**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 - 1»**

**тема:**

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

|  |  |
| --- | --- |
| **Керівник – Карпенко С.Г.** | **Виконав Андрєєв Д .М.** |
| **Допущено до захисту** | **Студент 2-го курсу** |
| **\_\_ \_\_\_\_\_\_\_\_ 2019 р.** | **Групи ТІ-82** |
| **Захищено з оцінкою** | **залікова книжка № ТІ-81131** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

***2019***

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 - 1»

(назва дисципліни)

на тему:

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

Студента групи ТІ-82

спеціальність **121 «Інженерія програмного забезпечення»**

спеціалізація «**Програмне забезпечення кібер-енергетичних систем»**

Андрєєва Д. М.

(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 - 2019 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 ***121 «Інженерія програмного забезпечення»***

(шифр і назва)

спеціалізація *«****Програмне забезпечення кібер-енергетичних систем»***

(шифр і назва)

ЗАВДАННЯ

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Андрєєву Данилу Максимовичу

(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 – 24 грудня 2019 р.

3. Вихідні 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 успадковування, інкапсуляцію, перевантажені операції, шаблони функцій, запис у двійкові файли та обробку виключень.

5. Перелік графічного матеріалу –

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

6. Дата видачі завдання – 24 вересня 2019 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсової роботи | Строк виконання етапів роботи | Примітка |
|  | Побудова ієрархічної моделі класів | 15.10.2019 |  |
|  | Розробка та застосування інструментальних засобів обробки даних | 19.11.2019 |  |
|  | Написання розрахунково-пояснювальної записки | 24.12.2019 |  |

**Студент \_\_\_\_\_\_\_\_\_ Андрєєв Д. М.**

( підпис ) (прізвище та ініціали)

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

( підпис ) (прізвище та ініціали)

АНОТАЦІЯ

Програма описує інформаційну модель для збереження операцій видач і повернень дисків, а саме створення та подальша робота з об’єктами, що характеризують операції, дату, диск, людину тощо.

У пояснювальній записці викладена інформація щодо об’єктно-орієнтованого програмування та його особливостей й концепцій а також реалізація об’єктно-орієнтованого програмування у мові програмування С++ та її особливості й деякі елементи.

Також описано інформаційну модель, ієрархію класів та інші особливості виконання даного завдання.

**SUMMARY**

The program describes an information model for storing operations of ejections and disks, namely the creation and subsequent work with objects that characterize the operations, date, disk, person, etc.

The explanatory note provides information on object-oriented programming and its features and concepts, as well as the implementation of object-oriented programming in the C ++ programming language and its features and some elements.

The information model, class hierarchy, and other features of this task are also described.

зміст

[АНОТАЦІЯ 4](#_Toc27517730)

[ЗМІСТ 5](#_Toc27517731)

[ВСТУП 6](#_Toc27517732)

[Основна частина 7](#_Toc27517733)

[1. Теоретична частина 7](#_Toc27517734)

[1.1 Відомості про об’єктно-орієнтоване програмування 7](#_Toc27517735)

[1.2 Шаблони в мові програмування С++ 10](#_Toc27517736)

[2. Ключові моменти курсової роботи 14](#_Toc27517737)

[2.1 Опис інформаційної моделі 14](#_Toc27517738)

[2.2 Ієрархія класів 15](#_Toc27517739)

[2.3 Перевантаження операторів 16](#_Toc27517740)

[2.4 Обробка виключних ситуацій 18](#_Toc27517741)

[2.5 Обмеження певних даних у програмі 18](#_Toc27517741)

[2.6 Інтерфейс 18](#_Toc27517741)

[2.7 Багатопоточність 18](#_Toc27517741)

[2.8 Обмін данними в мережі 18](#_Toc27517741)

[2.9 Бібліотеки 20](#_Toc27517742)

[3. Можливі варіанти модернізації програми 21](#_Toc27517743)

[Висновки 22](#_Toc27517744)

[Список використаних джерел 23](#_Toc27517745)

[Додаток 1. Текст програми 24](#_Toc27517746)

[Додаток 2. Опис програми 46](#_Toc27517747)

Зм.

Лист

№ докум.

Підпис

Дата

Аркуш

5

УКР.НТУУ “КПІ імені І.Сікорського”\_ТЕФ\_АПЕПС\_TІ81131 3 КР 81-1

Розробив

Андрєєв Д. М.

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

52

ТЕФ

ВСТУП

З’являються нові мови програмування, новы стандарти і підходи до програмування. Цілью цієї роботи було створити систему, яка б описувала деякий процесс таким чином, як він існує в світі за допомогою концепцій об’єктно орієнтованого програмування. Також створити зручні інтерфейси для роботи і подальшої модифікації программного продукту, розділити програму на модулі, можливо, навіть на декілька программ, що працюють разом в тандемі.

Мова с++ є дуже гнучким інструментом для досягнення поставленної мети тому, що вона реаліхує в собі концепцію ООП і дозволяю дуже тонко контролювати процесси в комп’ютері. Мова дозволяє користуватися поліморфізмом, динамічним виділенням пам’яті, перегрузкою функцій і операторів, використання шаблонів тощо.

Створена модель повинна бути інтуїтивно зрозумілою і легкою в освоюванні.

Основна частина

# 1. Теоретична частина

## 1.1 Відомості про об’єктно-орієнтоване програмування

Об’єктно-орієнтоване програмування (надалі – ООП) – це дуже популярний підхід в програмуванні, який дозволяє об’єднувати деякий функціонал в упаковки - класси, які представляють собою окремі залежні або незалежні об’єкти, які мають свої поля і методи роботи з ними.

На думку Аллана Кея, розробника мови Smalltalk, якого вважають одним з засновників ООП, підхід полягає в наступних принципах:

1 Все є об’єктами

2 Всі дії та розрахунки виконуються шляхом взаємодії між об’єктами

3 Кожен об’єкт має незалежну пам’ять, яка складається з інших об’єктів

4 Кожен об’єкт є представником класу, який виражає загальні властивості об’єктів

5 У класі задається поведінка об’єкта

6 Класи організовані у єдину деревоподобну структуру з загальним корінням, яка називається ієрархією успадкування.

Таким чином, класс - це спеціальний опис того, як буде себе вести екземпляр цього классу, яким чином він буде змінювати совї поля, робити розрахунки тощо.

Об’єкт - це екземпляр классу з виділенною пам’ятю під всі свої змінні.

В ООП є три головні концепції: інкапсуляція, поліморфізм, та успадковування.

Інкапсуляція – правила доступу до внутрішніх полів та методів классу задля безопасності і логісного структурування.

В мові C++ існують такі специфікатори доступу:

1 private (закритий);

2 public (відкритий);

3 protected (захищений).

Якщо поле або метод знаходиться в межах дії специфікатора pubic, доступ до нього можуть отримувати всі класси або функції.

Специфікатор private дозволяє доступ до полів або методів классу тільки при умові, що функція, що намагається отримати доступ є методом цього классу або задекларована як дружня функція.

Специфікатор доступу protected розширює правила специфікатора private таким чином, що класс, успадкований від даного классу буде мати доступ до методів або полів з protected.

Наслідування - принцип, при якому поля і методи классу батька передаються классу нащадку.

При наслідуванні, можна змінити доступ до фунцій або полів(надалі елементів) калсу. Наприклад, при наслідуванні public всі елементи, що прийшли з базового класу не змінять свій стан доступу, при наслудуванні з специфікатором protected - всі public елементи стануть protected, а при наслідуванні private - всі public і protected елементи стануть private. Також, в похідному класі можна змінити доступ до деяких методів за допомогою ключового слова using

Наприклад:

using baseClass::inharitedFunction;

Також можна наслідувати класс від деяких базових. При цьому може з’явитися проблема ромбовидного наслідування, про яка буде описана пізніше.

Поліморфізм – це концепція, відповідно до якої використовується спільний інтерфейс для обробки даних різних типів.

В мові C++ є можливість перевантаження функцій а також операторів, що є функціями і методів классу, які також є функціями.

Поліморфізм зручно використовувати для того, щоб об’єднати в одну логічну структуру схожі методи роботи з даними. Також є поняття динамічного поліморфізму і пізнього зв’язування, суть яких в тому, що программа може викликати на стадії виконання функцію, невідому під час компіляції. Це дуже сильний метод в ООП і був реалізований в деяких модулях цієї курсової роботи.

