**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра автоматизації проектування  
енергетичних процесів і систем**

***КУРСОВА РОБОТА***

**з дисципліни: «Об’єктно-орієнтоване програмування»**

**тема:**

**«Створення моделей об’єктів реального світу на мові C++»**

|  |  |
| --- | --- |
| **Керівник – Карпенко С.Г.** | **Виконав Григорук П.О.** |
| **Допущено до захисту** | **Студент 2-го курсу** |
| **\_\_ \_\_\_\_\_\_\_\_ 2020 р.** | **Групи ТР-91** |
| **Захищено з оцінкою** | **залікова книжка № ТР-91101** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

***2020***

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра автоматизації проектування  
енергетичних процесів і систем

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

(назва дисципліни)

на тему:

«Створення моделей об’єктів реального світу на мові C++»

Студента групи **ТР-91**

спеціальність **122 «Комп’ютерні науки»**

освітня програма Комп’ютерний моніторинг та геометричне моделювання процесів та систем

Григорука П.О.

(прізвище та ініціали)

Керівник доцент, канд. фіз.-мат. наук Карпенко С.Г.

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

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_ доц., к. ф.-м. н. Карпенко С.Г.

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

\_\_\_\_\_\_\_\_\_\_\_ доц., к.е.н. Левченко Л.О.

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

\_\_\_\_\_\_\_\_\_\_\_ доц., к.т.н. Шаповалова С.І.

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

Київ - 2020 рік

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет ТЕПЛОЕНЕРГЕТИЧНИЙ ФАКУЛЬТЕТ

( повна назва)

Кафедра автоматизації проектування енергетичних процесів та систем

(повна назва)

Освітньо-кваліфікаційний рівень бакалавр

спеціальність **122 «Комп’ютерні науки»**

освітня програма **Комп’ютерний моніторинг та  
геометричне моделювання процесів та систем**

ЗАВДАННЯ

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Григоруку Петру Олексійовичу

(прізвище, ім’я, по батькові)

1. Тема роботи «Створення моделей об’єктів реального світу на мові C++»

керівник курсової роботи –

Карпенко Станіслав Григорович, канд. фіз.-мат. наук, доцент

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи – 19 грудня 2020 р.

3. Початкові дані до проекту (роботи): мова C++, модель об’єкту реального світу.

4. Зміст пояснювальної записки курсової роботи (перелік питань, які потрібно розробити) – Розробити та обґрунтувати ієрархічну модель класів для опису об’єкту реального світу, використовуючи інструментальні можливості мови C++, зокрема успадкування, інкапсуляцію, перевантажені операції, шаблони функцій, запис у двійкові та текстові файли та обробку виключень.

5. Перелік графічного матеріалу –

графічне зображення ієрархічної моделі класів з урахуванням як успадкування, так й включення; відображення значень основних параметрів реальних об’єктів у вигляді таблиці.

6. Дата видачі завдання – 03 листопада 2020 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсової роботи | Строк виконання етапів роботи | Примітка |
|  | Побудова ієрархічної моделі класів | 17.11.2020 | **+** |
|  | Розробка та застосування інструментальних засобів обробки даних | 01.12.2020 | **+** |
|  | Написання розрахунково-пояснювальної записки | 19.12.2020 | **+** |

**Студент \_\_\_\_\_\_\_\_\_ Григорук П.О.**

( підпис ) (прізвище та ініціали)

**Керівник курсової роботи \_\_\_\_\_\_\_\_\_ Карпенко С.Г.**

( підпис ) (прізвище та ініціали)

# Анотація

Документ слугує для ознайомлення з програмою курсової роботи.

Програма зчитує дані з фалу. Може з текстового , а може з бінарного. Так само це стосується запису. В програмі є меню. В програмі є обробка виключних ситуацій.

Дані можна вводити вручну. Програма підраховує середні значення за тиждень , конкретних величин, згідно з варіантом.

В цьому документі є опис конкретних поставлених питань. Також тут є основні потрібні для цього опису елементи структури – основна частина , додатки тощо. У окремому файлі міститься код цілком.

# ЗМІСТ

1. [Вступ x](#_ВСТУП)
   1. [Актуальність теми, ії значимість x](#_Актуальність_теми,_ії)
   2. [Мета і завдання курсової роботи x](#_Мета_і_завдання)
   3. [Методи дослідження, що використовувались під час написання x](#_Методи_дослідження,_що)
2. [Основна частина x](#_ОСНОВНА_ЧАСТИНА)
   1. [Постановка задачі згідно з варіантом x](#_Постановка_задачі_згідно)
   2. [Виконання задачі x](#_ВИКОНАННЯ_ЗАДАЧІ)
   3. [Ієрархічна модель графічно x](#_Ієрархічна_модель_графічно)
      1. [Взаємозв’язки між класами x](#_Взаємозв'язки_між_класами)
      2. [Вибір обмежєнь для полів x](#_Вибір_обмежень_для)
      3. [Вибір перевантажень операцій x](#_Вибір_перевантажень_операцій)
      4. [Фрагменти коду , для яких треба встановити обробку виключних ситуацій x](#_Фрагменти_коду,_для)
      5. [Шаблонна функція x](#_Шаблонна_функція)
   4. [Невеликий підсумок до основної частини x](#_Невеликий_підсумок_до)
3. [Висновки x](#_Висновки)
   1. [Підсумки з освітлених питань , шляхи вирішення аспектів проблеми теми x](#_Підсумки_з_освітлених)
   2. [Рекомендації x](#_Рекомендації)
4. [Список використаних джерел x](#_Список_використаних_джерел)
5. [Додатки x](#_Додатки_1)
   1. [Додаток 1 x](#_Додаток_1)
   2. [Додаток 2 x](#_Додаток_2)
   3. [Додаток 3 x](#_Додаток_3)

Зм.

Лист

№ докум.

Підпис

Дата

Аркуш

1

УКР.НТУУ “КПІ імені І.Сікорського”\_ТЕФ\_АПЕПС\_TР91101 3 КР 81-1

Розробив

Григорук П.О.

Перевірив

Карпенко С.Г.

Рецензент.

Норм Контр.

Кузьміна Л.П.

Затвердив

Коваль О.В.

Назва теми

«Створення моделей об’єктів реального світу на мові C++»

Лит.

Аркушів

1

ТЕФ

# ВСТУП

[**C++** (**Сі-плюс-плюс**)[1]](#_Список_використаних_джерел)— мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом (англ. *Bjarne Stroustrup*) в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові програмування С. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014.

У 1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. С++ суттєво вплинула на інші популярні сьогодні мови програмування: С# та Java.

## Актуальність теми, ії значимість

Темою курсової роботи було задано “Створення моделей об’єктів реального свту на мові C++”. Актуальність даної теми пояснюється тісним пов’язанням концепцій, парадигм, відношень у мові C++ з реальним світом.

Люди сворювали перші прототипи літальних апаратів, надхненні природою : пташками, метеликами, тощо. Так само об’єктно орінтоване програмування було створене з використанням базових відношень між об’єктами в реальному житті. Це дає змогу програмісту розуміти ООП простіше й ефективніше. Саме тому значимість теми складно переоцінити.

## Мета і завдання курсової роботи

Метою роботи є вивчення і закріплення пройденного матеріалу задля покращення професійного рівня володіння мовою C++ під час виконання наступного завдання:

Розробити та обґрунтувати ієрархічну модель класів для опису об’єкту реального світу, використовуючи інструментальні можливості мови C++, зокрема успадкування, інкапсуляцію, перевантажені операції, шаблони функцій, запис у двійкові та текстові файли та обробку виключень.

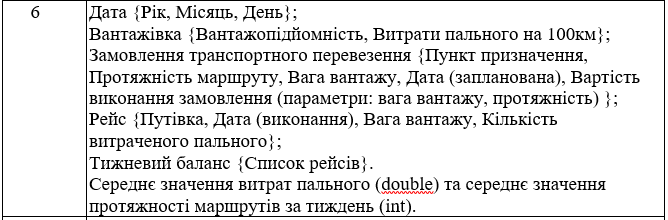
## Методи дослідження, що використовувались під час написання

Спостереження, порівняння, узагальнення, абстрагування, формалізація, конкретизація, декомпозиція тощо. Ці методи домогли в створенні правильної іїрархії класів.

