**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»**

Мациха М. В.

(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**

Мациху Міхаілу Валерійовичу

(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 \_\_\_\_\_\_\_\_\_ Мацих М. В.**

( підпис ) (прізвище та ініціали)

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

( підпис ) (прізвище та ініціали)

АНОТАЦІЯ

Програма описує інформаційну модель для змісту каталогу, а саме створення та подальша робота з об’єктами, що характеризують текстові файли, елементи архіву, 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**

Program describes model of folder contents, for example creation and use of objects that describe text files, archive elements, 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 in the note.

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\_TІ82108 3 КР 81-1

Розробив

Мацих М. В.

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

Відомо, що технології зараз розвинуті більше, ніж будь-коли в історії людства. Також очевидно, що вже завтра вони стануть ще більш розвинутими. Рушійною силою прогресу на сьогоднішній день є технології, для створення й використання яких є необхідним процес програмування.

Основи програмування беруть свій початок ще в середині двадцятого століття, і з розвитком почали з’являтися нові мови програмування, що використовували нові парадигми програмування й мали певні особливості. Ні для кого не є секретом той факт, що об’єктно-орієнтоване програмування є найрозповсюдженішою парадигмою програмування, якої й сьогодні дотримуються майже всі новітні мови програмування.

С++, в свою чергу, безумовно є найпотужнішим інструментом, що тільки може бути використаний для створення програмного забезпечення. Проте за якість доводиться платити, а тому мова С++ має багато особливостей та підводних каменів, вивчаючи які розробник програмного забезпечення отримує можливість користуватися усіма сильними сторонами цієї мови програмування.

Увесь світ користується програмним забезпеченням, і в кожній галузі своя інформаційна модель, свої потреби до виконуваних операцій та результату. І саме тому вміння створювати моделі об’єктів реального світу є обов’язковим для будь-якого розробника програмного забезпечення.

Основна частина

# 1. Теоретична частина

## 1.1 Відомості про об’єктно-орієнтоване програмування

Об’єктно-орієнтоване програмування (надалі – ООП) – найрозповсюдженіша парадигима програмування, що полягає в об’єднанні таких елементів коду, як змінні й методи в певні класи, створені користувачем для полегшення читання програмного коду та зменшення його об’єму.

Вперше парадигма ООП була представлена в мові програмування Simula (укр. Симула) в 1960-ті роки. Попри те, що це стало новизною в світі програмного забезпечення, мова, як і сама парадигма стали набирати популярності лише в 1990-ті роки разом із стрімким технологічним розвитком. Мова програмування Симула є менш відомою за мову Smalltalk хоча б через той факт, що розробником мови Smalltalk став творець парадигми ООП Алан Кей. На його думку справедливі наступні твердження щодо об’єктно-орієнтованого програмування:

Усе є об’єктами.

Програма побудована на взаємодії об’єктів між собою, тобто якщо об’єкт потребує певних даних, їх обчислення буде зроблене іншим спеціальним об’єктом.

Будь-який об’єкт може складатися з інших об’єктів.

Об’єкт – певна одиниця, що представляє собою клас.

Клас зберігає в собі повний функціонал об’єкта, тому всі об’єкти одного й того самого класу мають однаковий функціонал.

Класи організовані в структуру – ієрархію успадкування. Будь-який клас, розташований в цій ієрархії, має повний доступ до елементів класів, що розташовані вище.

Тепер дамо повне означення двом основним термінам ООП:

Клас – це спеціальна конструкція, що використовується для групування змінних та методів. При цьому, згідно з термінологією ООП, глобальні змінні класу називаються полями даних, а члени-функції називають методами класу.

Об’єкт – це створений та ініціалізований екземпляр класу.

Основу ООП складають три головні концепції: інкапсуляція, поліморфізм, та успадковування.

Інкапсуляція – регулювання доступу до елементів класу задля полегшення читання програмного коду та підвищення його безпеки.

В мові C++ існують наступні три специфікатори доступу:

* private (закритий);
* public (відкритий);
* protected (захищений).

Елементи, позначені специфікатором private можуть бути викликані лише в цьому класі. Зазвичай доступ до цих елементів отримується завдяки public методам.

Специфікатор public надає доступ як з цього класу, так і з його об’єктів та об’єктів інших класів.

Специфікатор protected має той самий функціонал, що і private, та додаткові характеристики, що будуть розглянуті пізніше.

Приклад:

class MyClass{ // Створення класу

private:

int a;

public:

int b;

};

int main(){

MyClass obj; // Створення об’єкту класу

obj.a; // Помилка

obj.b; // Помилка відсутня

return 0;

}

Поліморфізм – це здатність використовувати спільний інтерфейс для обробки даних різних типів.