## 1.2 Шаблони в мові програмування С++

Шаблони - це механізм, який дозволяє написати класс або фукнцію, які будуть працювати з наперед невідомим типом даних. Наприклад, у бібліотеці stl всі контейнери працюють з шаблонами, бо алгоритм роботи, наприклад, динамічного массиву однаковий для будь-якого типу даних.

Функція, що працює з шаблонами називається generic(родовою) функцією, а класс - родовим классом.

Шаблони допомогають структурувати логіку коду і зменшити кількість схожого коду. Для того, щоб об’явити родову функцію, потрібно вказати псевдонім наперед невідомого типу даних.

template <class T> (за старим стандартом typename T)

Важливо те, що на стадії компіляції створюються поліморфні функції для кожного типу даних, які передаються в ці функцію. Для визначення типу даних, ща потрапляє в функцію можна застосувати typeid із бібліотеки stl

Якщо у алгоритма родової функції є обмеження на тип даних, наприклад тільки чисельні типи, то можна застосувати функції stl is\_integral() тощо, або за допомогою функції static\_assert() збудити помилку помпілятора, і в ній описати що функція працює лише з чисельними даними.

Також, тип шаблону можна вказати при наслідуванні від родового класу за допомогою іншого шаблону або реального типу даних.

Для того, щоб викликати родову функцію або створити родовий класс, потрібно вказати в лапках <> тип даних.

int a = f<int>();

Class1<int> cls();

# **2. Ключові моменти курсової роботи**

## 2.1 Опис інформаційної моделі

Інформаційна модель, наведена у завданні описує структуру зберігання інформації стосовно певних об’єктів, а саме – денних звітів.

Денний звіт містить у собі два списка об’єктів - список операцій повернення диску і список операцій видачі диску.

Операція містить в собі дату, абонента, якому був виданий диск (який повертає диск) і сам диск.

Диск містить в собі інформацію про його довжину, дасту запису і студію, яка записала інформацію на цей диск.

Також присутні такі елементи, як дата, абонент і студія – вони є необхідними для коректного та повного опису моделі.

Задана модель може набути використання в різних галузях за умови, що ціллю буде зручне впорядкування інформації про певні об’єми операцій, де під певними об’ємами маються на увазі операції в кількості від кількох до кількох сотень тисяч.

Така модель буде зручною ще й тому, що за її допомогою може здійснюватися контроль за кількістю, датою операцій тощо, або можна використовувати її з метою швидкого пошуку операції в базі за певним параметром – цим параметром може бути ім’я абонента або його рік народження.

Для того, щоб показати кількість можливостей для застосування даної інформаційної моделі, наведемо кілька прикладів:

1. База даних магазину, що надає можливість контролювати брак і слідкувати за продажами.
2. Інформаційна база бібліотеки, що видає диски, що надає можливість слідкувати за видачею і поверненням дисків.

## 2.2 Ієрархія класів

Date2

Date1

Object

Рисунок 2.1 – Схематичне зображення ієрархії класів

Fileable

IsReturn

DailyReport

Operation

Serializeable

Disk

Abonent

Nameable

Date1

Studio

Date2

Object

Saveable

Loadable

Як видно з рисунку 2.1, у виконанні присутні як випадки агрегації, так й випадки успадковування. Перед тим як розібратись чому саме була вибрана дана ієрархічна модель, дамо означення терміну “агрегація”.

Агрегація – такий варіант створення класів, за якого інші класи включаються компонентно у вигляді полів класу радше ніж засобами успадковування. Агрегація є необхідною у тих випадках, коли похідний клас не доповнює логіку та структуру базового, а лише користується їм для певних цілей.

Найкращим прикладом успадковування є успадковування Date2 від Date1, так як клас Date2 доповнює батьківський клас та розширює його можливості.

Використання агрегації для подання списку елементів є очевидним, так як тільки таким чином можна досягти максимальної зручності й ефективності для цієї задачі.

Клас Abonent та класс Disk входить до классу Operation у вигляді агрегації за тією причиною, що Operation доповнює цей, а лише використовує в якості вказівників на особу і диск. Основною причиною для агрегації є те, що клас Operation (наприклад) не реалізує собою людину або диск, а лише посилається не деяку іншу людину і диск. Також, наприклад, класс Disk використовує агрегацію для збереження студії тому, що обидва цих класса наслідують дату, і наслідування диску від студії призведе до ромбовидного наслідування, що набажано.

Ромбовидне наслідування - тип наслідування, коли два класси наслідуються від одного спільного, а потом ще один класс наслідується від попередніх двох, що призводить до конфлікту імен.

## 2.3 Перевантаження операторів

Для зручності використання та підвищення функціоналу програми була використана така можливість ООП в С++, як перевантаження операторів.

Деякі оператори зазнали подальшого використання під час виконання інших завдань курсової роботи.

Для дати були перевантажені наступні оператори: >, <, ==, >=, <=.

Усі оператори порівняння повертають інтуїтивно зрозумілий результат – чим місце на часовій шкалі більше, тим дата й більше. Так, today >= yesterday поверне істину.

Перевантаження операторів с С++ є гарним механізмом, але не завжди інтуїтивно зрозумілим. Наприклад, можна перевантажити оператор додавання між диском і студією, але інтуїтивно незрозуміло що вийде. Дуже дивно бачити такий код: Elephant + MathFunction, незрозуміло що автор має на увазі, але якщо використовувати методи, все становиться зрозуміло: Elephant.getMassKg() + MathFunction.getXvalue()

Для классів, що повинні вміти копіюватися перевантажені оператори =, щоб копіювання проходило корректно. Особливо це важливо для классів, що викиристовують динамічне виділення пам’яті або мають вказівники на інші обєкти, які не повинні розділятися між різними єкземплярами классу, наприклад дескриптор потоку або дескриптор pipe

Для корректного запису в файл було створено класс FVector, мова про який піде пізніше. Для цього классу навмисно не було додано оператор індексування по двум причинам:

1. Цей оператор перейшов по наслідству від классу vector
2. Оператор індексування не завжди зручно і доцільно використовувати, так як може з’явитися плутаниця при роботу з вказівником на класс, коли замість оператора ідексування классу буде викликатися оператор індексування вказівника, тобто ми будемо рухатить по пам’яті і зловимо segmentation fault коли вийдемо за межі своєї пам’яті. Також в ситуації, коли маємо вказівник на массив вказівників на массив вказівників (наприклад), будемо мати з оператором індексування:

(\*((\*((\*arr)[i]))[i]))[i], що дуже погано читається, але з методом все дуже зрозуміло і розбiрливо: arr->get(i)->get(i)->get(i)

Для зручності, в деяких классах, які повинні виводитися на екран, перевантажено оператор << за стандартом.

## 2.4 Обробка виключних ситуацій

Під час виконання програми модуть виникати виключні ситуації, або, інакше кажучи, помилки. В мові C++ існує механізм обробки виключних ситуацій. Для того, чтоб ініціювати помилку, треба застосувати команду throw, вона завершує роботу функції пускає вниз по ієрархії викликів функцій обьект, який був переданий як параметр в throw.

Якщо функція буда викликана в блоку try, то почнеться виконання коду в блоці catch, якщо тип даних в кетч співпадає з типом в throw або класс, який був кинутий як виключна ситуація має в своєму дереві наслідуваннь класс, який приймається в блоці catch.

Якщо функція декларується з ключовим словом noexcept, то вона не може ініціювати виключны ситуації.

Під час виконання курсової роботи був використаний спосіб обробки виключних ситуацій, що використовує спеціальний клас з модуля <exception>, а саме std::exception.

Всі виключні ситуації мають в своєму дереві наслідувань класс std::exception, кожен з класів виключних ситуацій перевизначає метод

const char\* what() const noexcept;

, який повертає вказівник на строку, в якій вказан текст помилки.

