізвище та ім'я голови цеху Бойко_Богдан
      Прізвище та ім'я майстра Сметана_Віктор
      Прізвище та ім'я бригадира Максименко_Микола
Многогрішний Григорій
Тютюнник Григорі
Квітковий Любомир
Рожко Максим
      Прізвище та ім'я бригадира Винник_Тарас
Тютюнник Григорі
Авраменко Олександр
Гравненко Кирило
      Прізвище та ім'я майстра Білий_Роман
      Прізвище та ім'я бригадира Карий_Олег
Давиденко Віталій
Кісільовський Антон
      Прізвище та ім'я бригадира Якименко_Богдан
Гарновський Степан
Федорчук Пилип
```

```
      Прізвище та ім'я голови цеху Антонюк Віталій
      Прізвище та ім'я майстра Палагнюк Денис
      Прізвище та ім'я бригадира Баран Юрій
Антонюк Михайло
Мусякувич Ігор
Федорчук Микола
Павленко Петро
      Прізвище та ім'я бригадира Ревенко Віктор
Петренко Олексій
Давиденко Денис
Легін Олег
Сегін Андрій
Борбулевич Антон
      Прізвище та ім'я майстра Деркевич Тарас
      Прізвище та ім'я бригадира Рева Віталій
Винник Арсен
Литвиненко Віктор
Карпенко Андрій
Глінський Борис
Тарасенко Гліб
      Прізвище та ім'я бригадира Авраменко Олександр
Варварич Данило
Шевчук Михайло
Бойко Ярослав
Якименко Віталій
Гаврилюк Павло
```

Арк.  
122

	Зм	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

## Перегляд складу підлеглого зазначеного керівника.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

База даних автопідприємства

Види запитів:

[1]Перегляд даних про автопарк

[2]Перегляд розподілу автобусів по маршрутах

[3]Перегляд списку і кількості водіїв

[4]Перегляд даних про об'єкти гаражного господарства

[5]Перегляд списку і кількості водіїв зазначеної автомашини

[6]Перегляд розподілу водіїв по автомобілях

[7]Перегляд підпорядкованості персоналу

[8]Перегляд складу підлеглого зазначеного керівника

[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 8

Введіть прізвище та ім'я керівника через нижнє підкреслення \_ Максименко\_Микола

Прізвище та ім'я бригадира Максименко\_Микола

Многогрішний Григорій

Тютюнник Григорі

Квітковий Любомир

Рожко Максим

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

База даних автопідприємства

Види запитів:

[1]Перегляд даних про автопарк

[2]Перегляд розподілу автобусів по маршрутах

[3]Перегляд списку і кількості водіїв

[4]Перегляд даних про об'єкти гаражного господарства

[5]Перегляд списку і кількості водіїв зазначеної автомашини

[6]Перегляд розподілу водіїв по автомобілях

[7]Перегляд підпорядкованості персоналу

[8]Перегляд складу підлеглого зазначеного керівника

[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 8

Введіть прізвище та ім'я керівника через нижнє підкреслення \_ Палагнюк\_Денис

Прізвище та ім'я майстра Палагнюк\_Денис

Прізвище та ім'я бригадира Баран\_Юрій

Антонюк Михайло

Мусякувич Ігор

Федорчук Микола

Павленко Петро

Прізвище та ім'я бригадира Ревенко\_Віктор

Петренко Олексій

Давиденко Денис

Легін Олег

Сегін Андрій

Борбулевич Антон

Арк.  
123

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	

# База даних автопідприємства

## Види запитів:

- [1]Перегляд даних про автопарк
- [2]Перегляд розподілу автобусів по маршрутах
- [3]Перегляд списку і кількості водіїв
- [4]Перегляд даних про об'єкти гаражного господарства
- [5]Перегляд списку і кількості водіїв зазначеної автомашини
- [6]Перегляд розподілу водіїв по автомобілях
- [7]Перегляд підпорядкованості персоналу
- [8]Перегляд складу підлеглого зазначеного керівника
- [9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 8

Введіть прізвище та ім'я керівника через нижнє підкреслення \_ Бойко\_Богдан

Прізвище та ім'я голови цеху Бойко\_Богдан

Прізвище та ім'я майстра Сметана\_Віктор

Прізвище та ім'я бригадира Максименко\_Микола

Многогрішний Григорій

Тютюнник Григiр

Квітковий Любомир

Рожко Максим

Прізвище та ім'я бригадира Винник\_Тарас

Тютюнник Григiр

Авраменко Олександр

Травненко Кирило

Прізвище та ім'я майстра Білий\_Роман

Прізвище та ім'я бригадира Карий\_Олег

Давиденко Віталій

Кісільовський Антон

Прізвище та ім'я бригадира Якименко\_Богдан

Тарновський Степан

Федорчук Пилип

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	



## Перегляд пробігу автотранспорту певної категорії за вказану дату.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

```
База даних автопідприємства

Види запитів:
[1]Перегляд даних про автопарк
[2]Перегляд розподілу автобусів по маршрутах
[3]Перегляд списку і кількості водіїв
[4]Перегляд даних про об'єкти гаражного господарства
[5]Перегляд списку і кількості водіїв зазначеної автомашини
[6]Перегляд розподілу водіїв по автомобілях
[7]Перегляд підпорядкованості персоналу
[8]Перегляд складу підлеглого зазначеного керівника
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 9
Введіть категорію через нижнє підкреслення _ Вантажний_транспорт
Введіть день 4
Введіть місяць 6
Введіть рік 2021
2298 40 км
2400 40 км
2967 40 км
2344 40 км
2434 40 км
2834 40 км
2814 40 км
2561 40 км
2122 40 км
2174 40 км
2187 40 км
2182 40 км
2194 40 км
2123 40 км
2114 40 км
```

## Реакція системи у разі некоректного введення запиту.

Курсова робота студентки групи ІП-20-2 Кашуби Наталії

```
База даних автопідприємства

Види запитів:
[1]Перегляд даних про автопарк
[2]Перегляд розподілу автобусів по маршрутах
[3]Перегляд списку і кількості водіїв
[4]Перегляд даних про об'єкти гаражного господарства
[5]Перегляд списку і кількості водіїв зазначеної автомашини
[6]Перегляд розподілу водіїв по автомобілях
[7]Перегляд підпорядкованості персоналу
[8]Перегляд складу підлеглого зазначеного керівника
[9]Перегляд пробігу автотранспорту певної категорії за вказану дату

Оберіть дію 0
Ви ввели неправильне значення.
```

Арк.  
125

	Зм.	Арк№ докум.	Підп.	Дата	КР.ІП –12.00.00.000 ПЗ	