В мові програмування С++ поліморфізм виражено в можливості перевантаження методів, операторів тощо. Так, в мові програмування, що не дотримується концепції поліморфізму не є припустимим існування двох методів addOne(…), навіть якщо один із них оперує типом даних integer, тобто цілочисленним, а інший – float, тобто дійсним.

Найчастіше поліморфізм використовується для перевантаження операторів, що надає змогу розробнику програмного забезпечення вивести взаємодію між об’єктами на новий рівень.

Саме тому припустимими є наступні конструкції:

1 + 2;

0.78 + 12.92;

“Hello “ + “world!”;

Унаслідування – це можливість створення нових класів на основі вже існуючих, при тому, що функціонал базових класів повністю передається похідним.

Прикладів унаслідування достатньо в реальному світі: електросамокат повністю зберігає функціонал звичайного самоката, та накладає новий функціонал поверх існуючого.

У механізмі успадковування мови програмування С++ важливу роль відіграють специфікатори доступу, що були розглянуті раніше: таким чином, private успадковування закриває для доступу всі поля базового класу, навіть ті, що мають специфікатор public.

public успадковування залишає усі елементи зі своїми специфікаторами, не змінюючи їх прав доступу.

protected успадковування залишає доступ до protected та public елементів, але тільки з самого класу.

Мова програмування С++ також підтримує множинне успадковування, що дозволяє одному класу успадковувати відразу від декількох базових.

Приклад успадковування:

class Derived : public Base{…

class Derived : private Base1, public Base2{…

class Derived : \*type\* Base1, \*type\* Base2, …, \*type\* BaseN{…

## 1.2 Шаблони в мові програмування С++

Велику роль в програмуванні на мові С++ відіграють шаблони. Вони дають змогу не писати один й той самий програмний код по декілька разів для кожного типу параметра функції або класу.

Наприклад, якщо необхідно написати функцію, що повертає найбільше з трьох чисел, де числа можуть бути будь-яким типом даних, замість написання трьох й більше функцій буде достатнім написати й реалізувати одну наступну:

template <typename Type>

Type findMaxValue(Type num\_1, Type num\_2, Type num\_3);

При цьому правильними будуть усі наступні використання цієї функції:

int a1 = 1, b1 = 2, c1 = 3, res1;

float a2 = 1.5, b2 = 7.9, c2 = 0.1, res2;

double a3 = 15, b3 = -0.3, c3 = 1.102, res3;

res1 = findMaxValue(a1, b1, c1);

res2 = findMaxValue(a2, b2, c2);

res3 = findMaxValee(a3, b3, c3);

Більше того, за потреби можна вказати тип шаблону вручну, що необхідно, наприклад, за умови неоднозначності:

res3 = findMaxValue<double>(a1, a2, a3);

Розглянемо синтаксис оголошення шаблонної функції:

* template – початок опису шаблонної функції.
* typename – вказує на те, що під шаблон може бути підставлений тип даних.
* Type – назва типу, який тепер може бути використаним усередині цієї функції.

Тип може мати будь-яку назву, а не лише Type, а замість ключового слова typename може бути використане інше ключове слово. Найбільш використовуваними є ключові слова typename та class.

Різниця між ключовими словами typename і class полягає в тому, що якщо typename дозволяє використовувати будь-який тип даних у якості шаблону, то class змушує використовувати лише написані програмістом класи.

Наприклад, для функції:

**template <class Class>**

**double calculateAverage(Class object);**

Правильним буде її наступний виклик:

**Square sqr(…);**

**double res = calculateAverage(sqr);**

При чому виклик:

**float f = 3.7;**

**double res = calculateAverage(f);**

Збудить помилку під час компілювання програми. Підіб’ємо підсумок:

Шаблонною функцією називається функція, що може приймати в якості параметрів змінні різних типів та виконувати для них однакові дії.

Другий спосіб використання шаблонів у С++ – це написання шаблонних класів.

Нехай необхідно написати клас-контейнер, що буде містити динамічний масив елементів певного типу. Тоді опис цього класу (а саме його частина, що нас цікавить в даний момент) буде виглядати наступним чином:

**template <typename Type>**

**class Array{**

**private:**

**Type \*array;**

**unsigned int size;**

**public:**

**…**

**};**

А створити об’єкт цього класу можна наступним чином:

**Array <int> int\_array;**

**Array <float> f\_array;**

**Array <MyClass> custom\_array;**

Як бачимо, синтаксис залишається той самий, що й для створення шаблонних функцій. Єдиною різницею є те, що за відсутності шаблонного конструктору класу тип повинен бути явно вказаний (наприклад, <int>). Дамо визначення шаблонному класу.

Шаблонним класом називається клас, який може містити елементи шаблонного типу. Якщо клас не містить шаблонних змінних, проте містить шаблонні функції, це не робить його шаблонним класом, а лише класом із шаблонними функціями.