Для демонстрації можливостей такого способу була створена ієрархія класів виключень (рис 2.2)

Рисунок 2.2 – Ієрархія класів виключень

UnexpectedTypeException

IncorrectObjectDataException

WrongInputFileException

InputIsNotReturn

InputIsNotExtradition

WrongInputValuesException

ZeroPointerException

IndexOutOfBoundsException

IncorrectRangeException

NoObjectFoundException

SearchException

WrongInputFileException

WrongInputDataException

UnknownDataTypeException

exception

## 2.5 Обмеження певних даних у програмі

Для перевірки корректності даних були використані наступні декілька функцій. Вони приймають строку і вертають булеве значення, істина при коректному вводі і брехня при некорректному. Ці функції мають наступні прототипи:

* bool isDay(wstring& input);
* bool isMonth(wstring& input);
* bool isInteger(wstring& input);
* bool isName(wstring& input);

## 2.6 Інтерфейс

Програмний продукт розділено на дві частини: front-end(надалі КЛІЄНТ) і back-end(надалі СЕРВЕР), які виконуюсть різні функції. СЕРВЕР займається зберіганням структури даних, інформації про операції, диски тощо і статистичною обробкою даних. В свою чергу КЛІЄНТ призначений для взаємодії з СЕРВЕРОМ. КЛІЄНТ може запросити деякі дані або, навпаки, відправити СЕРВЕРУ дані. Користувач керує КЛІЄНТОМ за допомогою меню (див у додатку WonderMenu).

В меню додані деякі команди, наприклад команда print виводить інформацію про операції видачі та повернення у табличному режимі (див у додатку ATable). Також присутні команди add(додає індормацію), stat(виводить статистичну інформацію у табличному режимі), save(зберігає інформацію у текстовий чи бінариний файл), load(загружає інформацію з текстового чи бінарного файлу), exit(вимикає програму), find(виводить відфільтровану інформацію про операції за параметром у табличному режимі)

Обмеження даних відбувається на стадії їх вводу на КЛІЄНТІ за допомогою функції wstring queryInput(bool (\*checker)(wstring& str), wstring message), в яку передається вкащзівник на функцію, що перевіряє корректність даних і повідомлення, що буде виведене при запиті даних.

Меню реалізовано за допомогою самописної бібліотеки WonderMenu. Для цієї роботи були створені класси пунктів меню і унаслідувані від абстрактного

классу WMenu::MenuCommand, з яким і працює бібліотека WonderMenu. Для цього був використаний динамічний поліморфізм, а саме перевизначення методу handleCommand void handleCommnad(wstring inputData)

Вивід меню оганізовано за допомогою классу Menu, який було унаслідувано від классу WMenu::WonderMenu і в ньому перевизначено віртуальний метод print, який слугуємеханізмом виводу таблиці на екран.

Таким чином, за допомогою пізднього зв’язування, вдалося створити меню (детальніше у додатку WonderMenu)

MenuCommand

MenuCommandStat

MenuCommandAdd

MenuCommandLoad

MenuCommandExit

MenuCommandFind

MenuCommandSave

MenuCommandPrint

Рисунок 2.3 - Ієрархія классів команд меню

## 2.7 Багатопоточність

Для досягнення концепції розділення програмного продукту на дві составні частини був використаний механізм багатопоточності, що надається самописною бібліотекою TurboPipes. Программа повинна одночасно спілкуватись і з корисувачем, і з сервером. Саме тому спілкування з сервером відбувається в окремому потоці. Для синхронізації програми в критичних секціях був використаний механізм mutex.

Критична секція - частина коду, робота з якою повинна виконуватись тільки синхронно, тобто тільки в одному потоці. Наприклад, додавання елементу у массив або виведення повідомлення в консоль.

Mutex представляє собою об’єкт, який має два стани: заблокований і разблокований. Коли процесс починає виконувати частину коду, що є критичною секцією, він повинен заблокувати mutex, а по завершенню виконання критичної секції разблокувати mutex. Якщо процесс намагається заблокувати все заблокований іншим процессом mutex - перший встане в режим очікування до тих пір, коли другий процесс разблокує mutex.

## 2.8 Обмін даними в мережі

Для обміну даними між сервером і клієнтом був застосований механізм named pipes, для цього була створена самописна бібліотека TurboPipes (див додаток TurboPipes). Программи обмінюються даними в форматі серіалізації JSON, для чого була написана бібліотека MagicJSON (див додаток MagicJSON)

Програми обмінюються даними за допомогою деяких, наперад обумовлених ключів. Наприклад, команда запиту даних у сервера виглядає так: {"command\_type":"get\_data","data\_type":"data\_all"}

Ключ команди - get\_data

Ключ даних - data\_all

Сервер інтерпретує цю команду як запит всієї інформації і висилає клієнту цю інформацію за допомогою таких же ключів. В свою чергу клієнт приймає повідомлення в форматі JSON, інтерпретує його і викинує деякі операції. Обробник даних можна знайти в классі NetworkMessagesHandler, який було унаслідувано від классу з бібліотеки TurboPipes: TurboPipes::Pipeable.

Для загрузки з текстового файлу, клієнт передає серверу повідомлення в форматі json з даними для загрузки, а сервер вже десеріалізує інформацію. Наприклад, так виглядає команда серверу загрузки з текстового фалу:

{"command\_type":"load\_data","load\_data":"load\_text","value":{"\_\_hash":-762614920,"\_\_type":"dailyreport","extraditions":[{"\_\_hash":433472154,"\_\_type":"operation","abonent":{"\_\_hash":-1498815132,"\_\_type":"abonent","date":{"\_\_hash":-482192549,"\_\_type":"date","month":7,"year":1990},"name":"Jim","surename":"Lowel"},"date":{"\_\_hash":-358701478,"\_\_type":"date","day":10,"month":11,"year":2001},"disk":{"\_\_hash":918228461,"\_\_type":"disk","date":{"\_\_hash":-358701478,"\_\_type":"date","day":19,"month":5,"year":2000},"length":1000,"name":"Ripers","studio":{"\_\_hash":1292480535,"\_\_type":"studio","date":{"\_\_hash":-358701478,"\_\_type":"date","day":23,"month":3,"year":1999},"name":"Loosers"}},"is\_return":0}],"returns":[{"\_\_hash":433472154,"\_\_type":"operation","abonent":{"\_\_hash":-1498815132,"\_\_type":"abonent","date":{"\_\_hash":-482192549,"\_\_type":"date","month":12,"year":1783},"name":"John","surename":"Parker"},"date":{"\_\_hash":-358701478,"\_\_type":"date","day":5,"month":10,"year":1993},"disk":{"\_\_hash":918228461,"\_\_type":"disk","date":{"\_\_hash":-358701478,"\_\_type":"date","day":30,"month":2,"year":1992},"length":100,"name":"Arrival","studio":{"\_\_hash":1292480535,"\_\_type":"studio","date":{"\_\_hash":-358701478,"\_\_type":"date","day":5,"month":3,"year":1987},"name":"Legacy"}},"is\_return":1}]}}

В якості параметру з ключом value передається інший json об’єкт, з якого вміє десеріалізуватися класс DailyReport на сервері.

Передача бінарного файлу на сервер реалізована за допомогою алгоритму base64, суть якого заключається в тому, щоб представити бінарні дані у текстовому вигляді, тобто за допомогою великих і малих літер латинського алфавіту і деяких знаківб таких як =, / тощо. Алгоритм працює по принципу розбиття даних на группи по 3 байта так представлення іх в 4-х байтовому вигляду зі здвигом значень байт так, щоб вони попадали в діапазон значень символів алфавіту і допоміжних знаків кодування ASCII. За допомогою цього алгоритму можжна передати бінарні дані у текстовому вигляді, але у метода є недолік - кількість пам’ятіб що займяє інформація, збільшується. Мережеве з’єднання в моїй програмі реалізовано за принципом питання - відповідь, тому більш доцільно передати інформацію за допомогою одної текстової pipe, замість того, щоб відкривати ще одну бунірну pipe, обмінюватись дескрипторами і передавати дані в чисто бінарному вигляді. Це призвело б до необхідності реалізації більш складного механіхму синхронізації, що недоцільно в даному випадку.

## 2.8 Бібліотеки

# Під час написання програми використовувалися наступні самописні бібліотеки:

# 1 ATable

# 2 TurboPipes

# 3 MagicJSON

# 4 WonderMenu

# Опис всіх цих бібліотек можна знайти в додатках.

# 3. Можливі варіанти модернізації програми

Як і будь-яку іншу програму, створену для вирішення певної прикладної проблеми, цю програму можна вдосконалити багатьма способами. Я хочу звернути увагу на деякі з них.