Взаємодію з користувачем зробили методом аналізу та експерименту.

# ОСНОВНА ЧАСТИНА

## Постановка задачі згідно з варіантом



Створити дану ієрархію. Оформити ії графічно на uml. Також виконати усі вимоги, поставлені нам.

## ВИКОНАННЯ ЗАДАЧІ

Для виконання заданої задачі нам потрібно знати та вміти використовувати ооп термінологію. Давайте дамо визначення усім необхідним термінам, відношенням тощо, щобо посилатись на них далі.

#### **Визначення ООП**

[Об'єктно-орієнтоване програмування[2]](#_Список_використаних_джерел) — це метод програмування, заснований на поданні програми як сукупності взаємодіючих об'єктів, кожен з яких є примірником певного класу, а класи є членами певної ієрархії наслідування.

#### **Фундаментальні поняття**

В результаті [дослідження[3]](#_Список_використаних_джерел) Дебори Дж. Армстронг (англ. Deborah J. Armstrong) комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень об'єктно-орієнтованого програмування. До них належить:

**Клас**

Клас визначає абстрактні характеристики деякої сутності, включно з характеристиками самої сутності (її **атрибутами** або **властивостями**) та діями, які вона здатна виконувати (її **поведінкою**, **методами** або **можливостями**). Наприклад, клас Собака може характеризуватись рисами, притаманними всім собакам, зокрема: порода, колір хутра, здатність гавкати. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Зазвичай клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його **членами**.

**Об'єкт**

Окремий екземпляр класу (створюється після запуску програми і ініціалізації полів класу). Клас Собака відповідає всім собакам шляхом опису їхніх спільних рис; об'єкт Сірко є одним окремим собакою, окремим варіантом значень характеристик. Собака має хутро; Сірко має коричнево-біле хутро. Об'єкт Сірко є **екземпляром** (**примірником**) класу Собака. Сукупність значень атрибутів окремого об'єкта називається **станом**. На основі класу Собака можна, також, створити інший об'єкт Дружок, який відрізнятиметься від об'єкта Сірко своїм станом (наприклад кольором хутра). Обидва об'єкта (Сірко і Дружок) є екземплярами класу Собака.

**Метод**

Можливості об'єкта. Оскільки Сірко — Собака, він може гавкати. Тому гавкати() є одним із методів об'єкта Сірко. Він може мати й інші методи, зокрема: місце(), або їсти(). В межах програми, використання методу має впливати лише на один об'єкт; всі Собаки можуть гавкати, але треба щоб гавкав лише один окремий собака.

**Успадкування (наслідування)**

Клас може мати «підкласи», спеціалізовані, розширені версії надкласу. Можуть навіть утворюватись цілі дерева успадкування. Наприклад, клас Собака може мати підкласи Коллі, Пекінес, Вівчарка тощо. Так, Сірко може бути екземпляром класу Вівчарка. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть вводити свої власні. Успадкування може бути одиничне (один безпосередній батьківський клас) та множинне (кілька батьківських класів). Це залежить від вибору програміста, який реалізовує клас та мови програмування. Так, наприклад, в Java дозволене лише одинарне успадкування, а в С++ і те і інше.

**Приховування інформації (інкапсуляція)**

Приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення. Так, наприклад, клас Собака має метод гавкати(). Реалізація цього методу описує як саме повинно відбуватись гавкання (приміром, спочатку вдихнути() а потім видихнути() на обраній частоті та гучності). Петро, хазяїн пса Сірка, не повинен знати як він гавкає. Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт надає кожному іншому класу певний інтерфейс — члени, доступні іншим класам. Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дасть змогу полегшити внесення змін без потреби змінювати і користувачів інтерфейсу. Наприклад, інтерфейс може гарантувати, що щенята можуть додаватись лише до об'єктів класу Собака кодом самого класу. Часто, члени класу позначаються як **публічні** (англ. public), **захищені** (англ. protected) та **приватні** (англ. private), визначаючи, чи доступні вони всім класам, підкласам, або лише до класу в якому їх визначено. Деякі мови програмування йдуть ще далі: Java використовує ключове слово **private** для обмеження доступу, що буде дозволений лише з методів того самого класу, **protected** — лише з методів того самого класу і його нащадків та з класів із того ж самого пакету, C# та VB.NET відкривають деякі члени лише для класів із тієї ж збірки шляхом використання ключового слова **internal** (C#) або **Friend** (VB.NET), а Eiffel дозволяє вказувати які класи мають доступ до будь-яких членів.

**Абстрагування**

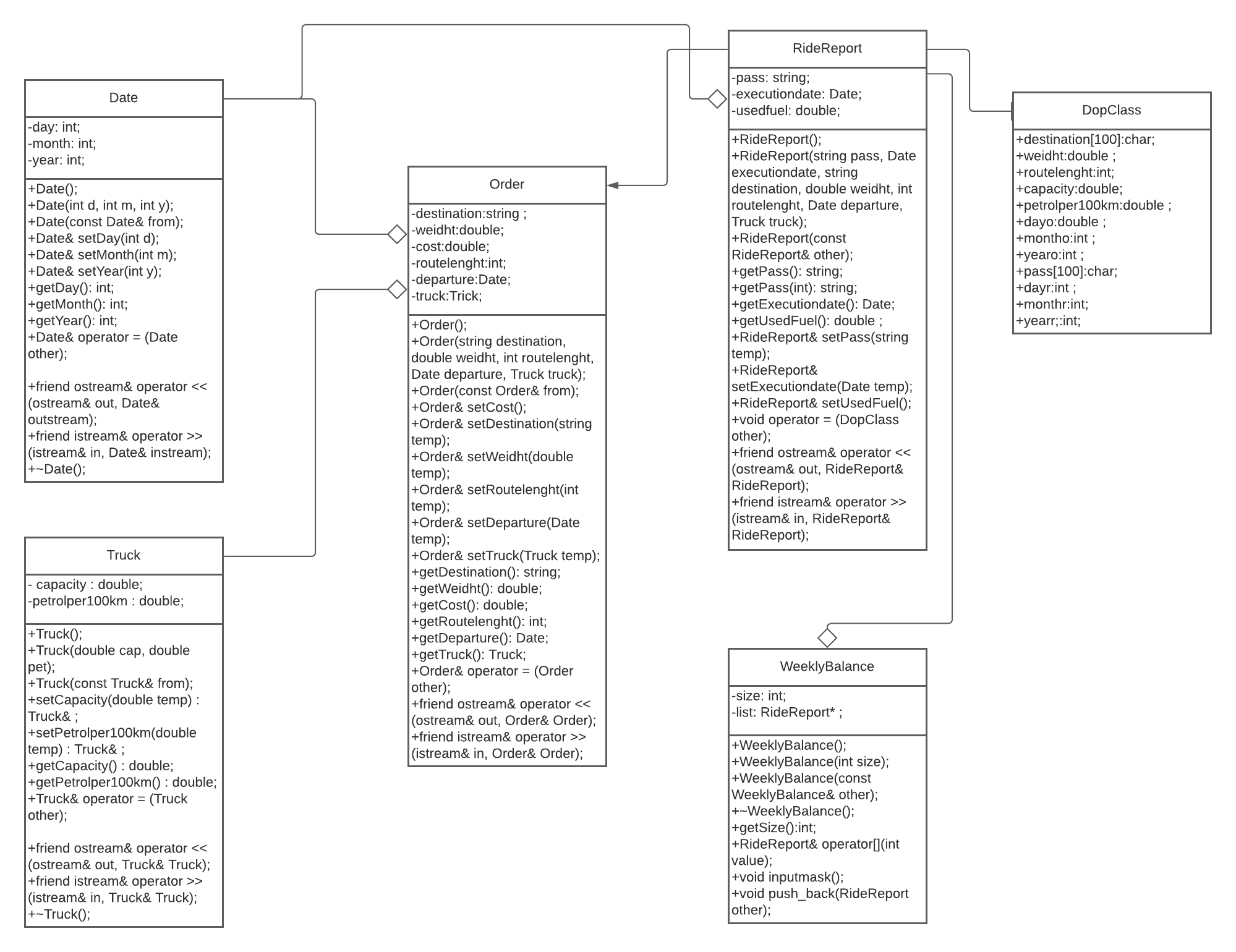
Спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми. Наприклад Собака Сірко більшу частину часу може розглядатись як Собака, а коли потрібно отримати доступ до інформації специфічної для собак породи коллі — як Коллі і як Тварина (можливо, батьківський клас Собака) під час підрахунку тварин Петра.

