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 - 2»

(назва дисципліни)

на тему:

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

Студента групи ТВ-61

спеціальність **121 «Інженерія програмного забезпечення»**

спеціалізація **Програмне забезпечення розподілених систем**

Петровського О.Г.

(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 - 2017 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 розподілених систем»***

(шифр і назва)

ЗАВДАННЯ

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Петровському Олександру Григоровичу

(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 – 22 грудня 2017 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 – 30 жовтня 2017 р.

#### КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсової роботи | Строк виконання етапів роботи | Примітка |
|  | Побудова ієрархічної моделі класів | 17.11.2017 |  |
|  | Розробка та застосування інструментальних засобів обробки даних | 08.12.2017 |  |
|  | Написання розрахунково-пояснювальної записки | 22.12.2017 |  |

**Студент \_\_\_\_\_\_\_\_\_ Петровський О.Г.**

( підпис ) (прізвище та ініціали)

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

( підпис ) (прізвище та ініціали)

# ЗМІСТ

[ЗМІСТ 3](#_Toc501741311)

[АНОТАЦІЯ 4](#_Toc501741312)

[ВСТУП 5](#_Toc501741313)

[1.ТЕОРЕТИЧНІ ВІДОМОСТІ 8](#_Toc501741314)

[*1.1.Принципи Об’єктно-орієнтованого програмування* 8](#_Toc501741315)

[*1.4.Перевантаження операцій* 10](#_Toc501741316)

[*1.4 Шаблони функцій* 12](#_Toc501741317)

[2.Обгрунтування ієрархії класів 13](#_Toc501741318)

[*2.1.Успадкування* 14](#_Toc501741319)

[*2.2.Агрегація* 15](#_Toc501741320)

[*2.3. Структура програми* 15](#_Toc501741321)

[3.ОБҐРУНТУВАННЯ ВИБОРУ 17](#_Toc501741322)

[*3.1. Вибір типу властивостей* 17](#_Toc501741323)

[*3.2. Вибір перевантажених операцій* 17](#_Toc501741324)

[*3.3.Вибір виключних місць* 17](#_Toc501741325)

[ВИСНОВКИ 18](#_Toc501741326)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc501741327)

[ДОДАТОК 1. ТЕКСТ ПРОГРАМИ 20](#_Toc501741328)

[ДОДАТОК 2. ОПИС ПРОГРАМИ 54](#_Toc501741329)

# АНОТАЦІЯ

Програма моделює роботу каталогу архівів. Вона дозволяю користувачу заповнювати дані про архіви, переглядати їх у табличному вигляді, зберігати у файлі. У програмі присутній пошук за текстовим та числовим полями.

У записці розповідається про основні поняття ООП, можливості мови програмування C++, обґрунтовується вибір студента стосовно типів зв’язків між класами, типів властивостей, виключних місць.

The program simulates the work of the archive directory. It allows the user to fill out data about archives, view them in tabular form, save in a file. The program has a search for text and numeric fields.

The note describes the basic notions of the OOP, the ability of the C ++ programming language, and justifies the choice of the student in relation to the types of relationships between classes, types of properties, and exclusive places.

# ВСТУП

Об’єктно-орієнтоване програмування на сьогодні є найбільш поширеною парадигмою програмування. Згідно ООП, кожна програма представляється у вигляді сукупності об’єктів, що мають певні властивості (стан певного об’єкта, що виокремлює його серед сукупності подібних об’єктів) та певний набір дій, що може виконувати цей об’єкт. Об’єкти взаємодіють між собою, посилаючи повідомлення та змінюючи свій внутрішній стан.

Популярність ООП пояснюється її основними принципами: поліморфізм, успадкування, інкапсуляція та абстракція. Дані принципи дозволяють розбити велику програму на набір окремих об’єктів, створивши певну ієрархію між ними, тим самим спростивши етап проектування програмного продукту, розподілення праці та написання програмного коду. Завдяки ООП, програмістам стала доступна можливість повторно використовувати вже написаний код, не вдаючись у подробиці від його внутрішньої будови і користуючись лише його зовнішніми членами – методами. Розмірковуючи над цим, можна також зробити висновок, що модифікування вже написаного коду на об’єктно-орієнтованій мові може протікати менш болісно, ніж модифікування аналогічного коду на процедурній мові. Більш детальне пояснення принципів ООП наводиться в теоретичних відомостях.

ООП з’явилося ще в 60-х роках минулого століття, але почала широко використовуватися тільки у 90-х, коли розміри коду програмних продуктів стали надто складними для конструювання та, що більш важливо, підтримки.

Першою об’єктно-орієнтованою мовою була Симула. В ній з’явилися такі нові поняття як об’єкти, класи, підкласи, віртуальні методи та автоматичне прибирання пам’яті. Після з’явився Smalltalk, що увібрав у себе ці нові концепції. Саме Smalltalk став першою широко розповсюдженою об’єктно-орієнтованою мовою.

На сьогоднішній популярними ООП-мовами є C++, що є удосконаленням популярною процедурної мови C, мова Java, що з’явилася в 90-х роках та зробила величезний крок у полегшенні життя програміста.

Повертаючись до теми курсової роботи, слід сказати, що конструювання ієрархії об’єктів та виокремлення сутностей загалом є найважливішим етапом конструювання програмного забезпечення на об’єктно-орієнтованій мові. Сама можливість закарбувати об’єкт реального світу в програмному коді полегшує її конструювання та розбиття на окремі модулі. При чому не важливо, чи ця сутність є конкретним предметом чи абстрактним поняттям, бо для таких випадків об’єктно-орієнтовані мови оснащенні спеціальними поняттями та відповідними синтаксичними конструкціями, як, наприклад, модифікатор abstract, що забороняє створювати об’єкти класу, помічені цим словом. На даному етапі відбувається вибір типу відношень між об’єктами різних класів та область видимості певних їх частин. З’являються поняття успадкування та агрегація, що у свою чергу ділиться на чисту агрегацію та композицію.