Надання програмі графічного інтерфейсу користувача дозволило б спростити взаємодію користувача з програмою до найпростіших операцій, а працівнику закладу, де використовують це програмне забезпечення, не потрібно було б навіть пояснювати, що потрібно робити і де.

Враховуючи те, що клієнт і сервер дві отремі незалежні програми - написання іншого клієнту, наприклад з GUI, стане задачею підключення до pipe і зв’язок із сервером за допомогою вже описаних ключів і команд, тобто в цій роботі буда розроблена гнучка система, що дозволяє легко підключати сторонні інтерфейси до серверу.

Доповнення до мережевого механізму можливість роботи з протоколами HTTP для того, щоб розширити можливосты мережевого з’єднання, в тому числі по інтернет протоколам а не тільки в локальній мережі.

Ткаож можна розширити мережеві можливості серверу таким чином, щоб він міг працювати з декількома клієнтами одночасно, наприклад домовитись,що pipe з назвою “vortex\_dispatcher” відповідає за ініціалізацію клаєнта і відправляє йому дескриптор pipe, на якій буде проходити сессія з цим клієнтом. Тобто, кожен клієнт спочатку заходить на pipe “vortex\_dispathcer”, вітається, називає себе і отримує дескриптор pipe для подальшого спілкування, відключається від “vortex\_dispatcher”, підключається на виданий pipe і спілкується з сервером там. Сервер, в свою чергу, створює новий потік, в якому виконується робота з клієнтом. В такому випадку, на сервері потрубно буде реалізувати механізм синхронізації в критичних секціях за допомогою, наприклад, mutex

Висновки

У ході виконання курсової роботи були досліджені важливі елементи об’єктно-орієнтованої мови програмування С++ та ООП в цілому. Були ретельно вивчені такі концепції ООП, як:

* інкапсуляція;
* поліморфізм;
* унаслідування;
* абстракція (яку іноді також приписують до головних концепцій ООП).

Окрім того, що ці концепції були вивчені й засвоєні в теорії, вони всі також були використані й на практиці.

Також був отриманий досвід у самостійній побудові ієрархії класів з використанням як унаслідування, так і агрегації.

Наступні можливості мови С++ були використані під час виконання роботи:

* Перевантажування операторів;
* Використання шаблонних функцій;
* Робота з файлами, в тому числі в двійковому режимі;
* Обробка виняткових ситуацій.
* Робота з багатопоточністю
* Мережеве з’єднання

Важливим моментом є те, що розроблений продукт гіпотетично може бути використаний на практиці за умови доробки необхідних елементів.

Список використаних джерел

1. "Latest news." TIOBE Index | TIOBE - The Software Quality Company. N.p., n.d. Web. 5 June 2017.

2. ["ISO/IEC TR 24733:2011"](https://www.iso.org/standard/38843.html). International Organization for Standardization.

3.  [Sutter, Herb](https://en.wikipedia.org/wiki/Herb_Sutter" \o "Herb Sutter); [Alexandrescu, Andrei](https://en.wikipedia.org/wiki/Andrei_Alexandrescu" \o "Andrei Alexandrescu) (2004). C++ Coding Standards: 101 Rules, Guidelines, and Best Practices. Addison-Wesley.

Додаток 1. Текст програми

Abonent.h

#pragma once

#include "Date1.h"

#include "Nameable.h"

#include "FWString.h"

class Abonent : public Date1, public Nameable{

protected:

FWString surename;

public:

Abonent(const FWString firstname, const FWString surename, const short year, const short month) : Date1(year, month), Nameable(firstname), surename(surename) {}

Abonent() : Date1(), Nameable(), surename(L"") {}

Abonent(const Abonent& reference);

~Abonent() {}

void setSurename(const FWString surename);

FWString getSurename() const;

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator=(const Abonent& reference);

};

Abonent.cpp

#include "Abonent.h"

Abonent::Abonent(const Abonent& reference) : Date1(reference), Nameable(reference) {

this->surename = reference.surename;

}

void Abonent::setSurename(const FWString surename) {

this->surename = surename;

}

FWString Abonent::getSurename() const {

return this->surename;

}

void Abonent::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Abonent).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

this->name.save(stream);

this->surename.save(stream);

Date1::save(stream);

}

void Abonent::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Abonent).hash\_code()) {

throw WrongInputFileException();

}

this->name.load(stream);

this->surename.load(stream);

Date1::load(stream);

}

MagicJSON::JsonObject Abonent::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"abonent");

json.addInteger(L"\_\_hash", typeid(Abonent).hash\_code());

json.addString(L"name", this->name);

json.addString(L"surename", this->surename);

json.addObject(L"date", Date1::serialize());

return json;

}

void Abonent::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"abonent") {

throw UnexpectedTypeException();

}

this->name = json.getString(L"name");

this->surename = json.getString(L"surename");

Date1::deserialize(json.getObject(L"date"));

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Abonent::operator=(const Abonent& reference) {

this->surename = reference.surename;

Date1::operator=(reference);

Nameable::operator=(reference);

}

DailyReport.h

#pragma once

#include "Object.h"

#include "FVector.h"

#include "Operation.h"

#include "DailyReportException.h"

class DailyReport : public Object, public Fileable, public Serializeable{

protected:

FVector returns;

FVector extraditions;

public:

DailyReport();

~DailyReport() {}

DailyReport(const DailyReport& reference);

void addReturn(Operation\* return\_);

Operation\* getReturn(const size\_t index) const;

size\_t getReturnsSize();

void addExtradition(Operation\* extradition);

Operation\* getExtradition(const size\_t index) const;

size\_t getExtraditionsSize();

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

Operation find(const wstring name) const;

Operation find(const short year) const;

public:

void operator= (const DailyReport& reference);

};

DailyReport.cpp

#include "DailyReport.h"

DailyReport::DailyReport() {

this->extraditions.addObjectCreator((hash\_code)typeid(Operation).hash\_code(), []() -> Fileable \* { return new Operation(); });

this->returns.addObjectCreator((hash\_code)typeid(Operation).hash\_code(), []() -> Fileable \* { return new Operation(); });

}

DailyReport::DailyReport(const DailyReport& reference) : Object(reference) {

this->returns = reference.returns;

this->extraditions = reference.extraditions;

}

void DailyReport::addReturn(Operation\* return\_) {

if (!return\_->isReturn()) {

throw InputIsNotReturn();

}

this->returns.push\_back(return\_);

}

Operation\* DailyReport::getReturn(const size\_t index) const {

Operation\* temp = dynamic\_cast<Operation\*>(this->returns.get(index));

if (!temp) {

throw ZeroPointerException();

}

return temp;

}

size\_t DailyReport::getReturnsSize() {

return this->returns.size();

}

void DailyReport::addExtradition(Operation\* extradition) {

if (extradition->isReturn()) {

throw InputIsNotExtradition();

}

this->extraditions.push\_back(extradition);

}

Operation\* DailyReport::getExtradition(const size\_t index) const {

Operation\* temp = dynamic\_cast<Operation\*>(this->extraditions.get(index));

if (!temp) {

throw ZeroPointerException();

}

return temp;

}

size\_t DailyReport::getExtraditionsSize() {

return this->extraditions.size();

}

void DailyReport::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(DailyReport).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

this->extraditions.save(stream);

this->returns.save(stream);

}

void DailyReport::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(DailyReport).hash\_code()) {

throw WrongInputFileException();

}

this->extraditions.load(stream);

this->returns.load(stream);

}

MagicJSON::JsonObject DailyReport::serialize() {

MagicJSON::JsonObject json;

MagicJSON::JsonArray jreturns;

MagicJSON::JsonArray jextraditions;

for (Fileable\* item : this->returns) {

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) != nullptr) {

jreturns.addObject(serializeable->serialize());

}

}

for (Fileable\* item : this->extraditions) {

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) != nullptr) {

jextraditions.addObject(serializeable->serialize());

}

}

json.addString(L"\_\_type", L"dailyreport");

json.addInteger(L"\_\_hash", typeid(DailyReport).hash\_code());

json.addArray(L"returns", jreturns);

json.addArray(L"extraditions", jextraditions);

return json;