**Поліморфізм**

Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. Наприклад, якщо Собака отримує команду голос(), то у відповідь можна отримати Гав; якщо Свиня отримує команду голос (), то у відповідь можна отримати Рох-рох. На практиці - це реалізовується шляхом реалізації ряду підпрограм (функцій, процедур, методи тощо) з однаковими іменами, але з різними параметрами. Залежно від того, що передається, і вибирається відповідна підпрограма.

### Ієрархічна модель графічно

[Диаграмма классов (англ. class diagram) [4]](#_Список_використаних_джерел)— структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.



### Взаємозв'язки між класами

Клас Date є базовим класом.

Клас Truck є базовим класом.

Клас Order є базовим класом, у Order агрегований клас Date, бо замовлення має містити дату , але не є нею. Так само це стосується агрегації Truck в Order: це обумовлено тим самим чином : Truck існує незалежно, проте, Order повинен мати розуміння, який Truck йому використовувати. Про використання – далі.

Клас RideReport Наслідується від Order. Аргументуємо це тим, що RideReport – це по суті, результат виконання Order (у реальному житті) , а , відповідно, RideReport повинен мати деякі поля з Order. До того ж , агрегуємо Date до RideReport. Нам необхідно зберігати одну дату для замовлення, а одну – для його виконання, тож агрегація Date є у Order і у RideReport ,ці дати – різні.

Клас WeeklyBalance містить список RideReport’ів , тож агрегуємо RideReport у нього.

### Вибір обмежень для полів

Ми беремо тип int , тип double тощо, як основні, використовуємо відопідно те , що більш підходить для наших ситуацій. Не використовуємо long double, якщо можна просто double тощо.

Якщо, наприклад маємо масив char, то його довжина буде не більше, ніж може бути фактично це слово. Давайте наведемо приклади подібних обмежень

**class Date {**

**int day, month, year;**

**…**

**class Order**

**{**

**string destination;**

**double weidht, cost;**

**…**

**class DopClass**

**{**

**public:**

**char destination[100];**

### Вибір перевантажень операцій

Почнему з класу Date.

Перевантажимо операцію = , щоб, наприклад, можна було присвоїти одній даті значення іншої . І написати date1 = date2 . У випадку, коли, наприклад, 2 події пройшли в один день(рік,місяць) .

Перевантажимо >> для того, щоб зчитувати з файлу чи консолі поля Date. Перевантажими << , щоб виводити дані у файл та консоль. Робота з вводом-виводом інформації.

Клас Truck

Перевантажимо операцію =, щоб, наприклад, при виникненні ситуації, коли потрібно зробити різні, але з однаковими зарактеристиками вантажівки, ми могли це зрубити не стандрартним чином, який пропонує нам компілятор, а так, якби об’єкти класу Truck були звичайними змінними.

Перевантажимо >> для того, щоб зчитувати з файлу поля Truck. Перевантажими << , щоб виводити дані у файл. Робота з вводом-виводом інформації.

Клас Order

Перевантажимо операцію присвоєння. Причини – аналогічні.

Теж само для >> і <<. Робота з вводом-виводом інформації.

Клас RideReport

Все те ж саме , окрім того , що в RideReport.hзнаходиться також клас DopClass. Він містить той самий сенс, що і RideReport, але використовується для побітного запису в файл. Через те, що в класі RideReport використані string, в DopClass замінені string на const char .

І перегрузку оператора присвоєння зробимо також для нього.

Клас WeeklyBalance

Він має перегрузку операції індексування , для того, щоб ми мали можливість використовувати коректно масив RideReport [].

Також, в продукті ми можем спостерігати приклад поліморфізму, який виник внаслідок умов завдання(потрібно одночасно мати можливість вивести значення поля в стилому форматі та повністю, що й призвело до появи двох геттерів для одного поля) приклад поліморфізму можемо побачити в RideReport

**string RideReport::getPass()**

**{**

**return this->pass;**

**}**

**string RideReport::getPass(int)**

**{**

**string str1, str2, str;**

**bool ch = 0;**

**int i = 0;**

**while(ch == 0)**

**{**

**if (this->pass[i] != '-' && i < this->pass.size()) str1.push\_back(this->pass[i]);**

**else**

**{**

**ch = 1;**

**}**

**i++;**

**}**

**i = this->pass.size() - 1;**

**while (ch == 1)**

**{**

**if (this->pass[i] != '-' && i >= 0) str2.push\_back(this->pass[i]);**

**else**

**{**

**ch = 0;**

**}**

**i--;**

**}**

**reverse(str2.begin(), str2.end());**

**str += str1;**

**str += "-";**

**str += str2;**

**return str;**

**}**

### Фрагменти коду, для яких треба встановити обробку виключних ситуацій

Тепер розлянемо обробку виключних ситуацій. Для цього використовуємо бібліотеку <exception>, а саме try-catch-throw. Почнемо з обмежень у полях класу Date. Тут все просто: стандартні обмеження для року, місяця та дня.

Далі, розглянемо клас Truck. Зробимо значення < 0 для capacity забороненими. Теж саме для petrolper100km.

Теперь перейдемо до класу Order. Зробимо виключення, щоб вага вантажу не була вищою за вантажопідйомність вантажівки. Також довжина маршруту не може бути < 0.

**if (weidht < 0) throw exception("Weight < 0");**

**if (weidht > truck.getCapacity()) throw exception("Weidht > TruckCapacity");**

**if (routelenght < 0) throw exception("Routelenght < 0");**

У weeklyBalance також є виключення стандартного типу, тобто перевірка отрманого значення на додатність.

**if (size < 0) throw exception("Array size < 0");**

Також є виключення, коли ми вводимо індекс масиву, що виходить за межі виділенної пам’яті.

**throw(exception("Index not in array"));**

### Шаблонна функція

Дана функція приймає два параметри, один з яких – массив шаблонного типу, інший – розмір місиву. Функція рахує середнє значення чисел з масиву.

**template<typename T>**

**T Middle(T\* temp, int n)**

**{**

**T sum = 0;**

**for (int i = 0; i < n; i++)**

**{**

**sum += temp[i];**

**}**

**return sum/n;**

**}**

Функція використовується в інших функціях, які шукають середні значення, відповідно. Перша – середню витрату пального за період, друга – середню довжину шляху за період .

**void MiddlePetrolUs()**

**{**

**system("CLS");**

**double\* mass = new double[A.getSize()];**

**for (int i = 0; i < A.getSize(); i++)**

**{**

**mass[i] = A[i].getUsedFuel();**

**}**

**cout << "Middle use of petrol: " << Middle(mass, A.getSize());**

**BackToMenu();**

**}**

**void MiddleRouteLeng()**

**{**

**system("CLS");**

**int\* mass = new int[A.getSize()];**

**for (int i = 0; i < A.getSize(); i++)**

**{**

**mass[i] = A[i].getRoutelenght();**

**}**

**cout << "Middle routelenght " << Middle(mass, A.getSize());**

**BackToMenu();**

**}**

## Невеликий підсумок до основної частини

В ході виконання основної частини, було створено програму, що оброблює запити користувача через зручне меню. Також дотримались вимог, які були поставлені.

Задля меню зробили окремі файли. Функції теж розмістили в окремий файл для зручності.

Під час створення ієрархії дотримувались загальноприйнятої логіки.

Обробку виключних ситуацій зробили за допомогою бібліотеки <exception>.

Взаємодія з програмою знаходиться далі , у Додатку2.

# Висновки

## Підсумки з освітлених питань , шляхи вирішення аспектів проблеми теми

Під час написання курсової роботи, ми ретельно закріпили пройдений матеріал задля покращення професійного рівня володіння мовою С++ , а також взагалі задля загального сприйняття ООП, не стосовно конкретної мови, а абстрактно.

Під час виконання завдання курсової роботи ми створили ієрархічну модель класів для опису об’єктів реального світу, використовуючи інструментальні можливості мови C++.