Після побудови ієрархії написаний код ще не є готовим продуктом – він позбавлений інструментів, завдяки яким відбувається маніпулювання об’єктами цієї ієрархії. При цьому важливо закінчити модифікування ієрархії до початку створення інструментарію. Далі програміст зіштовхується із проблемою взаємодії програми із користувачем – зрозумілий інтерфейс, захист від помилкових даних. Тут на допомогу приходить обробка виключних ситуацій, що є особливістю об’єктно-орієнтованих мов.

Метою курсової роботи є навчитися реалізувати ієрархію класів, використовуючи основі методи та парадигми об’єктно-орієнтованого програмування.

# 

# 1.ТЕОРЕТИЧНІ ВІДОМОСТІ

## 

## *1.1.Принципи Об’єктно-орієнтованого програмування*

ООП будується на багатьох парадигмах, але виділяють чотири основних:

* Абстракція;
* Інкапсуляція;
* Успадкування;
* Поліморфізм.

Абстракція – надання об’єкту характеристик, що відрізняють його від усіх подібних об’єктів, чітко визначаючи його концептуальні границі. Основна ідея полягає в тому, щоб відокремити спосіб використання складових об'єктів даних від деталей їх реалізації у вигляді більш простих об'єктів, подібно до того, як функціональна абстракція розділяє спосіб використання функції і деталей її реалізації в термінах більш примітивних функцій, таким чином, дані обробляються функцією високого рівня за допомогою виклику функцій низького рівня. Такий підхід є основою об'єктно-орієнтованого програмування. Це дозволяє працювати з об'єктами, не вдаючись у особливості їх реалізації. У кожному конкретному випадку застосовується той чи інший підхід: інкапсуляція, поліморфізм або успадкування.

Інкапсуляція - властивість мови програмування , що дозволяє користувачеві не замислюватися про складність реалізації використовуваного програмного компонента ( що у нього всередині? ), а взаємодіяти з ним за допомогою наданого інтерфейсу (публічних методів і членів) , а також об'єднати і захистити життєво важливі для компонента дані. При цьому користувачеві надається тільки специфікація (інтерфейс ) об'єкта . Користувач може взаємодіяти з об'єктом тільки через цей інтерфейс . Реалізується за допомогою ключового слова: public. Користувач не може використовувати закриті дані і методи. Реалізується за допомогою ключових слів: private, protected. Приховування реалізації доцільно застосовувати в наступних випадках: гранична локалізація змін при необхідності таких змін, прогнозованість змін (які зміни в коді треба зробити для заданої зміни функціональності) і прогнозованість наслідків змін.

Успадкування - один з найважливіших механізмів об'єктно-орієнтованого програмування, що дозволяє описати новий клас на основі вже існуючого ( батьківського), при цьому властивості і функціональність батьківського класу запозичуються новим класом . Іншими словами , клас - спадкоємець реалізує специфікацію вже існуючого класу. Це дозволяє звертатися до об'єктів класу - спадкоємця так само, як до об'єктів базового класу. Просте успадкування : клас – спадкоємець має лише один батьківський клас. Класи, які походять від базового, називаються нащадками, спадкоємцями або похідними класами. Абстрактний клас - це клас, що містить хоча б один абстрактний метод, він описаний в програмі, має поля, методи і не може використовуватися для безпосереднього створення об'єкта. Тобто від абстрактного класу можна тільки наслідувати. Об'єкти створюються тільки на основі похідних класів, успадкованих від абстрактного. Наприклад, абстрактним класом може бути базовий клас «співробітник вузу», від якого успадковуються класи «аспірант», «професор» і т.д. Так як похідні класи мають спільні поля і функції (наприклад, поле «рік народження»), то ці члени класу можуть бути описані в базовому класі . У програмі створюються об'єкти на основі класів «аспірант», «професор» , але немає сенсу створювати об'єкт на основі класу «співробітник вузу».

Множинне спадкування

При множинному успадкуванні у класу може бути більше одного предка. У цьому випадку клас успадковує методи всіх предків. Переваги такого підходу в більшій гнучкості. Множинне успадкування реалізовано в C ++. Але множинне успадкування - потенційне джерело помилок, які можуть виникнути через наявність однакових імен методів у предках. У мовах, які позиціонуються як спадкоємці C ++ (Java, C # та ін.), від множинного успадкування було вирішено відмовитися на користь інтерфейсів. Практично завжди можна обійтися без використання даного механізму.

Поліморфізмом називають явище, при якому функції (методу) з одним і тим же ім'ям відповідає різний програмний код (поліморфний код) в залежності від того, об'єкт якого класу використовується при виклику даного методу. Поліморфізм забезпечується тим, що в класі-нащадку змінюють реалізацію методу класу-предка з обов'язковим збереженням сигнатури методу. Це забезпечує збереження незмінним інтерфейсу класу-предка і дозволяє здійснити зв'язування імені методу в коді з різними класами - з об'єкта якого класу здійснюється виклик, з того класу і береться метод з такою назвою. Такий механізм називається динамічним (або пізнім) зв'язуванням - на відміну від статичного (раннього) зв'язування, здійснюваного на етапі компіляції.