}

void DailyReport::deserialize(MagicJSON::JsonObject json) {

try {

this->returns.clear();

this->extraditions.clear();

wstring type = json.getString(L"\_\_type");

if (type.compare(wstring(L"dailyreport"))) {

throw UnexpectedTypeException();

}

MagicJSON::JsonArray jreturns = json.getArray(L"returns");

MagicJSON::JsonArray jextraditions = json.getArray(L"extraditions");

for (size\_t i = 0; i < jreturns.size(); i++) {

MagicJSON::JsonObject object = jreturns.getObject(i);

long hash = object.getInteger(L"\_\_hash");

size\_t s\_hash = (size\_t)hash;

Fileable\* item = this->returns.getObjectCreator(s\_hash)();

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) == nullptr) {

throw IncorrectObjectDataException();

}

serializeable->deserialize(object);

this->returns.push\_back(item);

}

for (size\_t i = 0; i < jextraditions.size(); i++) {

MagicJSON::JsonObject object = jextraditions.getObject(i);

Fileable\* item = this->returns.getObjectCreator(object.getInteger(L"\_\_hash"))();

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) == nullptr) {

throw IncorrectObjectDataException();

}

serializeable->deserialize(object);

this->extraditions.push\_back(item);

}

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

Operation DailyReport::find(const wstring name) const {

for (Fileable\* item : this->returns) {

Operation\* operation = dynamic\_cast<Operation\*>(item);

if (operation) {

if (operation->getAbonent().getName() == name) {

return \*operation;

}

}

}

throw NoObjectFoundException();

}

Operation DailyReport::find(const short year) const {

for (Fileable\* item : this->returns) {

Operation\* operation = dynamic\_cast<Operation\*>(item);

if (operation) {

if (operation->getAbonent().getYear() == year) {

return \*operation;

}

}

}

throw NoObjectFoundException();

}

void DailyReport::operator=(const DailyReport& reference) {

this->returns = reference.returns;

this->extraditions = reference.extraditions;

}

Date1.h

#pragma once

#include "Object.h"

#include "Fileable.h"

#include "Exception.h"

#include "Serializeable.h"

class Date1 : public Object, public Fileable, public Serializeable {

protected:

short year;

short month;

public:

Date1(const short year, const short month);

Date1() : Date1(2000, 1) {}

~Date1() {}

Date1(const Date1& reference);

short getYear() const;

void setYear(const short year);

short getMonth() const;

void setMonth(const short year);

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Date1& reference);

bool operator< (const Date1& right);

bool operator> (const Date1& right);

bool operator<= (const Date1& right);

bool operator>= (const Date1& right);

bool operator== (const Date1& right);

};

Date1.cpp

#include "Date1.h"

Date1::Date1(const short year, const short month) : Object() {

if (year < 0 || month <= 0 || month > 12) {

throw WrongInputValuesException();

}

this->year = year;

this->month = month;

}

Date1::Date1(const Date1& reference) : Object(reference) {

this->year = reference.year;

this->month = reference.month;

}

short Date1::getYear() const {

return this->year;

}

void Date1::setYear(const short year) {

if (year < 0) {

throw WrongInputValuesException();

}

this->year = year;

}

short Date1::getMonth() const {

return this->month;

}

void Date1::setMonth(const short month) {

if (month < 0 || month > 11) {

throw WrongInputValuesException();

}

this->month = month;

}

void Date1::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Date1).hash\_code();

stream.write((char\*)&hash, sizeof(hash\_code));

stream.write((char\*)&(this->year), sizeof(short));

stream.write((char\*)&(this->month), sizeof(short));

}

void Date1::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*) & hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Date1).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*) & (this->year), sizeof(short));

stream.read((char\*) & (this->month), sizeof(short));

}

MagicJSON::JsonObject Date1::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"date");

json.addInteger(L"\_\_hash", typeid(Date1).hash\_code());

json.addInteger(L"month", (long)this->month);

json.addInteger(L"year", (long)this->year);

return json;

}

void Date1::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"date") {

throw UnexpectedTypeException();

}

this->month = (short)json.getInteger(L"month");

this->year = (short)json.getInteger(L"year");

}

catch (MagicJSON::NoObjectFoundException e){

throw IncorrectObjectDataException();

}

}

void Date1::operator=(const Date1& reference) {

this->year = reference.year;

this->month = reference.month;

}

Date2.h

#pragma once

#include "Date1.h"

class Date2 : public Date1 {

protected:

short day;

public:

Date2(const short year, const short month, const short day); // TODO: Finish exceptions when 31 and 30 days

Date2() : Date1(), day(0) {}

~Date2() {}

Date2(const Date2& reference);

short getDay() const;

void setDay(const short day);

public:

void save(ofstream & stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator=(const Date2& reference);

};

Date2.cpp

#include "Date2.h"

Date2::Date2(const short year, const short month, const short day) : Date1(year, month) {

this->setDay(day);

}

Date2::Date2(const Date2& reference) : Date1(reference) {

this->day = reference.day;

}

short Date2::getDay() const {

return this->day;

}

void Date2::setDay(const short day) {

if (day < 0 || day > 31) {

throw WrongInputValuesException();

}

if (this->month % 2 && day > 30) {

throw WrongInputValuesException();

}

if (this->month == 11 && (day > 28)) {

throw WrongInputValuesException();

}

this->day = day;

}

void Date2::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Date2).hash\_code();

stream.write((char\*)&hash, sizeof(hash\_code));

stream.write((char\*) & (this->day), sizeof(short));

Date1::save(stream);

}

void Date2::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)&hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Date2).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*) & (this->day), sizeof(short));

Date1::load(stream);

}

void Date2::operator=(const Date2& reference) {

this->day = reference.day;

Date1::operator=(reference);

}

MagicJSON::JsonObject Date2::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"date");

json.addInteger(L"\_\_hash", typeid(Date2).hash\_code());

json.addInteger(L"month", (long)this->month);

json.addInteger(L"year", (long)this->year);

json.addInteger(L"day", (long)this->day);

return json;

}

void Date2::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"date") {

throw UnexpectedTypeException();

}

this->month = (short)json.getInteger(L"month");

this->year = (short)json.getInteger(L"year");

this->day = (short)json.getInteger(L"day");

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

Disk.h

#pragma once

#include "Date2.h"

#include "String.h"

#include "Studio.h"

#include "Nameable.h"

class Disk : public Date2, public Nameable {

protected:

unsigned int length;

Studio studio;

public:

Disk(const Studio studio, const unsigned int length, const FWString name, const short year, const short month, const short day) :

Date2(year, month, day), Nameable(name), studio(studio), length(length) {}

Disk() : Date2(), Nameable(), length(0), studio(Studio()) {}

Disk(const Disk& reference);

~Disk() {}

void setLength(const unsigned int length);

unsigned int getLength() const;

void setStudio(const Studio studio);

Studio getStudio() const;

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Disk& reference);

};

Disk.cpp

#include "Disk.h"

Disk::Disk(const Disk& reference) : Date2(reference), Nameable(reference) {

this->studio = reference.studio;

this->length = reference.length;

}

void Disk::setLength(const unsigned int length) {

this->length = length;

}

unsigned int Disk::getLength() const {

return this->length;

}

void Disk::setStudio(const Studio studio) {

this->studio = studio;

}

Studio Disk::getStudio() const {

return this->studio;

}

void Disk::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Disk).hash\_code();

stream.write((char\*)&hash, sizeof(hash\_code));

stream.write((char\*) & (this->length), sizeof(unsigned int));

this->studio.save(stream);

Date2::save(stream);

this->name.save(stream);

}

void Disk::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)&hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Disk).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*) & (this->length), sizeof(unsigned int));

this->studio.load(stream);

Date2::load(stream);

this->name.load(stream);

}

MagicJSON::JsonObject Disk::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"disk");

json.addInteger(L"\_\_hash", typeid(Disk).hash\_code());

json.addObject(L"date", Date2::serialize());

json.addString(L"name", this->name);

json.addInteger(L"length", this->length);

json.addObject(L"studio", this->studio.serialize());

return json;

}

void Disk::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"disk") {

throw UnexpectedTypeException();

}

this->name = json.getString(L"name");

this->length = json.getInteger(L"length");