Якщо посилатись на завдання дослівно, реалізовано наступне:

1. успадковування;

2. агрегація;

3. закритість внутрішніх полів класів (доступ лише за допомогою відкритих методів класу, що реалізують обмеження полів класу реальними значення);

4. методи змінення (обмеження можливих значень полів класу, обмеження на день, місяць і рік не зараховуються, як найбільш тривіальні);

5. визначення перевантажених операторів для об’єктів, створених в програмному продукті, (перевантаження операторів повинно мати ясний і доцільний зміст для використання);

6. застосування визначених перевантажених операторів;

7. визначення перевантаженого оператора індексування для внутрішнього динамічного масиву, для якого слід виділити мінімальну потрібну пам’ять;

8. застосування перевантаженого оператора індексування;

9. обробка виключних ситуацій;

10. застосування виключень, визначених як ієрархія класів;

11. визначення шаблону функції для підрахунку статистичної характеристики;

12. використання шаблону функції;

13. меню;

14. виведення даних у вигляді таблиці;

15. запис даних у текстовий файл;

16. зчитування даних з текстового файлу;

17. запис даних у бінарний файл;

18. зчитування даних з бінарного файлу;

19. пошук інформації за числовим полем;

20. пошук інформації за текстовим полем.

Також в ході написання записки до курсової роботи було дано змістовне пояснення вибору наступних елементів програми (посилаємось на завдання дослівно):

1. взаємозв'язки між класами (які зв’язки між класами обумовили вибір агрегації, або успадкування в ієрархії класів);
2. вибір конкретних значень обмежень для полів різних класів, щоб деяким полям класів не можна було присвоїти нереальні значення (обмеження на нереальні значення днів та місяців дат не зараховуються);
3. вибір певних перевантажених операцій для об’єктів класів та доцільність і зручність їх застосування в програмному коді;
4. вибір певних фрагментів програмного коду, для яких слід встановити обробку виключних ситуацій;
5. обґрунтування при оголошенні шаблону функції визначення певних параметрів шаблону функції в якості параметрів-типів, чи в якості параметрів-констант.

Також було реалізоване меню для взаємодії користувача з програмою – можна сказати, так званий API. Меню вирішено було зробити не простим контекстним, а більш зручним, з можливістю повернення до його початку, тощо.

Також реалізовано було взаєможію з бінарним (двійковим) і текстовим файлами. Це було виконано за допомогою відповідно пооб’єктного запису, зчитування і перевантаження операторів потокового вводу – виводу.

На мою думка завдання було виконане в повному обсязі і не вимагає довнень.

## Рекомендації

Хочеться відмітити готовність приймаючої роботу сторони надати допомогу при виникнених питаннях. Також хочеться відмітити допомогу викладача з практики.

Єдине, що – хотілося б трохи зпростити усі умови до роботи, умови до стилізації й оформлення. Все видалось якимось незручним, в різних місцях, незрозуміло структурованим. Хотілося б, щоб усе було лаконічно, мінімум того, що не потрібно, а більше схоже на ТЗ на реальній роботі, де тобі говорять по пунктах 1, 2, 3. А окрім цих пунктів роблять одну примітку з тим, що не стосується цих пунктів.

Наприклад:

1. Зробити ієрархію. Ієрархія для вас наступна :

…

Треба, щоб в класах ієрархії було :

1)

2)

3)

4)

В цьому ж документі наприкінці задати строки здачі

Зробити окремо документ для оформлення

Так було б, на мій погляд, трохи краще.

Щодо реалізації завдання – рекомендацій немає.

# Список використаних джерел

Визначення мови програмування високого рівня на dictionary.com(англ.)

Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс, 2008.- 720с.

 Armstrong, «The Quarks of Object-Oriented Development.» In descending order of popularity, the «quarks» are: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, Abstraction

Г. Буч, Д. Рамбо, И. Якобсон. Язык UML. Руководство пользователя = The Unified Modeling Language Usere Guide. — 2-е. — М. : ДМК Пресс, 2006. — 496 с. — ISBN 5-94074-334-X.

# Додатки

## Додаток 1

Специфікація

Microsoft Visual Studio Community 2019

Версия 16.7.3

VisualStudio.16.Release/16.7.3+30503.244

Microsoft .NET Framework

Версия 4.8.03752

Установленная версия: Community

Visual C++ 2019 00435-60000-00000-AA867

Microsoft Visual C++ 2019

ASP.NET and Web Tools 2019 16.7.532.28833

ASP.NET and Web Tools 2019

Microsoft JVM Debugger 1.0

Provides support for connecting the Visual Studio debugger to JDWP compatible Java Virtual Machines

Microsoft MI-Based Debugger 1.0

Provides support for connecting Visual Studio to MI compatible debuggers

ProjectServicesPackage Extension 1.0

ProjectServicesPackage Visual Studio Extension Detailed Info

Test Adapter for Boost.Test 1.0

Enables Visual Studio's testing tools with unit tests written for Boost.Test. The use terms and Third Party Notices are available in the extension installation directory.

Visual F# Tools 10.10.0.0 для F# 4.7 16.7.0-beta.20361.3+3ef6f0b514198c0bfa6c2c09fefe41a740b024d5

Microsoft Visual F# Tools 10.10.0.0 для F# 4.7

Адаптер теста для Google Test 1.0

Включает средства тестирования Visual Studio и модульные тесты для Google Test. Условия использования и уведомления сторонних производителей см. в каталоге установки расширения.

Диспетчер пакетов NuGet 5.7.0

Диспетчер пакетов NuGet в Visual Studio. Дополнительные сведения о NuGet см. на веб-сайте https://docs.nuget.org/

Инструменты C# 3.7.0-6.20427.1+18ede13943b0bfae1b44ef078b2f3923159bcd32

Компоненты C#, используемые в интегрированной среде разработки. В зависимости от типа и настроек проекта могут использоваться различные версии компилятора.

Инструменты Visual Basic 3.7.0-6.20427.1+18ede13943b0bfae1b44ef078b2f3923159bcd32

Компоненты Visual Basic, используемые в интегрированной среде разработки. В зависимости от типа и настроек проекта могут использоваться различные версии компилятора.

Инструменты Visual Studio для CMake 1.0

Инструменты Visual Studio для CMake

Мастера Microsoft Visual C++ 1.0

Мастера Microsoft Visual C++

Общие инструменты Windows Azure 1.10

Предоставляет общие службы для использования мобильными службами и инструментами Microsoft Azure.

Пакет VC Microsoft Visual Studio 1.0

Пакет VC Microsoft Visual Studio

Пакет размещения адаптеров отладки Visual Studio Code 1.0

Уровень взаимодействия для размещения адаптеров отладки Visual Studio Code в Visual Studio

Расширение IntelliCode 1.0

Подробные сведения о расширении IntelliCode для Visual Studio

Средства Службы приложений Azure, версия 3.0.0 16.7.532.28833

Средства Службы приложений Azure, версия 3.0.0

## Додаток 2

Увесь код міститься в файлі з кодом. Тут знаходяться значиміші блоки коду. Main та ті , що стосуються ієрархії. Перший – файл main.cpp , потім main.h Докладні коментарі допоможуть розібратися , що виконується в функції main().