## *1.4.Перевантаження операцій*

Іноді для полегшення коду та його читання хотілося би, наприклад, щоб операцією + можна було додавати не тільки цілі або дробові числа, а і поєднувати рядки:

char str1[15] = "Hello ";

char str2[] = "world!";

str1 + str2;

Для цього маємо сказати компілятору, що операцією + хочемо поєднати рядки, а не додати два числа. Така операція і називається перегрузкою оператора +. Окрім базових типів даних, у С++операції можна перевантажити і для користувацьких типів даних – структур та класів. Тоді кожен з них унікально реагуватиме на одну і ту ж операцію.

Для перевантаження оператора служить ключове слово operator, що визначає операторний метод, який у свою чергу визначає дію оператора щодо свого класу.

Існують дві форми операторних методів: одна – для унарних операторів, інша – для бінарних.

Роздивимося загальну форму перевантаження унарного оператора

public:

…

Тип\_повернення operator op (тип\_параметра1 операнд1, тип\_параметра2 операнд2)

{

//Операції

}

***1.3. Робота з файлами***

Робота з файлами є досить вагомою у програмуванні. Їх можна використовувати як для заповнення даних у програмі, так і для демонстрації результатів роботи програми разом із збереженням даних. Для роботи з файлами необхідно підключити заголовний файл <fstream>. У <fstream> визначені кілька класів і підключені заголовки <ifstream> - файловий ввід і <ofstream> - файловий вивід.

Файловий ввід / вивід аналогічний стандартному вводу / виводу, єдина відмінність - це те, що введення / виведення виконується на екран, а в файл. Якщо введення / виведення на стандартні пристрої виконується за допомогою об'єктів cin і cout, то для організації файлового введення / виводу досить створити власні об'єкти, які можна використовувати аналогічно операторам cin і cout.

Наприклад, необхідно створити текстовий файл і записати(зчитати) в нього рядок. Для цього необхідно виконати наступні кроки:

* створити об'єкт класу ofstream (ifstream);
* зв'язати об'єкт класу з файлом, в який буде проводитися запис;
* записати (зчитати) рядок в файл;
* закрити файл.

## *1.4 Шаблони функцій*

Шаблони класів визначають сімейство пов'язаних класів, заснованих на типі аргументів, переданих класу при створенні його примірника. Шаблони функцій схожі на шаблони класів, але визначають сімейство функцій. За допомогою шаблонів функцій можна задавати набори функцій, заснованих на одному коді, але діють в різних типах або класах. Наступний шаблон функції міняє місцями два елементи.

template< class T > void MySwap( T& a, T& b ) {

T c(a);

a = b;

b = c;

}

Цей код визначає сімейство функцій, які міняють місцями значення аргументів. З цього шаблону можна створювати функції, що змінюють місцями типи int і long, а також призначені для користувацькиї типів. Функція MySwap навіть міняє місцями класи, якщо правильно визначено конструктор копії і оператор присвоювання.

Крім того, шаблон функції не дозволяє розробнику міняти місцями об'єкти різних типів, оскільки компілятор під час компіляції знає типи параметрів a і b.

# 2.Обгрунтування ієрархії класів

Агрегація

Композиція

**Katalog**

**Archive\_item**

**Text\_doc**

**Date**

**File**

Для встановлення правильних зв’язків під час формування ієрархії класів важливо розуміти суть успадкування та агрегації. Під час успадковування клас – потомок по своїй логічній сутності є також об’єктом батьківського класу. Більше того він розширюється, уточнюється, набуває конкретних властивостей. Під агрегацією мають на увазі те, що клас, який агрегує інший клас не є по суті класом, який агрегує у собі, а містить його як частину. У якості прикладу роздивимося класи Person і Author. Безумовно, що автор є особою, має прізвище, ім’я, дату народження. Проте він може доповнюватися, наприклад, списком надрукованих творів. Легко бачити, що між цими двома класами встановлюється зв'язок успадкування. З іншої сторони і особа і автор мають дату народження, але вони не є датами, але мають її як частину себе. Встановлюється зв'язок агрегації.

У таблиці нижче описано завдання курсової роботи 18 варіанта.

Таблиця 2.2. Завдання курсової роботи 18 варіанта

|  |  |
| --- | --- |
| 18 | Дата {Рік, Місяць, День};  Особа {Прізвище, Ім’я, Дата (народження)};  Файл {Назва, Розширення, Дата (створення), Розмір};  Текстовий Документ {Файл, Розміщення (диск, шлях), Особа (автор), Ключові слова};  Елемент архіву {Список Текстових документ архіву, Дата (створення)};  Склад каталогу {Масив елементів архіву}.  Середнє значення кількості Текстових документів в елементах архіву та середнє значення ступеня архівації в елементах архіву. |

## *2.1.Успадкування*

Проаналізувавши дану ієрархію, видно, що класи Текстовий документ та

Елемент архіву можуть наслідуватися від класу Файл, отримавши від нього поля Назва, Розширення, Дата (створення) та розмір. Із завдання наслідування Елемента архіву від Файлу не очевидне, проте для виконання другої частини додаткового завдання з розрахунком середнього значення ступеня архівації необхідно, щоб об’єкт класу Елемент архіву мав свій розмір. Одне із запропонований полів Елементу архіва Дата (створення) включиться при наслідуванні, адже клас Файл уже має Дату (створення). Також архів має власне ім’я та розширення.

## *2.2.Агрегація*

Одразу важливо уточнити відмінності між чистою агрегацією та композицією. Композиція – це вид агрегації, що говорить нам про те, що об’єкт агрегованого класу існує тільки всередині об’єкту класу, що агрегує. Тому між Персоною та Текстовим документом обрана саме композиція. А також між датою та архівом елементів.