Важливо розуміти те, що написання шаблонних функцій та класів спрощує читання коду й зменшує його об’єм, проте не зменшу’є об’єм компільованого коду. Причина проста – компілятор розписує усі шаблонні функції та класи на необхідні для роботи програми вже під час компіляції.

# **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)

## 2.4 Обробка виключних ситуацій

Під час виконання програма може отримувати неправильні дані, виконувати неможливі операції, стикатися з помилками під час виділення пам’яті тощо. І якщо у випадку помилкового вводу програма сама запросить користувача повторити спробу (див. пункт 2.5), то для деяких випадків необхідно ввести механізм обробки виключних ситуацій.

Загальний синтаксис обробки виключень наступний: місце можливої появи виключень поміщається в блок try{…}, а місце обробки певної виключної ситуації – в блок catch(…){…}. Програма передає керування з блоку try в блок catch у випадку появи в блоці try ключового слова throw, що супроводжується чим завгодно від деякого числа до спеціального класу.

Таким чином, програма з блоком для обробки виключних ситуацій може мати наступний вигляд:

#include <iostream>

int main(){

int a = 5, b, c;

std::cout << “Enter integer: “;

std::cin >> b;

try{

if(!b) throw 0;

c = a / b;

}catch(int x){

std::cout << “Error #” << x << “ occurred!\n”;

}

return 0;

}

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

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

Кожне з виключень використовується у програмі за певних умов: помилка при виділенні пам’яті збудить AllocationException, помилка виконання якоїсь операції над об’єктом збудить OperationException, а неправильний індекс збудить OutOfBounds.

За причини використання саме ієрархії класів виключень немає необхідності писати три різні блоки catch для кожного типу виключення (хоча іноді це й потрібно), достатнім буде наступний запис:

}catch(Exception &e){

cout << “Exception occurred!\n” << e.what() << endl;

}

Де функція e.what() є перевантаженою функцією, що повертає повідомлення про помилку у вигляді const char \*.

## 2.5 Обмеження певних даних у програмі

Хоча користувач програми повинен бути завжди правий, у деяких випадках користувач просто не може знати те, що насправді він неправий.

Для отримання вводу користувача були використані чотири невеликі функції, які будуть запрошувати користувача ввести певні дані до тих пір, поки користувач не введе їх коректно. Ці функції мають наступні прототипи:

* char getSymbol(const char \*prompt);
* std::string getWord(const char \*prompt, bool digits = true);
* long long getInt(const char \*prompt);
* double getFloat(const char \*prompt);

Де const char \*prompt – деяка строка-запрошення, у якій сказано, що саме повинен ввести користувач.

Параметр digits функції getWord() дозволяє чи забороняє використання цифр у зчитуваному слові. Якщо ім’я людини не може містити цифр, то для назви файлу цієї заборони вже немає.

## 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 перегрузка методу handleCommand void handleCommnad(wstring inputData)

Вивід меню оганізовано за допомогою классу Menu, який було унаслідувано від классу WMenu::WonderMenu і в ньому перегружено 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

## 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:

# 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.

Розробити графічний інтерфейс користувача можна за допомогою такої бібліотеки як QT разом із програмним забезпеченням QT Creator, використовуючи мову програмування С++.

Також можна збільшити ієрархічне дерево виключень та додати більше особливостей у кожен окремий клас. Наприклад, AllocationException може намагатися знайти адресу в пам’яті, з якою пов’язана помилка. Це могло б допомогти програмі в роботі на обчислювальних машинах з малими об’ємами пам’яті, таких, як мікропроцесори.

У випадку розширення головної ієрархії програми та появою там нових класів, що будуть зберігати ті чи інші динамічні масиви даних правильним й зручним буде створення шаблонного класу Container, який буде містити в собі необхідний функціонал для роботи з динамічними масивами різних типів, а інші класи на кшталт “Елемент архіву” будуть наслідуватися від Container.

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.

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

Список використаних джерел

1. Г. Буч. Объектно – ориентированный анализ и проектирование с примерами приложений на С++, 2-е изд./ Пер. с англ. – М.: "Издательство Бином". 1998 г. – 560 с., ил.
2. Павловская Т.А., Щупак Ю.А. С++. Объектно-ориентированное программирование: Практикум. – СПб.: Питер,2004 -288 с.
3. Павловская Т.А.С/С++ Программирование на языке высокого уровня. – СПб.: Питер,2007 -461 с.
4. Х.М.Дейтел., П.Дж.Дейтел. Как программировать на Си++. М .: ЗАО БИНОМ, 1999, 1000 с.
5. Карпов Б., Баранов Т. С++: специальный практикум. – СПб.: Питер,2000 -480 с.
6. Лаптев В.В. C++. Экспресс-курс. – СПб.: БХВ-Петербург, 2004. – 512 с.: ил.
7. Лафоре Р. Объектно-ориентированное программирование в С++. Классика Computer Science. 4 изд.. – СПб.: Питер, 2003.