**#include <windows.h>**

**#include <iostream>**

**#include <clocale>**

**#include <ostream>**

**#include <istream>**

**#include "WeeklyBalance.h"**

**#include "Functions.h"**

**#include "mainmenu.h"**

**#include "main.h"**

**WeeklyBalance A;**

**using namespace std;**

**extern HANDLE hStdOut;**

**extern CONSOLE\_SCREEN\_BUFFER\_INFO csbInfo;**

**extern SMALL\_RECT consolRect;**

**extern WORD woкkWindowAttributes;**

**extern EnumMenuItemsMain current\_MI;**

**int main()**

**{**

**try**

**{**

**setlocale(LC\_CTYPE, "rus");**

**hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);**

**consolRect = csbInfo.srWindow;**

**system("CLS");**

**DrawMenu();**

**}**

**catch (const exception & ex)**

**{**

**cout << ex.what() << endl;**

**}**

**}**

**void Exit()//Функція відповідає за вихід з меню**

**{**

**exit(0);**

**}**

**void Add()//Відповідає за позицію з додаванням новго елементу**

**{**

**system("CLS");**

**push(A);//Lviv 120 11 11 2020 1000 100 500 20 11 2020 Moskov-Lviv**

**BackToMenu();**

**}**

**void ToScreen()//Відповідає за позицію з виведенням на екран**

**{**

**system("CLS");**

**toScreen(A);**

**BackToMenu();**

**}**

**void ToBinaryFile()//Відповідає за позицію з виводом в бінарний файл**

**{**

**system("CLS");**

**string temp;**

**cout << "Please enter filename" << endl;**

**cin >> temp;**

**toBinaryF(A, temp);**

**BackToMenu();**

**}**

**void ToWordFile()//Відповідає за позицію з виводом в текстовий файл**

**{**

**system("CLS");**

**string temp;**

**cout << "Please enter filename" << endl;**

**cin >> temp;**

**toWordF(A, temp);**

**BackToMenu();**

**}**

**void FromBinaryFile()//Відповідає за позицію з зчитуванням в бінарного файлу**

**{**

**system("CLS");**

**string temp;**

**cout << "Please enter filename" << endl;**

**cin >> temp;**

**fromBinaryF(A, temp);**

**BackToMenu();**

**}**

**void FromWordFile()//Відповідає за позицію з зчитуванням в текстового файлу**

**{**

**system("CLS");**

**string temp;**

**cout << "Please enter filename" << endl;**

**cin >> temp;**

**fromWordF(A, temp);**

**BackToMenu();**

**}**

**void SearchByStr()//Відповідає за позицію з пошуком по введеній строці**

**{**

**system("CLS");**

**string temp;**

**cout << "Please enter num for search" << endl;**

**cin >> temp;**

**searchByS(A, temp);**

**BackToMenu();**

**}**

**void SearchByNum()//Відповідає за позицію з пошуком по введеному номеру**

**{**

**system("CLS");**

**double temp;**

**cout << "Please enter num for search" << endl;**

**cin >> temp;**

**system("CLS");**

**searchByN(A, temp);**

**BackToMenu();**

**}**

**//Шаблон функції для знаходження середнього значення масиву данних**

**template<typename T>**

**T Middle(T\* temp, int n)**

**{**

**T sum = 0;**

**for (int i = 0; i < n; i++)**

**{**

**sum += temp[i];**

**}**

**return sum/n;**

**}**

**void MiddlePetrolUs()//Відповідає за позицію з пошуком середніх витрат пального**

**{**

**system("CLS");**

**double\* mass = new double[A.getSize()];**

**for (int i = 0; i < A.getSize(); i++)**

**{**

**mass[i] = A[i].getUsedFuel();**

**}**

**cout << "Middle use of petrol: " << Middle(mass, A.getSize());**

**BackToMenu();**

**}**

**void MiddleRouteLeng()//Відповідає за позицію з пошуком середньої довжини маршруту**

**{**

**system("CLS");**

**int\* mass = new int[A.getSize()];**

**for (int i = 0; i < A.getSize(); i++)**

**{**

**mass[i] = A[i].getRoutelenght();**

**}**

**cout << "Middle routelenght " << Middle(mass, A.getSize());**

**BackToMenu();**

**}**

**#pragma once**

**//Функції привязані до меню, які викликають виконуючі функції**

**void Exit();**

**void Add();**

**void ToScreen();**

**void ToBinaryFile();**

**void ToWordFile();**

**void FromBinaryFile();**

**void FromWordFile();**

**void SearchByStr();**

**void SearchByNum();**

**void MiddlePetrolUs();**

**void MiddleRouteLeng();**

**#include "Date.h"**

**Date::Date()//Конструктор за замовчуванням**

**{**

**this->day = 1;**

**this->month = 1;**

**this->year = 2000;**

**}**

**Date::Date(int d, int m, int y)//Конструктор відповідно до переданим данним**

**{**

**//Обробка правильності введення інформації**

**if (d < 0) throw exception("Day < 0");**

**if (d > 31) throw exception("Day > 31");**

**if (m < 0) throw exception("Month < 0");**

**if (m > 12) throw exception("Month > 12");**

**if (y < 0) throw exception("Year < 0");**

**day = d;**

**month = m;**

**year = y;**

**}**

**Date::Date(const Date& from)//Конструктор копії**

**{**

**day = from.day;**

**month = from.month;**

**year = from.year;**

**}**

**//Сеттери для полів**

**Date& Date::setDay(int d)**

**{**

**//Обробка правильності введення інформації**

**if (d < 0) throw exception("Day < 0");**

**if (d > 31) throw exception("Day > 31");**

**day = d;**

**return \*this;**

**}**

**Date& Date::setMonth(int m)**

**{**

**//Обробка правильності введення інформації**

**if (m < 0) throw exception("Month < 0");**

**if (m > 12) throw exception("Month > 12");**

**month = m;**

**return \*this;**

**}**

**Date& Date::setYear(int y)**

**{**

**//Обробка правильності введення інформації**

**if (y < 0) throw exception("Year < 0");**

**year = y;**

**return \*this;**

**}**

**//Геттери для полів**

**int Date::getDay()**

**{**

**return day;**

**}**

**int Date::getMonth()**

**{**

**return month;**

**}**

**int Date::getYear()**

**{**

**return year;**

**}**

**//Перевантжений оператор присвоювання**

**Date& Date::operator=(Date other)**

**{**

**this->day = other.day; this->month = other.month; this->year = other.year; return \*this;**

**}**

**//Перевантаження оператору потокового вводу**

**istream& operator >> (istream& in, Date& Date) {**

**//Отримання інформації через екземпляр istream**

**in >> Date.day >> Date.month >> Date.year;**

**//Обробка правильності введення інформації**

**if (Date.day < 0) throw exception("Day < 0");**

**if (Date.day > 31) throw exception("Day > 31");**

**if (Date.month < 0) throw exception("Month < 0");**

**if (Date.month > 12) throw exception("Month > 12");**

**if (Date.year < 0) throw exception("Year < 0");**

**return in;**

**}**

**//Перевантаження оператору потокового виведення**

**ostream& operator << (ostream& out, Date& Date) {**

**//Отримання інформації через екземпляр istream**

**out << Date.day << " " << Date.month << " " << Date.year << " ";**

**return out;**

**}**

**#pragma once**

**#include <iostream>**

**#include <exception>**

**using namespace std;**

**class Date {**

**int day, month, year;**

**public:**

**Date();//Конструктор за замовчуванням**

**Date(int d, int m, int y);//Конструктор відповідно до переданим данним**

**Date(const Date& from);//Конструктор копії**

**//Сеттери для полів**

**Date& setDay(int d);**

**Date& setMonth(int m);**

**Date& setYear(int y);**

**//Геттери для полів**

**int getDay();**

**int getMonth();**

**int getYear();**

**//Перевантжений оператор присвоювання**

**Date& operator = (Date other);**

**//Підключення операторів потокового вводу та виводу інфорації**

**friend ostream& operator << (ostream& out, Date& outstream);**

**friend istream& operator >> (istream& in, Date& instream);**

**};**

**#include "Order.h"**

**Order::Order():departure(), truck()//Конструктор за замовчуванням**

**{**

**this->destination = " ";**

**this->weidht = 0;**

**this->routelenght = 0;**

**this->cost = 0;**

**}**

**Order::Order(string destination, double weidht, int routelenght, Date departure, Truck truck):departure(departure), truck(truck)//Конструктор відповідно до переданим данним**