## *2.3. Структура програми*

Дата представляє собою клас Date з полями день, місяць рік. Також він містить методи для встановлення або зміни полів з урахуванням на обмеженість числа допустимих значень. Даний клас перебуває у відношенні композиція з класами Person, File, Archive\_item.

Персона представляє собою клас Person з полями ім’я, прізвище та дата народження. Поля ім’я та прізвище обмені кількістю символів. Даний клас перебуває у відношенні композиція з класом Text\_doc.

Клас File має поля назва, розширення, розмір та дата створення. Назва та розширення обмежені кількістю символів. Другий також обженени форматом («.\*»). Розмір файлу також є обмеженим зверху. Даний клас є батьківським для класів Text\_doc i Archive\_item.

Текстовий документ представлений класом Text\_doc. Має власні поля розміщення, автор, список ключових слів та їх кількість, але так як він є наслідником класу File також присутні поля назва, розширення, розмір та дата створення.

Архів документів реалізований як Archive\_item. Він також є наслідником класу File. Містить поля Список текстових документів та їх кількість.

Останнім у ієрархії є клас Katalog, який являється списком елементів архіву. Також містить поле кількість архівів.

# 3.ОБҐРУНТУВАННЯ ВИБОРУ

## *3.1. Вибір типу властивостей*

Для більшості класів, якщо розробка програми того вимагала були реалізовані методи для заповнення з обмеженняит Set та отримання значень Get певних закритих полів. В результаті досягнуто заповнення об’єктів коректними даними.

## *3.2. Вибір перевантажених операцій*

Була перевантажена операція індексування для класу Katalog, що дозволило більш зручно та швидко звертатися до певного архіву у списку. Також для класів, поля яких містять рядок символів була перевантажена операція = для правильного копіювання.Для кожного класа перевантажений оператор << для легшого подальшого виведення у табличному режимі.

## *3.3.Вибір виключних місць*

Обробка виключних ситуацій застосовується при відкритті файлу, на випадок, якщо файл с таким ім’ям створити неможливо, відсутній доступ до відповідного диску чи каталогу, або відсутній файл для зчитування даних. Будь яке місце, де користувач вводить дані має додатково аналізуватися та оброблятися. В даній роботі встановленні обмеження на ввід цілих або дробович чисел (щоб виключити ввід букв). Поля, що є рядками символів обмежені кількістю символів. Унікальне поле розширення файлу вимагатиме від користувача строгої форми запису («.\*»). Також обробляються виключні ситуації при перевантаженні оператора індексування, щоб не вийти за межі масиву.

# ВИСНОВКИ

Під час написання курсової роботи були використані можливості мови С++ для створення ієрархії класів. Проаналізувавши задану ієрархію, побудувавши її графічну модель та опираючись на основні парадигми об’єктно-орієнтованого програмування: абстракцію, інкапсуляцію, успадкування та поліморфізм, - ієрархія була успішно реалізована у MVS 2012. Також була задіяна робота з файлами, виконане перевантаження операторів, оброблені виключні ситуації.

У результаті програмний продукт функціонально відповідає поставленому завданню: в програмі присутня можливість вводу даних з клавіатури, вивід у вигляді таблиці, запис у файл у текстовому та бінарному режимах, зчитування з текстових та бінарних файлів, пошут даних по текстових або числових полях, оброблені виключні ситуації, перевантажені оператори та зроблене просте меню для легкого орієнтування користувача у програмі.

# 

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Роберт Лафоре — Объектно-ориентированное программирование в С++
2. Бьерн Страуструп (Bjarne Stroustrup) — Язык программирования С++ (The C++ Programming Language)
3. http://msdn.microsoft.com
4. http://en.wikipedia.org
5. http://cyberforum.ru