Date2::deserialize(json.getObject(L"date"));

this->studio.deserialize(json.getObject(L"studio"));

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Disk::operator=(const Disk& reference) {

this->length = reference.length;

this->studio = reference.studio;

Date2::operator=(reference);

Nameable::operator=(reference);

}

Nameable.h

#pragma once

#include "String.h"

#include "FWString.h"

class Nameable {

protected:

FWString name;

public:

Nameable(const FWString name) : name(name) {}

Nameable() : Nameable(L"") {}

Nameable(const Nameable& reference) {

this->name = reference.name;

}

~Nameable() {}

void setName(const FWString name) {

this->name = name;

}

FWString getName() const {

return this->name;

}

public:

void operator= (const Nameable& reference);

};

Operation.h

#pragma once

#include "Returnable.h"

#include "Abonent.h"

#include "Date2.h"

#include "Disk.h"

class Operation : public Returnable, public Fileable, public Serializeable {

protected:

Date2 date;

Abonent abonent;

Disk disk;

public:

Operation(Date2 date, Abonent abonent, Disk disk, bool isreturn) : Returnable(isreturn), date(date), abonent(abonent), disk(disk) {}

Operation() : Returnable(), date(Date2()), abonent(Abonent()), disk(Disk()) {}

~Operation() {}

Operation(const Operation & reference);

void setDate(const Date2 date);

Date2 getDate() const;

void setAbonent(const Abonent abonent);

Abonent getAbonent() const;

void setDisk(const Disk disk);

Disk getDisk() const;

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Operation& reference);

};

Operation.cpp

#include "Operation.h"

Operation::Operation(const Operation& reference) {

this->date = reference.date;

this->abonent = reference.abonent;

this->disk = reference.disk;

}

void Operation::setDate(const Date2 date) {

this->date = date;

}

Date2 Operation::getDate() const {

return this->date;

}

void Operation::setAbonent(const Abonent abonent) {

this->abonent = abonent;

}

Abonent Operation::getAbonent() const {

return this->abonent;

}

void Operation::setDisk(const Disk disk) {

this->disk = disk;

}

Disk Operation::getDisk() const {

return this->disk;

}

void Operation::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Operation).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

stream.write((char\*)& this->isreturn, sizeof(bool));

this->date.save(stream);

this->abonent.save(stream);

this->disk.save(stream);

}

void Operation::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Operation).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*)& this->isreturn, sizeof(bool));

this->date.load(stream);

this->abonent.load(stream);

this->disk.load(stream);

}

MagicJSON::JsonObject Operation::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"operation");

json.addInteger(L"\_\_hash", typeid(Operation).hash\_code());

json.addObject(L"date", this->date.serialize());

json.addObject(L"abonent", this->abonent.serialize());

json.addObject(L"disk", this->disk.serialize());

json.addInteger(L"is\_return", (long)(this->isreturn));

return json;

}

void Operation::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"operation") {

throw UnexpectedTypeException();

}

this->date.deserialize(json.getObject(L"date"));

this->abonent.deserialize(json.getObject(L"abonent"));

this->disk.deserialize(json.getObject(L"disk"));

this->isreturn = (bool)json.getInteger(L"is\_return");

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Operation::operator=(const Operation& reference) {

this->date = reference.date;

this->abonent = reference.abonent;

this->disk = reference.disk;

}

Returnable.h

#pragma once

class Returnable {

protected:

bool isreturn;

public:

Returnable(const bool isreturn) : isreturn(isreturn) {}

Returnable() : Returnable(false) {}

~Returnable() {}

Returnable(const Returnable& reference);

bool isReturn() const { return this->isreturn; }

void setType(const bool isreturn) { this->isreturn = isreturn; }

public:

void operator= (const Returnable& reference);

};

Returnable.cpp

#include "Returnable.h"

Returnable::Returnable(const Returnable& reference) {

this->isreturn = reference.isreturn;

}

void Returnable::operator=(const Returnable& reference) {

this->isreturn = reference.isreturn;

}

Serializaeble.h

#pragma once

#include "MagicJSON.h"

#include "SerializeExceptions.h"

class Serializeable {

public:

virtual MagicJSON::JsonObject serialize() = 0;

virtual void deserialize(MagicJSON::JsonObject json) = 0;

};

Studio.h

#pragma once

#include "Date2.h"

//#include "FWString.h"

#include "Nameable.h"

class Studio : public Date2, public Nameable{

public:

Studio(const FWString name, const short year, const short month, const short day) : Date2(year, month, day), Nameable(name) {}

Studio() : Date2(), Nameable(L"") {}

Studio(const Studio& reference);

~Studio() {}

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Studio& reference);

};

Studio.cpp

#include "Studio.h"

Studio::Studio(const Studio& reference) : Date2(reference), Nameable(reference){

//this->name = reference.name;

}

void Studio::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Studio).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

Date2::save(stream);

this->name.save(stream);

}

void Studio::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Studio).hash\_code()) {

throw WrongInputFileException();

}

Date2::load(stream);

this->name.load(stream);

}

MagicJSON::JsonObject Studio::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"studio");

json.addInteger(L"\_\_hash", typeid(Studio).hash\_code());

json.addObject(L"date", Date2::serialize());

json.addString(L"name", this->name);

return json;

}

void Studio::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"studio") {

throw UnexpectedTypeException();

}

this->name = json.getString(L"name");

Date2::deserialize(json.getObject(L"date"));

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Studio::operator=(const Studio& reference) {

Date2::operator=(reference);

Nameable::operator=(reference);

}

Fileable.h

#pragma once

#include "Loadable.h"

#include "Saveable.h"

typedef long hash\_code;

class Fileable : public Loadable, public Saveable {

public:

Fileable() {}

~Fileable() {}

};

Loadable.h

#pragma once

#include <fstream>

using namespace std;

class Loadable {

public:

Loadable() {}

~Loadable() {}

virtual void load(ifstream& stream) = 0;

};

Saveable.h

#pragma once

#include <fstream>

using namespace std;

class Saveable {

public:

Saveable() {}

~Saveable() {}

virtual void save(ofstream& stream) const = 0;

};

FVector.h

#pragma once

#include "Fileable.h"

#include "HashSet.h"

#include "Object.h"

#include "Exception.h"

#include <vector>

class FVector : public vector<Fileable\*>, public Fileable, public Object{

HashSet<Fileable\* (\*)(void)> creators;

public:

FVector(): vector<Fileable\*>(), Fileable(), Object() {}

FVector(const FVector& reference);

~FVector() {}

virtual Fileable\* get(size\_t index) const;

virtual void save(ofstream& stream) const override;

virtual void load(ifstream& stream) override;

virtual void addObjectCreator(size\_t hash, Fileable\* (\*creator)(void));

virtual Fileable\* (\*getObjectCreator(size\_t hash))(void);

public:

void operator=(const FVector& reference);

};

FWString.h

#pragma once

#include "Fileable.h"

#include "Exception.h"

#include <windows.h>

class FWString : public wstring, public Fileable {

public:

FWString() : wstring(), Fileable() {}

FWString(const LPCWSTR str) : wstring(str), Fileable() {}

FWString(const FWString& reference) : wstring(reference), Fileable(reference) {}

FWString(const wstring& reference) : wstring(reference) {}

~FWString() {}

virtual void save(ofstream& stream) const override;

virtual void load(ifstream& stream) override;

public:

using wstring::operator+=;

using wstring::operator[];

using wstring::operator=;

};

HashSet.h

#pragma once

#include "Object.h"

#include "Exception.h"

#include <set>

#include <type\_traits>

namespace {

template <class T>

class HashSetNode : public Object {

public:

T value;

size\_t key;

HashSetNode(size\_t key, T value) : value(value), key(key) {}

HashSetNode() : HashSetNode(0, T()) {}

};

}

template <class T>

class HashSet : protected set<HashSetNode<T>, bool(\*)(const HashSetNode<T>& less, const HashSetNode<T>& higher)>, public Object{

typedef set<HashSetNode<T>, bool(\*)(const HashSetNode<T>& less, const HashSetNode<T>& higher)> Set;

public:

HashSet() : Set(&comparator) {}

HashSet(const HashSet& reference) : Set(reference), Object(reference){}

~HashSet() {}

virtual void add(size\_t key, T element);

virtual T get(const size\_t key) const;

using Set::cend;

using Set::cbegin;

using Set::size;

using Set::operator=;

using Set::find;

using Set::empty;

using Set::max\_size;

using Set::clear;

using Set::erase;

protected:

static bool comparator(const HashSetNode<T>& less, const HashSetNode<T>& higher) {

return less.key < higher.key;

}

};

template<class T>

inline void HashSet<T>::add(size\_t key, T element) {

this->insert(HashSetNode<T>(key, element));

}

template<class T>

inline T HashSet<T>::get(const size\_t key) const {

if (!this->count(HashSetNode<T>(key, T()))) {

throw NoObjectFoundException();

}

return this->find(HashSetNode<T>(key, T()))->value;

}

NetworkMessageKeys.h

#pragma once

#include <string>

using namespace std;

// KEYS

const wstring COMMAND\_TYPE\_KEY = L"command\_type";

const wstring DATA\_TYPE\_KEY = L"data\_type";

const wstring REPORT\_TYPE\_KEY = L"report\_type";

const wstring FILTER\_TYPE\_KEY = L"filter\_type";

const wstring FILTER\_VALUE\_KEY = L"filter\_value";

const wstring LOAD\_DATA\_KEY = L"load\_data";

const wstring SAVE\_DATA\_KEY = L"save\_data";

const wstring VALUE\_KEY = L"value";

const wstring SIZE\_KEY = L"size";

const wstring PATH\_KEY = L"path";

const wstring ERROR\_TYPE\_KEY = L"error\_type";

const wstring SUCCESS\_TYPE\_KEY = L"success\_type";

// COMANDS

const wstring COMMAND\_GET\_DATA = L"get\_data";

const wstring COMMAND\_GET\_REPORT = L"get\_report";

const wstring COMMAND\_ADD\_DATA = L"add\_data";

const wstring COMMAND\_SEND\_DATA = L"send\_data";

const wstring COMMAND\_ERROR = L"error";

const wstring COMMAND\_SUCCESS = L"success";

const wstring COMMAND\_LOAD = L"load\_data";

const wstring COMMAND\_SAVE = L"save\_data";

const wstring COMMAND\_TERMINATE = L"terminate";

// ERROR VALUES

const wstring ERROR\_INVALID\_COMMAND = L"invalid\_command";

const wstring ERROR\_INVALID\_FILE = L"invalid\_file";

const wstring ERROR\_INVALID\_VALUES = L"invalid\_values";

// SUCCESS VALUES

const wstring SUCCESS\_READING\_FILE = L"success\_reading\_file";

const wstring SUCCESS\_ADDING\_DATA = L"success\_adding\_data";

// DATA TYPES

const wstring DATA\_ALL = L"data\_all";

const wstring DATA\_FILTERED = L"data\_fitered";

const wstring DATA\_OPERATION = L"data\_operation";

// FILTER TYPES

const wstring FILTER\_TYPE\_STRING = L"filter\_string";

const wstring FILTER\_TYPE\_INTEGER = L"filter\_integer";

// LOAD TYPES

const wstring LOAD\_BINARY = L"load\_binary";

const wstring LOAD\_TEXT = L"load\_text";

// SAVE TYPES

const wstring SAVE\_BINARY = L"save\_binary";

const wstring SAVE\_TEXT = L"save\_text";

// REPORT TYPES

const wstring REPORT\_STATISTIC = L"report\_statistic";

NetworkMessagesHandler.h

#pragma once

#include "TurboPipes.h"

#include "DailyReport.h"

#include "NetworkMessageKeys.h"

#include <string>

#include <iostream>

using namespace std;

class NetworkMessagesHandler : public TurboPipes::PipeableString {

protected:

DailyReport\* dailyReport;

public:

NetworkMessagesHandler(DailyReport\* dailyReport);

~NetworkMessagesHandler();

public:

void handleMessage(wstring& message) override;

protected:

void handleGetAllMessage();

void handleGetFilteredMessage(MagicJSON::JsonObject message);

void handleLoadTextFileMessage(MagicJSON::JsonObject message);

void handleSaveTextFileMessage(MagicJSON::JsonObject message);

void handleLoadBinaryFileMessage(MagicJSON::JsonObject message);

void handleSaveBinaryFileMessage(MagicJSON::JsonObject message);

void handleAddOperationMessage(MagicJSON::JsonObject message);

void handleReportStaticticMessage(MagicJSON::JsonObject message);

static MagicJSON::JsonObject buildOperationJson(Operation\* operation);

};

DailyReportExceptions.h

#pragma once

#include "Exception.h"

class InputIsNotReturn : public WrongInputDataException {

public:

char const\* what() const noexcept override { return "Error: InputIsNotReturn"; }

};

class InputIsNotExtradition : public WrongInputDataException {

public:

char const\* what() const noexcept override { return "Error: InputIsNotExtradition"; }

};

Exception.h

#pragma once

#include <exception>

using namespace std;

class WrongInputFileException : public exception {

public:

char const\* what() const noexcept override { return "Error: WrongInputFileException"; }

};

class UnknownDataTypeException : public exception {

public:

char const\* what() const noexcept override { return "Error: UnknownDataTypeException"; }

};

class IndexOutOfBoundsException : public exception {

public:

char const\* what() const noexcept override { return "Error: IndexOutOfBoundsException"; }

};

class WrongInputDataException : public exception {

public:

char const\* what() const noexcept override { return "Error: WrongInputDataException"; }

};

class WrongInputValuesException : public WrongInputDataException {

public:

char const\* what() const noexcept override { return "Error: WrongInputValuesException"; }

};

class SearchException : public exception {

public:

char const\* what() const noexcept override { return "Error: SearchException"; }

};

class NoObjectFoundException : public SearchException {

public:

char const\* what() const noexcept override { return "Error: NoObjectFoundException"; }

};

class IncorrectRangeException : public SearchException {

public:

char const\* what() const noexcept override { return "Error: IncorrectRangeException"; }

};

class ZeroPointerException : public exception {

public:

char const\* what() const noexcept override { return "Error: ZeroPointerException"; }

};

SerializeExceptions.h

#pragma once

#include <exception>

using namespace std;

class SerializeException : public exception {

public:

const char\* what() const noexcept override { return "Error: SerializeException"; }

};

class IncorrectObjectDataException : public SerializeException {

public:

const char\* what() const noexcept override { return "Error: IncorrectObjectDataException"; }

};

class UnexpectedTypeException : public SerializeException {

public:

const char\* what() const noexcept override { return "Error: UnexpectedTypeException"; }

};

Object.h

#pragma once

using namespace std;

class Object {

public:

Object() {}

Object(const Object& reference) {}

virtual ~Object() {}

};

StatTools.h

#pragma once

#include "DailyReport.h"

#include <string>

namespace StatTools {

template <class T>

double statistic(DailyReport\* target) {

double total = 0;

size\_t number = 0;

for (size\_t i = 0; i < target->getReturnsSize(); i++) {

if (typeid(T) == typeid(int)) {

total += target->getReturn(i)->getDisk().getLength();

number++;

}

if (typeid(T) == typeid(std::wstring)) {

total += target->getReturn(i)->getDisk().getName().length();

number++;

}

}

for (size\_t i = 0; i < target->getExtraditionsSize(); i++) {

if (typeid(T) == typeid(int)) {

total += target->getExtradition(i)->getDisk().getLength();

number++;

}

if (typeid(T) == typeid(std::wstring)) {

total += target->getExtradition(i)->getDisk().getName().length();

number++;

}

}

if (!number) {

return 0;

}

return total / number;

}

}

Menu.h

#pragma once

#include "WMenu.h"

#include "ATable.h"

#include <iostream>

#include <string>

using namespace std;

class Menu : public WMenu::WonderMenu {

public:

Menu();

~Menu();

public:

void print() override;

};