**{**

**//Обробка неправильно введених данних**

**if (weidht < 0) throw exception("Weidht < 0");**

**if (weidht > truck.getCapacity()) throw exception("Weidht > TruckCapacity");**

**if (routelenght < 0) throw exception("Routelenght < 0");**

**this->destination = destination;**

**this->weidht = weidht;**

**this->routelenght = routelenght;**

**this->cost = (truck.getPetrolper100km() \* routelenght \* 28 + 5000);**

**}**

**Order::Order(const Order& from):departure(from.departure), truck(from.truck)//Конструктор копії**

**{**

**this->destination = from.destination;**

**this->weidht = from.weidht;**

**this->routelenght = from.routelenght;**

**this->cost = from.cost;**

**}**

**Order& Order::setCost()//Оператор для внутрішньої модифікації поля cost**

**{**

**this->cost = (this->truck.getPetrolper100km() \* this->routelenght \* 28 + 5000);;**

**return \*this;**

**}**

**Order& Order::setDestination(string temp)**

**{**

**this->destination = temp;**

**return \*this;**

**}**

**Order& Order::setWeidht(double temp)**

**{**

**//Обробка неправильно введених данних**

**if (temp < 0) throw exception("Weidht < 0");**

**if (temp > truck.getCapacity()) throw exception("Weidht > TruckCapacity");**

**weidht = temp;**

**return \*this;**

**}**

**Order& Order::setRoutelenght(int temp)**

**{**

**//Обробка неправильно введених данних**

**if (temp < 0) throw exception("Routelenght < 0");**

**this->cost = (truck.getPetrolper100km() \* routelenght \* 28 + weidht \* 10);**

**routelenght = temp;**

**return \*this;**

**}**

**Order& Order::setDeparture(Date temp)**

**{**

**departure = temp;**

**return \*this;**

**}**

**Order& Order::setTruck(Truck temp)**

**{**

**truck = temp;**

**this->cost = (truck.getPetrolper100km() \* routelenght \* 28 + weidht \* 10);**

**return \*this;**

**}**

**//Геттери для полів**

**string Order::getDestination()**

**{**

**return this->destination;**

**}**

**double Order::getWeidht()**

**{**

**return this->weidht;**

**}**

**double Order::getCost()**

**{**

**return this->cost;**

**}**

**int Order::getRoutelenght()**

**{**

**return this->routelenght;**

**}**

**Date Order::getDeparture()**

**{**

**return this->departure;**

**}**

**Truck Order::getTruck()**

**{**

**return this->truck;**

**}**

**//Перевантжений оператор присвоювання**

**Order& Order::operator=(Order from)**

**{**

**this->destination = from.destination;**

**this->weidht = from.weidht;**

**this->routelenght = from.routelenght;**

**this->cost = from.cost;**

**this->truck = from.truck;**

**this->departure = from.departure;**

**return \*this;**

**}**

**//Перевантаження оператору потокового виведення**

**ostream& operator<<(ostream& out, Order& Order)**

**{**

**//Отримання інформації через екземпляр istream**

**out << Order.destination << " " << Order.routelenght << " " << Order.departure << " " << Order.truck << " " << Order.weidht;**

**return out;**

**}**

**//Перевантаження оператору потокового вводу**

**istream& operator>>(istream& in, Order& Order)**

**{**

**//Отримання інформації через екземпляр istream**

**in >> Order.destination >> Order.routelenght >> Order.departure >> Order.truck >> Order.weidht;**

**if (Order.routelenght < 0) throw exception("Routelenght < 0");**

**if (Order.weidht < 0) throw exception("Weidht < 0");**

**Order.setCost();**

**return in;**

**}**

**#pragma once**

**#include <string>**

**#include <iostream>**

**#include <ostream>**

**#include <exception>**

**#include "Date.h"**

**#include "Truck.h"**

**using namespace std;**

**class Order**

**{**

**string destination;**

**double weidht, cost;**

**int routelenght;**

**Date departure;**

**Truck truck;**

**public:**

**Order();//Конструктор за замовчуванням**

**Order(string destination, double weidht, int routelenght, Date departure, Truck truck);//Конструктор відповідно до переданим данним**

**Order(const Order& from);//Конструктор копії**

**//Сеттери для полів**

**Order& setCost();**

**Order& setDestination(string temp);**

**Order& setWeidht(double temp);**

**Order& setRoutelenght(int temp);**

**Order& setDeparture(Date temp);**

**Order& setTruck(Truck temp);**

**//Геттери для полів**

**string getDestination();**

**double getWeidht();**

**double getCost();**

**int getRoutelenght();**

**Date getDeparture();**

**Truck getTruck();**

**//Перевантжений оператор присвоювання**

**Order& operator = (Order other);**

**//Підключення операторів потокового вводу та виводу інфорації**

**friend ostream& operator << (ostream& out, Order& Order);**

**friend istream& operator >> (istream& in, Order& Order);**

**};**

**#include "RideReport.h"**

**RideReport::RideReport() :executiondate()//Конструктор за замовчуванням**

**{**

**this->usedfuel = 0;**

**this->pass = " ";**

**}**

**RideReport::RideReport(string pass, Date executiondate, string destination, double weidht, int routelenght, Date departure, Truck truck):executiondate(executiondate),Order(destination, weidht, routelenght, departure, truck)//Конструктор відповідно до переданим данним**

**{**

**this->usedfuel = this->getRoutelenght() \* this->getTruck().getPetrolper100km();**

**this->pass = pass;**

**}**

**RideReport::RideReport(const RideReport& other) : Order((Order&)other)//Конструктор копії**

**{**

**this->executiondate = other.executiondate;**

**this->pass = other.pass;**

**this->setUsedFuel();**

**}**

**//Геттери для полів**

**string RideReport::getPass()**

**{**

**return this->pass;**

**}**

**string RideReport::getPass(int)**

**{**

**string str1, str2, str;**

**bool ch = 0;**

**int i = 0;**

**while(ch == 0)//Прохід до завершення першого слова**

**{**

**if (this->pass[i] != '-' && i < this->pass.size()) str1.push\_back(this->pass[i]);**

**else**

**{**

**ch = 1;**

**}**

**i++;**

**}**

**i = this->pass.size() - 1;**

**while (ch == 1)//Прохід з кінця до початку останнього слова**

**{**

**if (this->pass[i] != '-' && i >= 0) str2.push\_back(this->pass[i]);**

**else**

**{**

**ch = 0;**

**}**

**i--;**

**}**

**//Конкатинація строк**

**reverse(str2.begin(), str2.end());**

**str += str1;**

**str += "-";**

**str += str2;**

**return str;**

**}**

**Date RideReport::getExecutiondate()**

**{**

**return this->executiondate;**

**}**

**double RideReport::getUsedFuel()**

**{**

**return usedfuel;**

**}**

**//Сеттери для полів**

**RideReport& RideReport::setPass(string temp)**

**{**

**this->pass = temp;**

**return \*this;**

**}**

**RideReport& RideReport::setExecutiondate(Date temp)**

**{**

**this->executiondate = temp;**

**return \*this;**

**}**

**RideReport& RideReport::setUsedFuel()//Метод для оновлення данних про викорастане паливо**

**{**

**this->usedfuel = this->getRoutelenght() \* this->getTruck().getPetrolper100km() / 100;**

**return \*this;**

**}**

**//Перевантжений оператор присвоювання для роботи з додатковою структорою данних**

**void RideReport::operator=(DopClass other)**

**{**

**this->setDestination(other.destination);**

**this->setRoutelenght(other.routelenght);**

**Truck truck(other.capacity, other.petrolper100km);**

**this->setTruck(truck);**

**this->setWeidht(other.weidht);**

**Date data(other.dayo, other.montho, other.yearo), data1(other.dayr, other.monthr, other.yearr);**

**this->setDeparture(data);**

**this->executiondate = data1;**

**this->pass = other.pass;**

**this->setUsedFuel();**

**this->setCost();**

**}**

**//Оператори потокового вводу і виводу**

**ostream& operator<<(ostream& out, RideReport& RideReport)**

**{**

**//Виведення інформації через екземпляр ostream**

**Date d;**

**d = RideReport.getDeparture();**

**Truck t;**

**t = RideReport.getTruck();**

**out << RideReport.getDestination() << " " << RideReport.getRoutelenght() << " " << d << t << RideReport.getWeidht() << " " << RideReport.executiondate << RideReport.getPass();**

**return out;**

**}**

**istream& operator>>(istream& in, RideReport& RideReport)**

**{**

**//Отримання інформації через екземпляр istream**

**Order OR;**

**in >> OR >> RideReport.executiondate >> RideReport.pass;**

**RideReport.setTruck(OR.getTruck());**

**RideReport.setWeidht(OR.getWeidht());**

**RideReport.setDestination(OR.getDestination());**

**RideReport.setRoutelenght(OR.getRoutelenght());**

**RideReport.setDeparture(OR.getDeparture());**

**RideReport.getUsedFuel();**

**return in;**

**}**

**//Перевантжений оператор присвоювання для коректного переносу данних з допоміжного в основний клас**

**void DopClass::operator=(RideReport other)**

**{**

**strcpy(this->destination, other.getDestination().c\_str());**

**this->routelenght = other.getRoutelenght();**

**this->capacity = other.getTruck().getCapacity();**

**this->petrolper100km = other.getTruck().getPetrolper100km();**

**this->weidht = other.getWeidht();**

**this->dayo = other.getDeparture().getDay();**

**this->montho = other.getDeparture().getMonth();**

**this->yearo = other.getDeparture().getYear();**

**strcpy(this->pass, other.getPass().c\_str());**

**this->dayr = other.getExecutiondate().getDay();**

**this->monthr = other.getExecutiondate().getMonth();**

**this->yearr = other.getExecutiondate().getYear();**

**}**

**#pragma once**

**#include <string>**

**#include <iostream>**

**#include <ostream>**

**#include <exception>**

**#include "Date.h"**

**#include "Order.h"**

**#include <algorithm>**

**using namespace std;**

**class DopClass;**

**class RideReport: public Order**

**{**

**string pass;**

**Date executiondate;//включення до класу власного класу дати**

**double usedfuel;**

**public:**

**RideReport();//Конструктор за замовчуванням**

**RideReport(string pass, Date executiondate, string destination, double weidht, int routelenght, Date departure, Truck truck);//Конструктор відповідно до переданим данним**