# ДОДАТОК 1. ТЕКСТ ПРОГРАМИ

**#pragma once**

**using namespace std;**

**#define OUT 50**

**class MyError**

**{ //обробка виключних ситуацій, визначених як ієрархії класів**

**protected:**

**char\* message;**

**public:**

**MyError()**

**{**

**message = new char[strlen("NoErr") + 1];**

**strcpy(message, "NoErr");**

**}**

**MyError(const char\* \_message)**

**{**

**message = new char[strlen(\_message) + 1];**

**strcpy(message, \_message);**

**}**

**MyError(const MyError& \_message)**

**{**

**message = new char[strlen(\_message.message) + 1];**

**strcpy(message, \_message.message);**

**}**

**virtual ~MyError()**

**{**

**if(!message) delete[] message;**

**}**

**MyError& operator=(const MyError& p)**

**{**

**message = new char[strlen(p.message) + 1];**

**strcpy(message, p.message);**

**}**

**virtual const char\* what() const**

**{**

**return message;**

**}**

**};**

**class FileError : public MyError**

**{**

**private:**

**char\* filename;**

**public:**

**FileError(const char\* \_message,const char\* \_filename)**

**{**

**message = new char[strlen(\_message) + 1];**

**strcpy(message, \_message);**

**filename = new char[strlen(\_filename) + 1];**

**strcpy(filename, \_filename);**

**}**

**FileError(const FileError& p)**

**{**

**message = new char[strlen(p.message) + 1];**

**strcpy(message, p.message);**

**filename = new char[strlen(p.filename) + 1];**

**strcpy(filename, p.filename);**

**}**

**~FileError()**

**{**

**if(!filename) delete[] filename;**

**}**

**FileError& operator=(const FileError& p)**

**{**

**if(message) delete[] message;**

**if(filename) delete[] filename;**

**message = new char[strlen(p.message) + 1];**

**strcpy(message, p.message);**

**filename = new char[strlen(p.filename) + 1];**

**strcpy(filename, p.filename);**

**}**

**const char\* what() const**

**{**

**return message;**

**}**

**};**

**class IndexError : public MyError {**

**public:**

**IndexError(char\* message) : MyError(message){}**

**IndexError(const IndexError& p) : MyError(p) {}**

**~IndexError() {}**

**IndexError& operator=(const IndexError& p) {**

**if (this == &p) return \*this;**

**MyError::operator=(p);**

**return \*this;**

**}**

**const char\* what() const {**

**return MyError::what();**

**}**

**};**

**class Date**

**{**

**int day,**

**month,**

**year;**

**public:**

**//конструктори**

**Date()**

**{**

**day = 1;**

**month = 1;**

**year = 1960;**

**}**

**Date(const int d, const int m, const int y) :**

**day((d > 0 && d < 31) ? d : 1),**

**month((m > 0 && m < 13) ? m : 1),**

**year((y > 1969 && y < 2018) ? y : 1980) {};**

**//методи заповнення полів с обмеженнями**

**void SetDay()**

**{**

**int d;**

**cout << "Day : ";**

**while(scanf("%d", &d)!=1 || d <= 0 || d > 30)**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**day = d;**

**}**

**void SetMonth()**

**{**

**int m;**

**cout << "Month: ";**

**while(scanf("%d", &m)!=1 || m <= 0 || m > 12)**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**month = m;**

**}**

**void SetYear()**

**{**

**int y;**

**cout << "Year(1960 - 2017): ";**

**while(scanf("%d", &y)!=1 || y < 1960 || y > 2017)**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**year = y;**

**}**

**//методи отримання значень закритих полів**

**int GetDay()**

**{**

**return day;**

**}**

**int GetMonth()**

**{**

**return month;**

**}**

**int GetYear()**

**{**

**return year;**

**}**

**//перевантаження операції <<**

**friend ostream& operator << (ostream& stream, const Date & date)**

**{**

**stream << " " << setw(2) << setfill('0') << right << date.day << "." << setw(2) << setfill('0') << date.month << "." << date.year << setfill(' ') << " ";**

**return stream;**

**}**

**};**

**class Person**

**{**

**char**

**\*name,**

**\*surname;**

**Date birthday;**

**public:**

**//конструктори**

**Person()**

**{**

**name = new char[8];**

**strcpy(name, "DefName");**

**surname = new char[10];**

**strcpy(surname, "DefSurame");**

**birthday = Date();**

**}**

**Person(const char\* \_name, const char\* \_surname, const Date & \_birthday) : birthday(\_birthday)**

**{**

**name = new char[strlen(\_name) + 1];**

**surname = new char[strlen(\_surname) + 1];**

**strcpy(name, \_name);**

**strcpy(surname, \_surname);**

**}**

**Person(const Person & copy) : birthday(copy.birthday)**

**{**

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

**surname = new char[strlen(copy.surname) + 1];**

**strcpy(name, copy.name);**

**strcpy(surname, copy.surname);**

**}**

**Person& operator = (const Person & copy)**

**{**

**if (this == &copy)**

**return \*this;**

**birthday = copy.birthday;**

**if (name)**

**delete[] name;**

**if (surname)**

**delete[] surname;**

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

**surname = new char[strlen(copy.surname) + 1];**

**strcpy(name, copy.name);**

**strcpy(surname, copy.surname);**

**return \*this;**

**}**

**//методи заповнення полів с обмеженнями**

**void SetName()**

**{**

**char \*\_name = new char[256];**

**do**

**{**

**cout << "Enter Author's name (up to 10 symbols): ";**

**cin >> \_name;**

**}**

**while(strlen(\_name) > 10);**

**if(name) delete[] name;**

**name = new char[strlen(\_name) + 1];**

**strcpy(name, \_name);**

**delete[] \_name;**

**}**

**void SetSurame()**

**{**

**char \*\_surname = new char[256];**

**do**

**{**

**cout << "Enter Author's surname (up to 10 symbols): ";**

**cin >> \_surname;**

**}**

**while(strlen(\_surname) > 10);**

**if(surname) delete[] surname;**

**surname = new char[strlen(\_surname) + 1];**

**strcpy(surname, \_surname);**

**delete[] \_surname;**

**}**

**void SetBirthday()**

**{**

**cout << "Enter Author's birthday : " << endl;**

**birthday.SetDay();**

**birthday.SetMonth();**

**birthday.SetYear();**

**}**

**//методи отримання значень із закритих полів**

**char\* GetName()**

**{**

**return name;**

**}**

**char\* GetSurname()**

**{**

**return surname;**

**}**

**Date GetBirthday()**

**{**

**return birthday;**

**}**

**friend ostream& operator << (ostream& stream, const Person & arg)**

**{**

**stream << setw(10) << left << arg.name << " " << setw(10) << left << arg.surname << (char)0xB3 << arg.birthday;**

**return stream;**

**}**

**~Person()**

**{**

**if(!name) delete[] name;**

**if(!surname) delete[] surname;**

**}**

**};**

**class File**

**{**

**protected:**

**char \*file\_name,**

**\*file\_extention;**

**Date file\_create\_date;**

**unsigned size;**

**public:**

**//конструктори**

**File()**

**{**

**file\_name = new char[12];**

**strcpy(file\_name, "DefFileName");**

**file\_extention = new char[5];**

**strcpy(file\_extention, ".def");**

**file\_create\_date = Date();**

**size = 0;**

**}**

**File(char\* \_file\_name, char\* \_file\_extention, Date \_create\_date, unsigned \_size) : file\_create\_date(\_create\_date), size(\_size >= 0 && \_size < 100000000 ? \_size : 0)**