Додаток 1. Текст програми

1. classes.cpp

#include "classes.h"

#include <iostream>

#include <cstdlib>

using namespace std;

bool Date::operator >(const Date &other) const{

if(year < other.year)

return false;

else if(year > other.year)

return true;

if(month < other.month)

return false;

else if(month > other.month)

return true;

if(day > other.day)

return true;

return false;

}

bool Date::operator <(const Date &other) const{

if(year < other.year)

return true;

if(month < other.month)

return true;

if(day < other.day)

return true;

return false;

}

bool Date::operator >=(const Date &other) const{

if(\*this > other || \*this == other)

return true;

return false;

}

bool Date::operator <=(const Date &other) const{

if(\*this < other || \*this == other)

return true;

return false;

}

bool Date::operator ==(const Date &other) const{

if(year == other.year && month == other.month && day == other.day)

return true;

return false;

}

Date Date::operator -(const Date &other) const{

Date res;

if(other > \*this)

throw OperationException("Second Date should be lower than first!");

int res\_day = day - other.day;

int res\_month = month - other.month;

res.year = year - other.year;

if(res\_day < 0){

res\_month--;

res\_day = 31 - (other.day - day);

}

if(res\_month < 0){

res.year--;

res\_month = 12 - (other.month - month);

}

res.day = res\_day;

res.month = res\_month;

return res;

}

Date Date::operator +(const Date &other) const{

Date res;

int res\_day = day + other.day;

int res\_month = month + other.month;

int res\_year = year + other.year;

if(res\_day > 31){

res\_month++;

res\_day -= 31;

}

if(res\_month > 12){

res\_year++;

res\_month -= 12;

}

if(res\_year > 2019)

throw OperationException("Resulting year is greater than 2019!");

res.day = res\_day;

res.month = res\_month;

res.year = res\_year;

return res;

}

Date::operator bool() const{

return (day && month && year);

}

void Date::operator =(const Date &other){

unsigned int d, m;

int y;

other.getDate(d, m ,y);

setDate(d, m, y);

}

Person::Person(const char \*second, const char \*first, Date date){

surname = new char[strlen(second) + 1];

name = new char[strlen(first) + 1];

if(!(surname && name))

throw AllocationException("Allocation error!");

strcpy(surname, second);

strcpy(name, first);

setDate(date);

}

Person::~Person(){

delete surname;

delete name;

}

bool Person::operator >(const Person &other) const{

return getBirthday() > other.getBirthday();

}

bool Person::operator <(const Person &other) const{

return getBirthday() < other.getBirthday();

}

bool Person::operator >=(const Person &other) const{

return getBirthday() >= other.getBirthday();

}

bool Person::operator <=(const Person &other) const{

return getBirthday() <= other.getBirthday();

}

bool Person::operator ==(const Person &other) const{

return getBirthday() == other.getBirthday();

}

File::File(const char \*n, const char \*ext, Date date, unsigned long long s, char disk, const char \*path\_to\_file){

name = new char[strlen(n) + 1];

extention = new char[strlen(ext) + 1];

Placement.path = new char[strlen(path\_to\_file) + 1];

if(!(name && extention && Placement.path))

throw AllocationException("Allocation error!");

strcpy(name, n);

strcpy(extention, ext);

strcpy(Placement.path, path\_to\_file);

setDate(date);

size = s;

Placement.cd = disk;

}

File::~File(){

delete name;

delete extention;

delete Placement.path;

}

void File::operator =(const File &other){

name = new char[strlen(other.name) + 1];

extention = new char[strlen(other.extention) + 1];

Placement.path = new char[strlen(other.Placement.path) + 1];

if(!(name && extention && Placement.path))

throw AllocationException("Allocation error!");

strcpy(name, other.name);

strcpy(extention, other.extention);

strcpy(Placement.path, other.Placement.path);

setDate(other.getCreationDate());

size = other.size;

Placement.cd = other.Placement.cd;

}

bool File::operator ==(const File &other){

if(!strcmp(name, other.name))

if(!strcmp(extention, other.extention))

if(Placement.cd == other.Placement.cd)

if(!strcmp(Placement.path, other.Placement.path))

if(size == other.size)

return true;

return false;

}

bool File::operator >(const File &other){

return size > other.size;

}

bool File::operator >=(const File &other){

return size >= other.size;

}

bool File::operator <(const File &other){

return size < other.size;

}

bool File::operator <=(const File &other){

return size <= other.size;

}

TextFile::TextFile(const char \*n, const char \*ext, Date date, unsigned long long s, char disk, const char \*path\_to\_file, Person \*author, char \*\*words, unsigned int words\_s) : File(n, ext, date, s, disk, path\_to\_file){

creator = author;

keywords = new char\*[words\_s];

if(!keywords)

throw AllocationException("Allocation error!");

for(unsigned int i = 0; i < words\_s; i++){

keywords[i] = new char[strlen(words[i]) + 1];

if(!keywords[i])

throw AllocationException("Allocation error!");

strcpy(keywords[i], words[i]);