**RideReport(const RideReport& other);//Конструктор копії**

**//Геттери для полів**

**string getPass();//Перевантаження повного виведення поля**

**string getPass(int);//Перевантаження частокового виведення поля виведення поля**

**Date getExecutiondate();**

**double getUsedFuel();**

**//Сеттери для полів**

**RideReport& setPass(string temp);**

**RideReport& setExecutiondate(Date temp);**

**RideReport& setUsedFuel();**

**//Перевантжений оператор присвоювання для роботи з додатковою структорою данних**

**void operator = (DopClass other);**

**//Підключення операторів потокового вводу та виводу інфорації**

**friend ostream& operator << (ostream& out, RideReport& RideReport);**

**friend istream& operator >> (istream& in, RideReport& RideReport);**

**};**

**class DopClass//Додаткова структура данних для роботи з файлами**

**{**

**public:**

**char destination[100];**

**double weidht;**

**int routelenght;**

**double capacity;**

**double petrolper100km;**

**int dayo;**

**int montho;**

**int yearo;**

**char pass[100];**

**int dayr;**

**int monthr;**

**int yearr;**

**//Перевантжений оператор присвоювання для коректного переносу данних з допоміжного в основний клас**

**void operator=(RideReport other);**

**};**

**#include "Truck.h"**

**Truck::Truck()//Конструктор за замовчуванням**

**{**

**this->capacity = 0;**

**this->petrolper100km = 0;**

**}**

**Truck::Truck(double cap, double pet)//Конструктор відповідно до переданим данним**

**{**

**//Перевірка на позитивність внесених данних**

**if (cap < 0) throw exception("Capacity < 0");**

**if (pet < 0) throw exception("Petrolper100km < 0");**

**capacity = cap;**

**petrolper100km = pet;**

**}**

**Truck::Truck(const Truck& from)//Конструктор копії**

**{**

**capacity = from.capacity;**

**petrolper100km = from.petrolper100km;**

**}**

**//Сеттери**

**Truck& Truck::setCapacity(double temp)**

**{**

**//Перевірка на позитивність внесених данних**

**if (temp < 0) throw exception("Capacity < 0");**

**capacity = temp;**

**return \*this;**

**}**

**Truck& Truck::setPetrolper100km(double temp)**

**{**

**//Перевірка на позитивність внесених данних**

**if (temp < 0) throw exception("Petrolper100km < 0");**

**petrolper100km = temp;**

**return \*this;**

**}**

**//Геттери**

**double Truck::getCapacity()**

**{**

**return capacity;**

**}**

**double Truck::getPetrolper100km()**

**{**

**return petrolper100km;**

**}**

**//Перевантжений оператор присвоювання**

**Truck& Truck::operator=(Truck other)**

**{**

**this->capacity = other.capacity;**

**this->petrolper100km = other.petrolper100km;**

**return \*this;**

**}**

**Truck::~Truck()//Деструктор**

**{**

**capacity = petrolper100km = 0;**

**}**

**//Оператори потокового вводу і виводу**

**istream& operator >> (istream& in, Truck& Truck) {**

**//Отримання інформації через екземпляр istream**

**in >> Truck.capacity >> Truck.petrolper100km;**

**return in;**

**}**

**ostream& operator << (ostream& out, Truck& Truck) {**

**//Виведення інформації через екземпляр ostream**

**out << Truck.capacity << " " << Truck.petrolper100km << " ";**

**if (Truck.capacity < 0) throw exception("Capacity < 0");**

**if (Truck.petrolper100km < 0) throw exception("Petrolper100km < 0");**

**return out;**

**}**

**#pragma once**

**#include <ostream>**

**#include <iostream>**

**#include <exception>**

**using namespace std;**

**class Truck**

**{**

**double capacity, petrolper100km;**

**public:**

**Truck();//Конструктор за замовчуванням**

**Truck(double cap, double pet);//Конструктор відповідно до переданим данним**

**Truck(const Truck& from);//Конструктор копії**

**//Сеттери для полів**

**Truck& setCapacity(double temp);**

**Truck& setPetrolper100km(double temp);**

**//Геттери для полів**

**double getCapacity();**

**double getPetrolper100km();**

**//Перевантжений оператор присвоювання**

**Truck& operator = (Truck other);**

**//Підключення операторів потокового вводу та виводу інфорації**

**friend ostream& operator << (ostream& out, Truck& Truck);**

**friend istream& operator >> (istream& in, Truck& Truck);**

**~Truck();//Деструктор**

**};**

**#include "WeeklyBalance.h"**

**WeeklyBalance::WeeklyBalance() //Конструктор за замовчуванням**

**{**

**this->size = 0;**

**list = new RideReport[0];**

**}**

**WeeklyBalance::WeeklyBalance(int size)//Конструктор, який стоврює кільість екзамплярів в масиві відповідно переданим данним**

**{**

**if (size < 0) throw exception("Array size < 0");//Перевірка чи задане число більше 0**

**this->size = size;**

**list = new RideReport[size];//Динамічне виілення памяті**

**for (int i = 0; i < this->size; i++)**

**{**

**cout << "Enter " << i + 1 << " element of list: ";**

**cin >> list[i];**

**}**

**}**

**WeeklyBalance::WeeklyBalance(const WeeklyBalance& other)**

**{**

**this->size = other.size;**

**list = new RideReport[size];**

**for (int i = 0; i < this->size; i++)**

**{**

**list[i] = other.list[i];**

**}**

**}//Конструктор копії**

**WeeklyBalance::~WeeklyBalance()**

**{**

**delete[] list;**

**}//Деструктор**

**int WeeklyBalance::getSize()**

**{**

**return this->size;**

**}//Геттер для отримання розміру масива**

**RideReport& WeeklyBalance::operator[](int value)//Оператор індексації, для отримання доступу для елементів масиву list**

**{**

**{**

**if ((value < size) && (value >= 0)) {**

**return this->list[value];**

**}**

**else**

**throw(exception("Index not in array"));//Перевірка на наявність індексу в масиві**

**}**

**}**

**void WeeklyBalance::inputmask()//Метод, який використовується задля створення маски вводу інформації**

**{**

**cout << "Destination" << " " << "Routelenght" << " " << "Departure(day mouth year)" << " " << "Truck(capacity petrolper100km)" << " " << "Weidht" << " " << "Pass(towns split by -)" << " " << "Executiondate(day mouth year)";**