**{**

**if(strlen(\_file\_name) < 20)**

**{**

**file\_name = new char [strlen(\_file\_name) + 1];**

**strcpy(file\_name, \_file\_name);**

**}**

**else**

**{**

**file\_name = new char[5];**

**strcpy(file\_name, "None");**

**}**

**if(strlen(\_file\_extention) < 5 && \_file\_extention[0] == '.')**

**{**

**file\_extention = new char [strlen(\_file\_extention) + 1];**

**strcpy(file\_extention, \_file\_extention);**

**}**

**else**

**{**

**file\_extention = new char[5];**

**strcpy(file\_extention, ".none");**

**}**

**}**

**File& operator = (const File& copy)**

**{**

**if (this == &copy)**

**return \*this;**

**size = copy.size;**

**if (file\_name)**

**delete[] file\_name;**

**if (file\_extention)**

**delete[] file\_extention;**

**file\_name = new char[strlen(copy.file\_name) + 1];**

**file\_extention = new char[strlen(copy.file\_extention) + 1];**

**strcpy(file\_name, copy.file\_name);**

**strcpy(file\_extention, copy.file\_extention);**

**file\_create\_date = copy.file\_create\_date;**

**return \*this;**

**}**

**//методи заповнення полів с обмеженнями**

**void SetFileName()**

**{**

**char \*\_file\_name = new char[256];**

**do**

**{**

**cout << "Enter file name (up to 20 symbols): ";**

**cin >> \_file\_name;**

**}**

**while(strlen(\_file\_name) > 20);**

**if(file\_name) delete[] file\_name;**

**file\_name = new char[strlen(\_file\_name) + 1];**

**strcpy(file\_name, \_file\_name);**

**delete[] \_file\_name;**

**}**

**void SetFileExtention()**

**{**

**char \*\_file\_extention = new char[256];**

**do**

**{**

**cout << "Enter file extention (example: .txt) : ";**

**cin >> \_file\_extention;**

**}**

**while(strlen(\_file\_extention) > 5 || \_file\_extention[0]!='.');**

**if(file\_extention) delete[] file\_extention;**

**file\_extention = new char[strlen(\_file\_extention) + 1];**

**strcpy(file\_extention, \_file\_extention);**

**delete[] \_file\_extention;**

**}**

**void SetFileCreateDate()**

**{**

**cout << "Enter date of creating :\n";**

**file\_create\_date.SetDay();**

**file\_create\_date.SetMonth();**

**file\_create\_date.SetYear();**

**}**

**void SetSize()**

**{**

**cout << "Enter size (byte): ";**

**while(scanf("%d", &size)!=1 || size < 0 || size > 1000000000)**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**}**

**//методи отримання значень із закритих полів**

**char\* GetFileName() const**

**{**

**return file\_name;**

**}**

**char\* GetFileExtention() const**

**{**

**return file\_extention;**

**}**

**int GetSize() const**

**{**

**return size;**

**}**

**Date GetCreatingDate() const**

**{**

**return file\_create\_date;**

**}**

**//деструктор**

**~File()**

**{**

**if (!file\_name) delete[] file\_name;**

**if (!file\_extention) delete[] file\_extention;**

**}**

**//перевантаження операції <<**

**friend ostream& operator << (ostream& stream, const File & f)**

**{**

**char\* full\_name = new char[strlen(f.file\_name) + strlen(f.file\_extention) + 1];**

**int i, k = strlen(f.file\_name);**

**for(i = 0; i < k; i++) full\_name[i] = f.file\_name[i];**

**for(i; i < k + strlen(f.file\_extention); i++) full\_name[i] = f.file\_extention[i-k]; full\_name[i] = '\0';**

**stream << setw(25) << left << full\_name << (char)0xB3 << f.file\_create\_date << (char)0xB3 << setw(10) << left << f.size;**

**return stream;**

**}**

**};**

**class Text\_doc : public File**

**{**

**char\* path;**

**Person author;**

**char\*\* key\_words;**

**int word\_count;**

**public:**

**//конструктори**

**Text\_doc()**

**{**

**path = new char[strlen("C:/default") + 1];**

**strcpy(path, "C:/default");**

**key\_words = new char\*[1];**

**key\_words[0] = new char[strlen("None") + 1];**

**strcpy(key\_words[0], "None");**

**author = Person();**

**word\_count = 1;**

**}**

**Text\_doc(File f, char\* \_path, Person \_author, char\*\* \_key\_words, int n) : author(\_author), word\_count(n >= 0 && n < 5 ? n : 0), File(f)**