}

size = words\_s;

}

TextFile::~TextFile(){

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

delete keywords[i];

}

delete keywords;

}

ArchiveElement::ArchiveElement(Date creation, double arc){

while(arc <= 0.0 || arc > 1.0){

cout << "Archivation level should be 0 < lvl <= 1!\nTry again: ";

char input[20];

cin >> input;

cin.ignore(100, '\n');

arc = atof(input);

}

archivation = arc;

top = 0;

size = 1;

setDate(creation);

files = (TextFile \*\*)malloc(sizeof(TextFile \*));

if(files == nullptr)

throw AllocationException("Allocation error!");

}

ArchiveElement::~ArchiveElement(){

for(unsigned int i = 0; i < top; i++)

delete files[i];

free(files);

}

void ArchiveElement::addFile(const TextFile file){

if(top == size){

size \*= 2;

files = (TextFile \*\*)realloc(files, sizeof(TextFile \*) \* size);

if(files == nullptr)

throw AllocationException("Allocation error!");

}

files[top++] = new TextFile(file);

}

TextFile \*ArchiveElement::operator [](unsigned int i){

if(i >= top)

throw OutOfBounds("Index out of range!");

return files[i];

}

FolderContent::FolderContent(){

top = 0;

size = 1;

array = (ArchiveElement \*\*)malloc(sizeof(ArchiveElement \*));

if(array == nullptr)

throw AllocationException("Allocation error!");

}

FolderContent::~FolderContent(){

for(unsigned int i = 0; i < top; i++)

delete array[i];

free(array);

}

ArchiveElement \*FolderContent::operator [](unsigned int i){

if(i >= top)

throw OutOfBounds("Index out of range!");

return array[i];

}

void FolderContent::addElement(ArchiveElement element){

if(top == size){

size \*= 2;

array = (ArchiveElement \*\*)realloc(array, sizeof(ArchiveElement \*) \* size);

if(array == nullptr)

throw AllocationException("Allocation error!");

}

array[top++] = new ArchiveElement(element);

}

2. classes.h

#pragma once

#include <cstring>

#include <iostream>

#include <exception>

class Date{

int year;

unsigned int month;

unsigned int day;

public:

Date();

bool setDate(unsigned int d, unsigned int m, int y);

void setDate(Date other);

bool setYear(int y);

bool setMonth(unsigned int m);

bool setDay(unsigned int d);

void getDate(unsigned int &d, unsigned int &m, int &y) const;

virtual void Show(std::ostream &stream = std::cout, bool compact = false) const;

bool operator >(const Date &other) const;

bool operator <(const Date &other) const;

bool operator >=(const Date &other) const;

bool operator <=(const Date &other) const;

bool operator ==(const Date &other) const;

Date operator -(const Date &other) const;

Date operator +(const Date &other) const;

void operator =(const Date &other);

operator bool() const;

};

class Person : public Date{

char \*surname;

char \*name;

public:

Person(const char \*second, const char \*first, Date date);

~Person();

Person(const Person &other);

char \*getFirst() const;

char \*getSecond() const;

Date getBirthday() const;

void operator =(const Person &other);

bool operator >(const Person &other) const;

bool operator <(const Person &other) const;

bool operator >=(const Person &other) const;

bool operator <=(const Person &other) const;

bool operator ==(const Person &other) const;

virtual void Show(std::ostream &stream = std::cout, bool compact = false);

};

class File : public Date{

char \*name;

char \*extention;

unsigned long long size;

struct{

char cd;

char \*path;

} Placement;

public:

File(const char \*n, const char \*ext, Date date, unsigned long long s, char disk, const char \*path\_to\_file);

~File();

File(const File &other);

char \*getName() const;

char \*getExtention() const;

Date getCreationDate() const;

unsigned long long getFileSize();

char getCD();

char \*getPath() const;

char \*getFullPath() const;

void setName(const char \*str);

void setExtention(const char \*str);

void setFileSize(unsigned long long num);

void setCD(const char str);

void setPath(const char \*str);

void operator =(const File &other);

bool operator ==(const File &other);

bool operator >(const File &other);

bool operator >=(const File &other);

bool operator <(const File &other);

bool operator <=(const File &other);

virtual void Show(std::ostream &stream = std::cout, bool compact = false);

};

class TextFile : public File{

Person \*creator;

char \*\*keywords;

unsigned int size;

public:

TextFile(const char \*n, const char \*ext, Date date, unsigned long long s, char disk, const char \*path\_to\_file, Person \*author, char \*\*words, unsigned int words\_s);

~TextFile();

TextFile(const TextFile &other);

Person \*getCreator() const;

char \*getKeyword(unsigned int i) const;

unsigned int getKeywordsSize();

virtual void Show(std::ostream &stream = std::cout, bool compact = false);