MenuCommandAdd.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

const LPCWSTR INCORRECT\_DAY\_MSG = L"Incorrect day, enter integer mote than 0 and less or equal 31";

const LPCWSTR INCORRECT\_MONTH\_MSG = L"Incorrect month, enter integer mote than 0 and less or equal 12";

const LPCWSTR INCORRECT\_YEAR\_MSG = L"Incorrect year, enter integer value";

const LPCWSTR INCORRECT\_NUMBER\_MSG = L"Incorrect number, enter integer value";

const LPCWSTR INCORRECT\_NAME\_MSG = L"Incorrect name, first letter should be capital";

class MenuCommandAdd : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandAdd(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandAdd();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandExit.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

class MenuCommandExit : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandExit(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandExit();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandFind.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

class MenuCommandFind : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandFind(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandFind();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandLoad.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

class MenuCommandLoad : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandLoad(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandLoad();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandPrint.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

class MenuCommandPrint : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandPrint(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandPrint();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandSave.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

class MenuCommandSave : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandSave(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandSave();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandStat.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

class MenuCommandStat : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandStat(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandStat();

public:

void handleCommnad(wstring inputData) override;

};

Directories.h

#pragma once

#include <string>

const std::wstring FILES\_DIRECTORY = L"files\\";

const std::wstring BINARY\_EXTENSION = L".binary";

const std::wstring TEXT\_EXTENSION = L".json";

InputChecker.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

bool isDay(wstring& input);

bool isMonth(wstring& input);

bool isInteger(wstring& input);

bool isName(wstring& input);

wstring queryInput(bool (\*checker)(wstring& str), wstring message);

NetworkMessagesHandler.h

#pragma once

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include "ATable.h"

#include <string>

#include <iostream>

using namespace std;

class NetworkMessagesHandler : public TurboPipes::PipeableString {

public:

NetworkMessagesHandler();

~NetworkMessagesHandler();

public:

void handleMessage(wstring& message) override;

protected:

void handleSendAllDataMessage(MagicJSON::JsonObject message);

void handleSaveTextMessage(MagicJSON::JsonObject message);

void handleSaveBinaryMessage(MagicJSON::JsonObject message);

void handleAddDataSuccessMessage(MagicJSON::JsonObject message);

void handleGetStatisticMessage(MagicJSON::JsonObject message);

static void addOperationToTable(MagicJSON::JsonObject operation, ATable::Table\* table);

static ATable::Table\* constructTableForOperations(string name);

};

Додаток 2. Опис програми

АНОТАЦІЯ

В додатку приведено ілюстрації до ходу виконання програми а також інша корисна інформація така як, наприклад, спосіб вводу та виводу даних та спосіб й вид зберігання цих даних на жорсткому диску комп’ютера.

**SUMMARY**

The addition provides illustrations of the progress of the program, as well as other useful information such as method of data insertion and data output and how it will be stored on your computer's hard drive.

**ЗМІСТ**

[АНОТАЦІЯ 47](#_Toc27517857)

[ЗМІСТ 48](#_Toc27517858)

[Інформаційна частина 49](#_Toc27517859)

[1. Загальна інформація 49](#_Toc27517860)

[2. Опис структури програми 49](#_Toc27517861)

[3. Використовувані технічні засоби 50](#_Toc27517862)

[4. Компіляція і запуск програми 50](#_Toc27517863)

[5. Вхідні й вихідні дані 50](#_Toc27517864)

Інформаційна частина

## 1. Загальна інформація

Програма має на меті максимально можливу для поставленої задачі простоту взаємодії користувача із терміналом, при чому не має жодного графічного інтерфейсу користувача. Таблиці для виводу даних побудовані за допомогою елементів псевдографіки. Повний функціонал програми зображено на рисунку 1.1, який відображає меню програми.

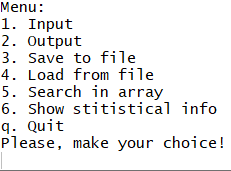


Рисунок 1.1 – Меню програми

## **2. Опис структури програми**

В головній функції програми int main() виконується цикл до тих пір, поки користувачем не буде введено символ виходу з програми. При переході до інших пунктів меню управління передається внутрішнім циклам з тією ж умовою або просто виконуються необхідні дії.

При записі даних в файл користувач повинен задати ім’я файлу й вибрати тип збереження, а саме текстовий чи двійковий.

При зчитуванні з файлу програма покаже назви файлів, з яких можливо зчитати дані, й користувачу залишиться лише ввести необхідну назву, як показано на рисунку 2.1.

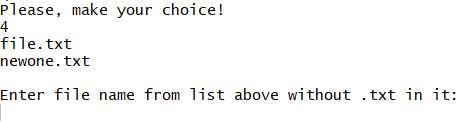


Рисунок 2.1 – Пропонування програмою можливих файлів

## **3. Використовувані технічні засоби**

Для написання курсової роботи були використані наступні технічні засоби:

Visual Studio Code – текстовий редактор.

cygwin – unix-подібний термінал.

gcc 7.3.0 – компілятор коду.

## 4. Компіляція і запуск програми

Скомпілювати програму можна за допомогою команди:

g++ -o application main.cpp classes.cpp

Або

g++ -o application main.cpp classes.cpp –Wall

Для відображення усіх попереджень при компілюванні. Для запуску необхідно виконати наступну команду:

./application

## 5. Вхідні й вихідні дані

Вхідні дані можуть бути представлені у вигляді файлу, згенерованого програмою самостійно, або методом послідовного створення будь-якої кількості елементів архівів за допомогою спеціального пункту меню.

Перший варіант виводу інформації – табличка з даними про певний елемент архіву (рис. 5.1).

Другий варіант – вивід під час пошуку елементів за певним параметром, в прикладі – за ключовим словом файлу. (рис. 5.2).

Третій варіант – вивід під час запиту щодо статистичної інформації (рис. 5.3).

Останній, четвертий варіант – запис даних у файл у компактному режимі, тобто режимі, зручному для подальшого зчитування інформації з файлу (рис. 5.4).

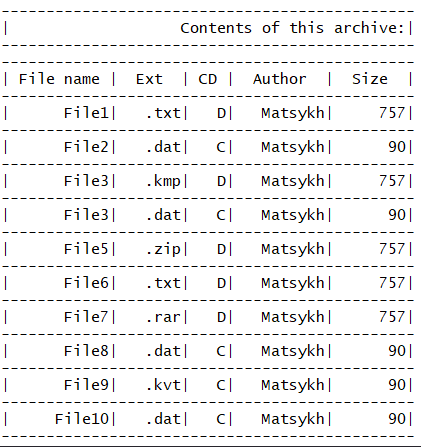


Рисунок 5.1 – Вивід у вигляді таблиці

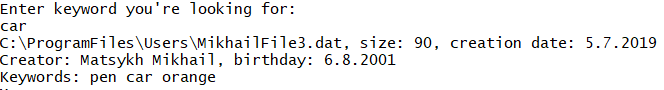


Рисунок 5.2 – Вивід даних після пошуку



Рисунок 5.3 – Вивід статистичної характеристики

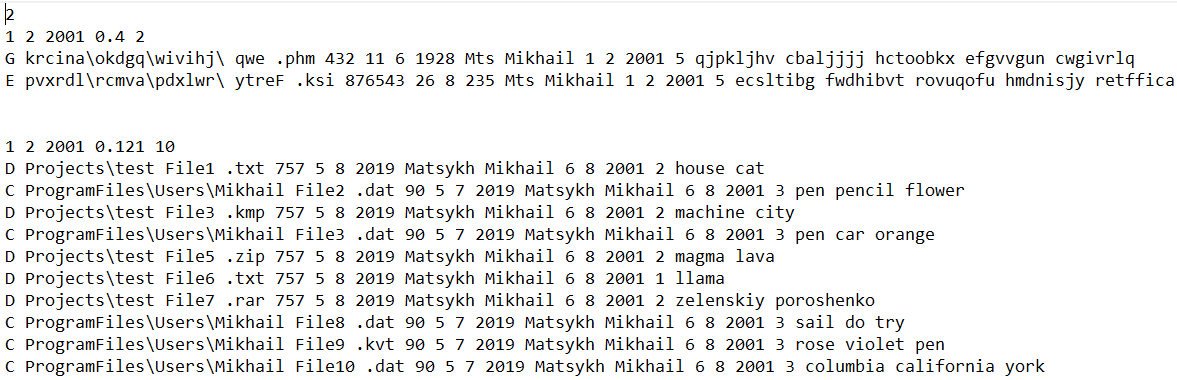


Рисунок 5.4 – Формат збереження даних у файлі