**{**

**if(strlen(\_path) < 30)**

**{**

**path = new char [strlen(\_path) + 1];**

**strcpy(path, \_path);**

**}**

**else**

**{**

**path = new char[5];**

**strcpy(path, "None");**

**}**

**key\_words = new char\*[word\_count];**

**for(int i = 0; i < word\_count; i++)**

**{**

**if(strlen(\_key\_words[i]) <= 10)**

**{**

**key\_words[i] = new char[strlen(\_key\_words[i]) + 1];**

**strcpy(key\_words[i], \_key\_words[i]);**

**}**

**else**

**{**

**key\_words[i] = new char[5];**

**strcpy(key\_words[i], "None");**

**}**

**}**

**}**

**Text\_doc(const Text\_doc& copy)**

**{**

**path = new char[strlen(copy.path) + 1];**

**strcpy(path, copy.path);**

**file\_name = new char[strlen(copy.file\_name) + 1];**

**strcpy(file\_name, copy.file\_name);**

**file\_extention = new char[strlen(copy.file\_extention) + 1];**

**strcpy(file\_extention, copy.file\_extention);**

**key\_words = new char\*[copy.word\_count];**

**for(int i = 0; i < copy.word\_count; i++)**

**{**

**key\_words[i] = new char[strlen(copy.key\_words[i]) + 1];**

**strcpy(key\_words[i], copy.key\_words[i]);**

**}**

**word\_count = copy.word\_count;**

**author = copy.author;**

**file\_create\_date = copy.file\_create\_date;**

**size = copy.size;**

**}**

**//методи заповнення полів с обмеженнями**

**void SetPath()**

**{**

**char\* \_path = new char[256];**

**do**

**{**

**cout << "Enter path (up to 30 symbols): ";**

**cin >> \_path;**

**}**

**while(strlen(\_path) > 30);**

**if(path) delete[] path;**

**path = new char[strlen(\_path) + 1];**

**strcpy(path, \_path);**

**delete[] \_path;**

**}**

**void SetAuthor()**

**{**

**author.SetName();**

**author.SetSurame();**

**author.SetBirthday();**

**}**

**void SetWordCount()**

**{**

**cout << "Enter words count (0 - 4) : ";**

**while(scanf("%d", &word\_count)!=1 || word\_count < 0 || word\_count > 4);**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**}**

**void SetKeyWords()**

**{**

**if(key\_words)**

**{**

**for(int i = 0; i < word\_count; i++) delete key\_words[i];**

**delete[] key\_words;**

**}**

**SetWordCount();**

**key\_words = new char\*[word\_count];**

**cout << "Enter new key words (up to 10 symbols) :\n";**

**for(int i = 0; i < word\_count; i++)**

**{**

**char\* temp = new char[256];**

**do**

**{**

**cout << i + 1 << ". ";**

**cin >> temp;**

**}**

**while(strlen(temp) > 10);**

**key\_words[i] = new char[strlen(temp) + 1];**

**strcpy(key\_words[i], temp);**

**delete[] temp;**

**}**

**}**

**//функції пошуку за текстовими та числовими полями**

**bool SearchText(char\* search)**

**{**

**if(strcmp(search, file\_name) == 0 || strcmp(search, file\_extention) == 0 || strcmp(search, path) == 0 || strcmp(search, author.GetName()) == 0 || strcmp(search, author.GetSurname()) == 0) return true;;**

**for(int i = 0; i < word\_count; i++)**

**if(strcmp(search, key\_words[i])== 0) return true;;**

**}**

**bool SearchNumber(double n)**

**{**

**bool cmp = false;**

**if(n == file\_create\_date.GetDay() || n == file\_create\_date.GetMonth() || n == file\_create\_date.GetYear() || n == size) cmp = true;**

**if(n == author.GetBirthday().GetDay() || n == author.GetBirthday().GetMonth() || n == author.GetBirthday().GetYear()) cmp = true;**

**return cmp;**

**}**

**//перевантаження операції =**

**Text\_doc& operator = (const Text\_doc& copy)**

**{**

**if(path) delete[] path;**

**if(file\_name) delete[] file\_name;**

**if(file\_extention) delete[] file\_extention;**

**if(key\_words)**

**{**

**for(int i = 0; i < word\_count; i++)**

**delete[] key\_words[i];**

**delete[] key\_words;**

**}**

**path = new char[strlen(copy.path) + 1];**

**strcpy(path, copy.path);**

**file\_name = new char[strlen(copy.file\_name) + 1];**

**strcpy(file\_name, copy.file\_name);**

**file\_extention = new char[strlen(copy.file\_extention) + 1];**

**strcpy(file\_extention, copy.file\_extention);**

**key\_words = new char\*[copy.word\_count];**

**for(int i = 0; i < copy.word\_count; i++)**

**{**

**key\_words[i] = new char[strlen(copy.key\_words[i]) + 1];**

**strcpy(key\_words[i], copy.key\_words[i]);**

**}**

**word\_count = copy.word\_count;**

**author = copy.author;**

**file\_create\_date = copy.file\_create\_date;**

**size = copy.size;**

**return \*this;**

**}**

**//деструктор**

**~Text\_doc()**

**{**

**if(!path) delete[] path;**

**for(int i = 0; i < word\_count; i++) delete[] key\_words[i];**

**delete[] key\_words;**

**}**

**//перевантаження операції <<**

**friend ostream& operator << (ostream& stream, const Text\_doc & td)**

**{**

**char\* full\_name = new char[strlen(td.file\_name) + strlen(td.file\_extention) + 1];**

**int i, k = strlen(td.file\_name), cur = 1;**

**for(i = 0; i < k; i++) full\_name[i] = td.file\_name[i];**

**for(i; i < k + strlen(td.file\_extention); i++) full\_name[i] = td.file\_extention[i-k]; full\_name[i] = '\0';**

**stream << setw(25) << left << full\_name << (char)0xB3 << setw(10) << left << td.size << (char)0xB3 << setw(35) << left << td.path << (char)0xB3 << td.file\_create\_date << (char)0xB3 << td.author << (char)0xB3;**