};

class ArchiveElement : public Date{

TextFile \*\*files;

unsigned int size;

unsigned int top;

TextFile \*get(unsigned int i) const;

double archivation;

public:

ArchiveElement(Date creation, double arc);

~ArchiveElement();

ArchiveElement(const ArchiveElement &other);

unsigned int getArchiveSize();

double getArchivationLevel();

Date getArchiveCreationDate() const;

void addFile(TextFile file);

TextFile \*operator [](unsigned int i);

void setArchiveCreationDate(Date newDate);

unsigned int getTop();

virtual void Show(std::ostream &stream = std::cout, bool compact = false);

};

class FolderContent{

ArchiveElement \*\*array;

unsigned int size;

unsigned int top;

ArchiveElement \*get(unsigned int i) const;

public:

FolderContent();

~FolderContent();

FolderContent(const FolderContent &other);

void addElement(ArchiveElement element);

ArchiveElement \*operator [](unsigned int i);

unsigned int getTop();

virtual void Show(std::ostream &stream = std::cout, bool compact = false);

TextFile \*FindByName(const char \*str);

TextFile \*FindBySize(unsigned long long s);

TextFile \*FindByKeyword(const char \*str);

};

class Exception : public std::exception{

char \*message;

public:

Exception(const char \*msg);

virtual const char \*what() const noexcept override;

~Exception();

};

class AllocationException : public Exception{

public:

AllocationException(const char \*msg) : Exception(msg) {}

};

class OutOfBounds : public Exception{

public:

OutOfBounds(const char \*msg) : Exception(msg) {}

};

class OperationException : public Exception{

public:

OperationException(const char \*msg) : Exception(msg) {}

};

3. main.cpp

#include "classes.h"

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cstring>

#include <sstream>

#include <fstream>

#include <iomanip>

using namespace std;

template <typename Type>

Type calcuateFiles(FolderContent &folder){

Type res = 0.0;

for(unsigned int arc = 0; arc < folder.getTop(); arc++){

res += folder[arc]->getTop();

}

res /= folder.getTop();

return res;

}

template <typename Type>

Type calcuateArchivation(FolderContent &folder){

Type res = 0.0;

for(unsigned int arc = 0; arc < folder.getTop(); arc++){

res += folder[arc]->getArchivationLevel();

}

res /= folder.getTop();

return res;

}

void SaveFolderBinary(FolderContent &folder, string fname="binary"){

stringstream strstream;

folder.Show(strstream, true);

ofstream fout(fname + ".txt", ios::binary | ios::out);

fout.write(strstream.str().c\_str(), strstream.str().length());

fout.close();

}

ArchiveElement LoadArchiveFromBinary(unsigned int index, Person \*creator, string fname = "binary"){

ifstream fin(fname + ".txt");

unsigned int arc\_num;

fin >> arc\_num;

if(index >= arc\_num)

throw OutOfBounds("Array index out of range!");

for(unsigned int archive = 0; archive < index; archive++){

string str\_skip;

unsigned int skip\_num;

for(int i = 0; i < 4; i++) fin >> str\_skip;

fin >> skip\_num;

for(unsigned int i = 0; i < skip\_num; i++){

unsigned int skip\_num\_inside;

for(int i = 0; i < 13; i++) fin >> str\_skip;

fin >> skip\_num\_inside;

for(unsigned int i = 0; i < skip\_num\_inside; i++) fin >> str\_skip;

}

}

unsigned int d, m, files\_num;

int y;

double archivation;

fin >> d >> m >> y >> archivation >> files\_num;

Date date;

date.setDate(d, m, y);

ArchiveElement res(date, archivation);

for(unsigned int i = 0; i < files\_num; i++){

string cd, path, name, ext, sn, fn, kw;

unsigned long long size;

unsigned int d1, d2, m1, m2, kw\_size;

int y1, y2;

fin >> cd >> path >> name >> ext >> size >> d1 >> m1 >> y1 >> sn >> fn >> d2 >> m2 >> y2 >> kw\_size;

char \*\*keywords = new char\* [kw\_size];

for(unsigned int j = 0; j < kw\_size; j++){

fin >> kw;

keywords[j] = new char[kw.length() + 1];

strcpy(keywords[j], kw.c\_str());

}

Date f\_date, p\_date;

f\_date.setDate(d1, m1, y1);

p\_date.setDate(d2, m2, y2);

creator = new Person(sn.c\_str(), fn.c\_str(), p\_date);

TextFile file(name.c\_str(), ext.c\_str(), f\_date, size, cd.c\_str()[0], path.c\_str(), creator, keywords, kw\_size);

res.addFile(file);

for(unsigned int j = 0; j < kw\_size; j++){

delete keywords[j];

}

delete keywords;

}

fin.close();

return res;