**}**

**void WeeklyBalance::push\_back(RideReport other)//Метод, який дозволяє додати один елемент в масив list**

**{**

**RideReport\* list1;//Створення нового масива**

**list1 = new RideReport[size];**

**for (int i = 0; i < size; i++)//Перезапис старого в новий**

**{**

**list1[i] = list[i];**

**}**

**delete[] list;//Очистка старого**

**size++;**

**list = new RideReport[size];//Динамічне виділення місця**

**for (int i = 0; i < size-1; i++)//Переписування маисвів**

**{**

**list[i] = list1[i];**

**}**

**list[size - 1] = other;**

**delete[] list1;//Очистка додаткового**

**}**

**#pragma once**

**#include <string>**

**#include <iostream>**

**#include <ostream>**

**#include <exception>**

**#include "RideReport.h"**

**class RideReport;**

**class WeeklyBalance**

**{**

**int size; //Розмірність масиву**

**RideReport\* list;//Масив елементів RideReport**

**public:**

**WeeklyBalance();//Конструктор за замовчуванням**

**WeeklyBalance(int size);//Конструктор, який стоврює кільість екзамплярів в масиві відповідно переданим данним**

**WeeklyBalance(const WeeklyBalance& other);//Конструктор копії**

**~WeeklyBalance();//Деструктор**

**int getSize();//Геттер для отримання розміру масива**

**RideReport& operator[](int value);//Оператор індексації, для отримання доступу для елементів масиву list**

**void inputmask();//Метод, який використовується задля створення маски вводу інформації**

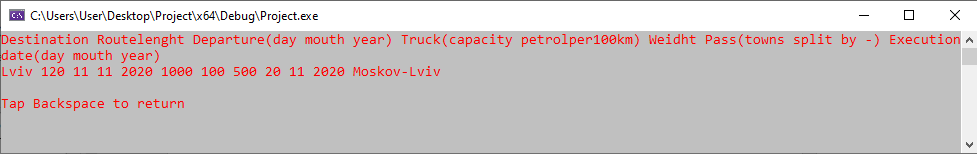
**void push\_back(RideReport other);//Метод, який дозволяє додати один елемент в масив list**

**};**

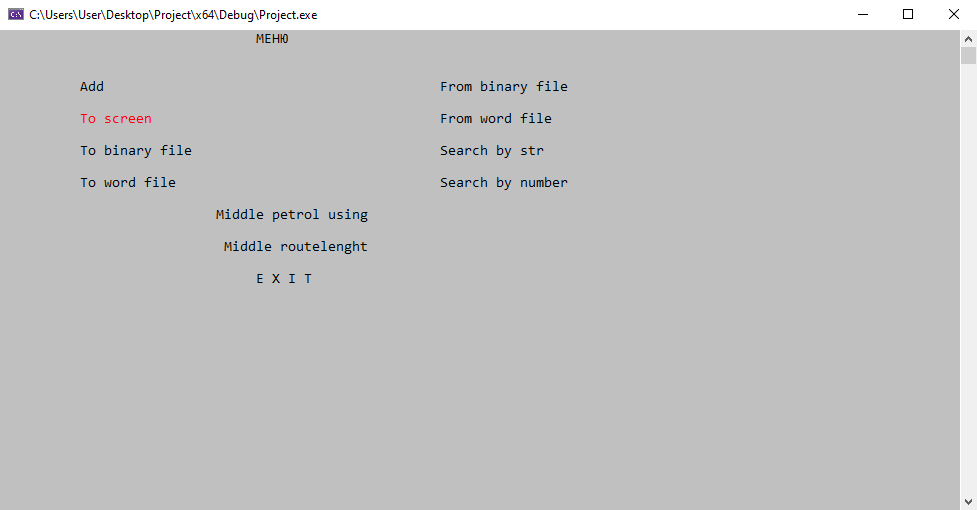
## Додаток 3

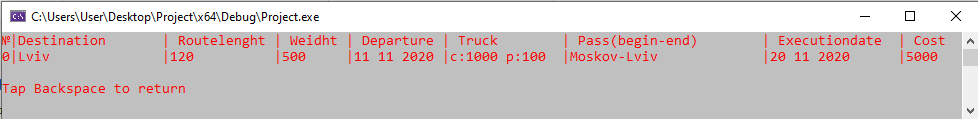
Опис програми.



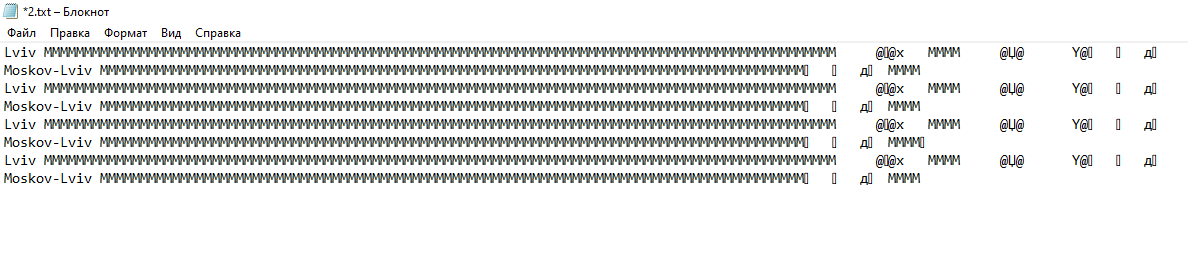


Ввели дані, натискнули Enter, зміни збереглись, повертаємось у головне меню

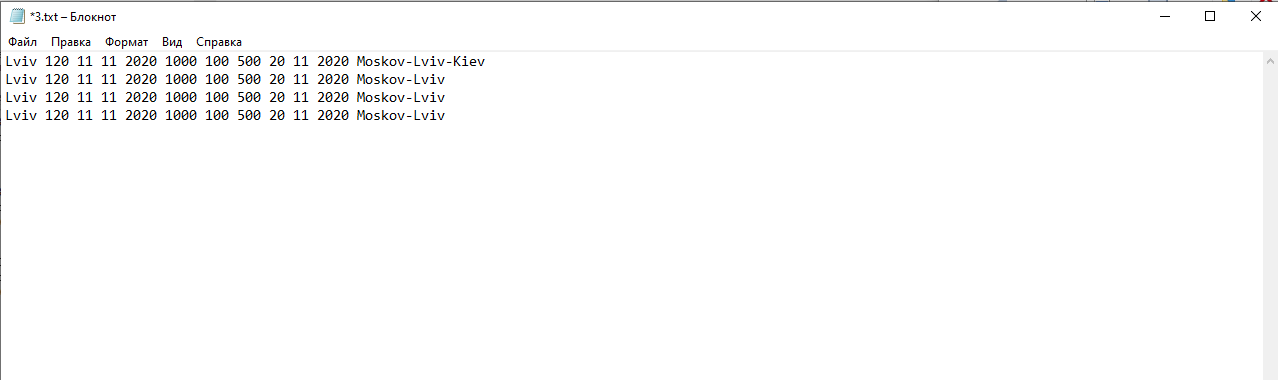




Вивели дані на екран

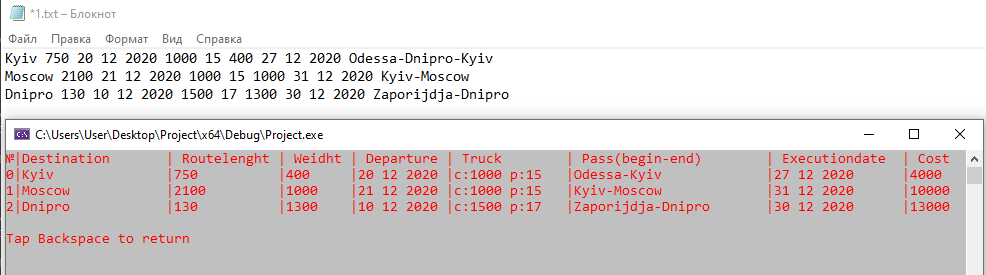


Так виглядає вивід у двійковий файл ( тут результат декількох тестових виводів )

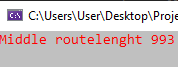


Так виглядає вивід у текстовий файл ( тут результат декількох тестових виводів )

Давайте додамо нову інформацію, щоб наочніше було обчислення MiddlePetrolUsing MirrdleRouteLength тощо.

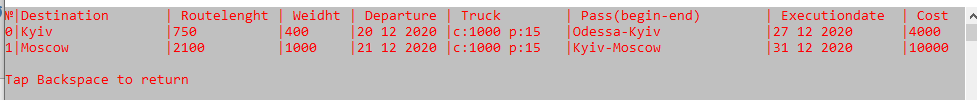




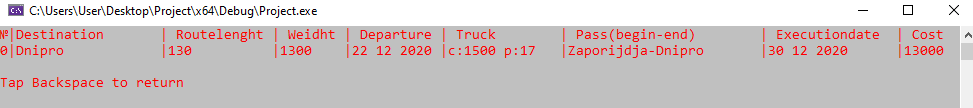




Ось що виводить на запит пошуку по номеру.



Пошук str – ввели Dnipro, результат такий :



До речі, можемо зчитати інформація з текстового файлу. Для цього треба додати цифру на початку, щоб програма знала скільки рядків файлу зчитувати.