}

int main(){

srand(time(nullptr));

Date someday;

Person \*author = nullptr;

FolderContent folder;

char word[100];

cout << "Menu:\n1. Input\n2. Output\n3. Save to file\n4. Load from file\n5. Search in array\n6. Show stitistical info\nq. Quit\n";

char ch = getSymbol("Please, make your choice!");

while(ch != 'q' && ch != 'Q'){

try{

switch(ch){

case '1':

{

cout << "Input menu:\n1. Input date\n2. Input author info\n3. Create new archive\nq. Back\n";

ch = getSymbol("Please, make your choice!");

while (ch != 'q' && ch != 'Q')

{

switch(ch){

case '1':

{

char d[10], m[10], y[10];

do{

cout << "d m y: ";

cin >> d >> m >> y;

cin.ignore(100, '\n');

}while(!someday.setDate(atoi(d), atoi(m), atoi(y)));

break;

}

case '2':

{

if(author != nullptr){

cout << "Author already exists!\n";

break;

}

string fname = getWord("First name:", false);

string sname = getWord("Second name:", false);

char d[10], m[10], y[10];

do{

cout << "Birthday (d m y): ";

cin >> d >> m >> y;

cin.ignore(100, '\n');

}while(!someday.setDate(atoi(d), atoi(m), atoi(y)));

author = new Person(sname.c\_str(), fname.c\_str(), someday);

break;

}

case '3':

{

if(author == nullptr){

cout << "Create author first!\n";

ch = '2';

break;

}

if(!someday){

char d[10], m[10], y[10];

do{

cout << "Creation (d m y): ";

cin >> d >> m >> y;

cin.ignore(100, '\n');

}while(!someday.setDate(atoi(d), atoi(m), atoi(y)));

}

double lvl = getFloat("Enter archivation level (0 < lvl <= 1):");

ArchiveElement arc(someday, lvl);

do{

cout << "a. Add file\nq. Stop and save\n";

do{

ch = getSymbol("Please, make your choice!");

}while(ch != 'a' && ch != 'A' && ch != 'q' && ch != 'Q');

if(ch == 'a' || ch == 'A'){

do{

cout << "1. Generate cd, path, extention and keywords automatically\n2. Manual input\n";

cin >> word;

cin.ignore(100, '\n');

}while(word[0] != '1' && word[0] != '2' && strlen(word) != 1);

string name = getWord("Enter file name:");

unsigned long long size = getInt("Enter file size:");

if(word[0] == '1'){

TextFile file(gen\_file(author));

file.setName(name.c\_str());

file.setFileSize(size);

arc.addFile(file);

}else{

char cd = getSymbol("Enter cd (one symbol):");

string path = getWord("Enter file path:");

string ext = getWord("Enter file extention:");

int kw\_num = getInt("Enter number of keywords:");

char \*\*kw = new char \* [kw\_num];

for(int i = 0; i < kw\_num; i++){

string w = getWord("Enter keyword:");

kw[i] = new char[w.length() + 1];

strcpy(kw[i], w.c\_str());

}

TextFile file(name.c\_str(), ext.c\_str(), someday, size, cd, path.c\_str(), author, kw, kw\_num);

file.setExtention(ext.c\_str());

for(int i = 0; i < kw\_num; i++){

delete kw[i];

}

delete kw;

arc.addFile(file);

}

}

}while(ch == 'a' || ch == 'A');

folder.addElement(arc);

break;

}

}

cout << "Input menu:\n1. Input date\n2. Input author info\n3. Create new archive\nq. Back\n";

ch = getSymbol("Please, enter your choice!");

}

break;

}

case '2':

{

if(!folder.getTop()){

cout << "Nothing to print!\n";

break;

}

cout << "Please select archive number (1 - " << folder.getTop() << ") to print table: ";

unsigned int res;

do{

res = getInt("Make a choice:");

}while(res > folder.getTop());

printTable(\*folder[res - 1]);

cout << endl;

break;

}

case '3':

{

string fname = getWord("Enter file name, where to save folder info:");

char choice;

do{

choice = getSymbol("Enter \'b\' to write binary, \'t\' to write text-like:");

}while(choice != 'b' && choice != 't');

if(choice == 'b'){

SaveFolderBinary(folder, fname);

}else{

ofstream fout(fname + ".txt");

folder.Show(fout, true);

fout.close();

}

fstream fout("files.dat", ios::out | ios::app);

fout << fname << ".txt\n";

break;

}

case '4':

{

Person \*cr;

ifstream fin("files.dat");

string info;

string full("");

while(!fin.eof()){

string tmp;

fin >> tmp;

if(tmp != info){

info = tmp;

full += info;

cout << info << endl;

}

}

string fname = getWord("Enter file name from list above without .txt in it:");

while(strstr(full.c\_str(), fname.c\_str()) == nullptr){

fname = getWord("Enter file name from list above without .txt in it:");

}

unsigned int arc\_num = ArchivesToLoad(fname);

for(unsigned int i = 0; i < arc\_num; i++){

ArchiveElement elem(LoadArchiveFromBinary(i, cr, fname));

folder.addElement(elem);

}

break;

}

case '5':

{

char choice;

do{

choice = getSymbol("Find by:\n1. Name\n2. Size\n3. Keyword\n");

}while(choice != '1' && choice != '2' && choice != '3');

TextFile \*res;

if(choice == '1'){

string str = getWord("Enter name you\'re looking for:");

res = folder.FindByName(str.c\_str());

}else if(choice == '2'){

unsigned long long size = getInt("Enter size you\'re looking for:");

res = folder.FindBySize(size);

}else{

string str = getWord("Enter keyword you\'re looking for:");

res = folder.FindByKeyword(str.c\_str());

}

if(res == nullptr){

cout << "No match!\n";

}else{

res->Show();

cout << endl;

}

break;

}

case '6':

{

if(!folder.getTop()){

cout << "Nothing to print!\n";

break;

}

cout << "Avarage quantity of files per archive in folder: " << calcuateFiles<double>(folder) << endl;

cout << "Average archivation level per archive in folder: " << calcuateArchivation<double>(folder) << endl;

break;

}

}

}catch(Exception &e){

cout << "An error occured: " << e.what() << endl;

}

cout << "Menu:\n1. Input\n2. Output\n3. Save to file\n4. Load from file\n5. Search in array\n6. Show stitistical info\nq. Quit\n";

ch = getSymbol("Please, enter your choice!");

}

if(author != nullptr)

delete author;

return 0;

}

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.

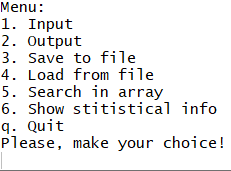


Рисунок 1.1 – Меню програми

## **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.

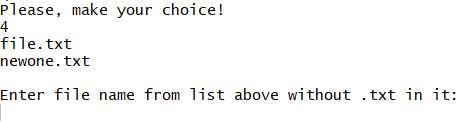


Рисунок 2.1 – Пропонування програмою можливих файлів

## **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).

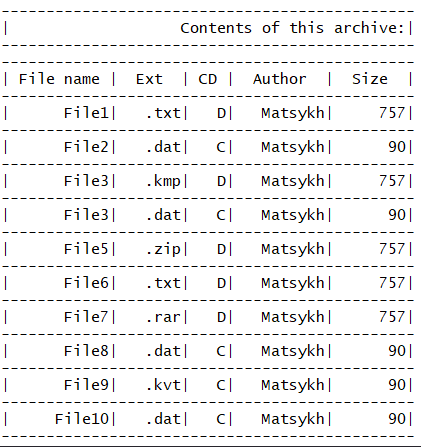


Рисунок 5.1 – Вивід у вигляді таблиці

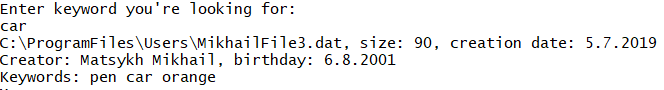


Рисунок 5.2 – Вивід даних після пошуку



Рисунок 5.3 – Вивід статистичної характеристики

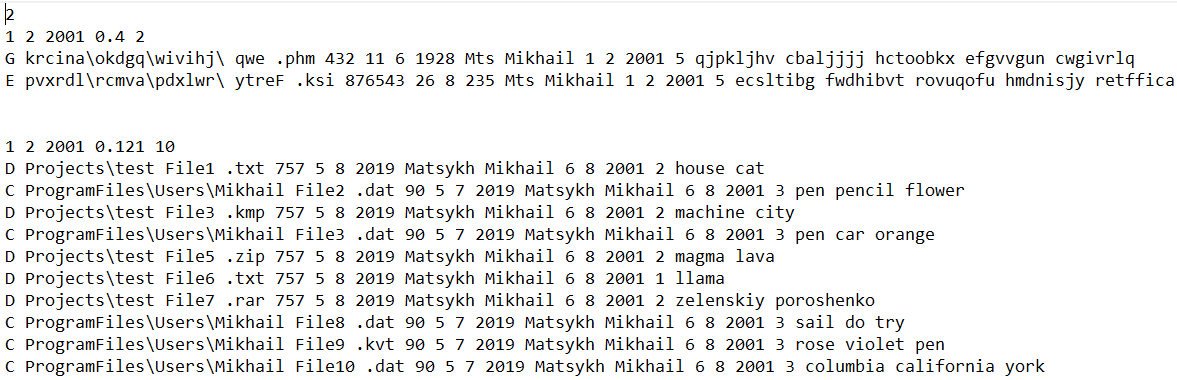


Рисунок 5.4 – Формат збереження даних у файлі