**for(int i = 0; i < td.word\_count; i++)**

**{**

**if(cur + strlen(td.key\_words[i]) + 2 < OUT)**

**{**

**stream << td.key\_words[i];**

**if(i+1 < td.word\_count) stream << ", ";**

**cur+=strlen(td.key\_words[i]) + 2;**

**}**

**}**

**stream << setw(OUT-cur);**

**return stream;**

**}**

**};**

**class Archive\_item : public File**

**{**

**Text\_doc\* doc\_list;**

**int count;**

**public:**

**//конструктори**

**Archive\_item()**

**{**

**count = 3;**

**doc\_list = new Text\_doc[count];**

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

**{**

**doc\_list[i] = Text\_doc();**

**}**

**}**

**Archive\_item(File \_file, Text\_doc\* fl, int \_count) : count(\_count > 0 && \_count < 50 ? \_count : 0)**

**{**

**file\_name = new char[strlen(\_file.GetFileName()) + 1];**

**strcpy(file\_name, \_file.GetFileName());**

**file\_extention = new char[strlen(\_file.GetFileExtention()) + 1];**

**strcpy(file\_extention, \_file.GetFileExtention());**

**size = \_file.GetSize();**

**file\_create\_date = \_file.GetCreatingDate();**

**doc\_list = new Text\_doc[count];**

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

**{**

**doc\_list[i] = fl[i];**

**}**

**}**

**Archive\_item(const Archive\_item& copy) : File(copy)**

**{**

**count = copy.count;**

**doc\_list = new Text\_doc[count];**

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

**{**

**doc\_list[i] = copy.doc\_list[i];**

**}**

**}**

**//перевантаження операції індексування**

**Text\_doc& operator [] (int index) const**

**{**

**return doc\_list[index];**

**}**

**//деструктор**

**~Archive\_item()**

**{**

**if(!doc\_list) delete[] doc\_list;**

**}**

**//перевантаження операції =**

**Archive\_item& operator = (const Archive\_item& copy)**

**{**

**if(this==&copy) return \*this;**

**if(file\_name) delete[] file\_name;**

**if(file\_extention) delete[] file\_extention;**

**file\_name = new char[strlen(copy.GetFileName()) + 1];**

**strcpy(file\_name, copy.GetFileName());**

**file\_extention = new char[strlen(copy.GetFileExtention()) + 1];**

**strcpy(file\_extention, copy.GetFileExtention());**

**size = copy.GetSize();**

**file\_create\_date = copy.GetCreatingDate();**

**count = copy.count;**

**if(doc\_list) delete[] doc\_list;**

**doc\_list = new Text\_doc[count];**

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

**{**

**doc\_list[i] = copy.doc\_list[i];**

**}**

**return \*this;**

**}**

**//функції пошуку по текстовому та числовому полях**

**void SearchInText(char\* search)**

**{**

**bool cmp = false;**

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

**{**

**cmp = doc\_list[i].SearchText(search);**

**if(cmp) break;**

**}**

**if(strcmp(search, file\_name) == 0 || strcmp(search, file\_extention) == 0) cmp = true;**

**if(cmp) cout << \*this << endl;**

**}**

**void SearchInNumbers(double n)**

**{**

**bool cmp = false;**

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

**{**

**cmp = doc\_list[i].SearchNumber(n);**

**if(cmp) break;**

**}**

**if(n == file\_create\_date.GetDay() || n == file\_create\_date.GetMonth() || n == file\_create\_date.GetYear() || n == size) cmp = true;**

**if(cmp) cout << \*this << endl;**

**}**

**//методи отримання значень із закритих полів**

**int GetCount()**

**{**

**return count;**

**}**

**//функція для отримання коеф. архівування**

**double GetArchCoefficient()**

**{**

**double sum = 0;**

**for(int i = 0; i < count; i++) sum+= doc\_list[i].GetSize();**

**if(sum == 0) return 0;**

**return sum/size;**

**}**

**//перевантаження операції <<**

**friend ostream& operator << (ostream& stream, const Archive\_item & ai)**

**{**

**char\* full\_name = new char[strlen(ai.file\_name) + strlen(ai.file\_extention) + 1];**

**int i, k = strlen(ai.file\_name), cur = 1;**

**for(i = 0; i < k; i++) full\_name[i] = ai.file\_name[i];**

**for(i; i < k + strlen(ai.file\_extention); i++) full\_name[i] = ai.file\_extention[i-k]; full\_name[i] = '\0';**

**for(int i = 0; i < ai.count; i++)**

**{**

**if(i==0) stream << (char)0xB3 << setw(25) << left << full\_name << (char)0xB3 << setw(10) << left << ai.size << (char)0xB3 << ai.file\_create\_date << (char)0xB3 << ai.doc\_list[i] << (char)0xB3 << endl;**

**else stream <<(char)0xB3 << setw(26) << (char)0xB3 << setw(11) << (char)0xB3 << setw(13) << (char)0xB3 << ai.doc\_list[i] << (char)0xB3 << endl;**

**}**

**return stream;**

**}**

**};**

**class Katalog**

**{**

**Archive\_item\* Catalog;**

**int items\_count;**

**public:**

**//конструктори**

**Katalog()**

**{**

**items\_count = 3;**

**Catalog = new Archive\_item[items\_count];**

**for(int i = 0; i < items\_count; i++)**

**{**

**Catalog[i] = Archive\_item();**

**}**

**}**

**Katalog(Archive\_item\* \_Catalog, int \_items\_count) : items\_count(\_items\_count < 50 && \_items\_count > 0 ? \_items\_count : 0)**

**{**

**Catalog = new Archive\_item[items\_count];**

**for(int i = 0; i < items\_count; i++) Catalog[i] = \_Catalog[i];**

**}**

**//перевантаження операції =**

**Katalog& operator = (const Katalog & copy)**

**{**

**if (this == &copy)**

**return \*this;**

**if (Catalog)**

**delete[] Catalog;**

**items\_count = copy.items\_count;**

**Catalog = new Archive\_item[items\_count];**

**for(int i = 0; i < items\_count; i++)**

**Catalog[i] = copy.Catalog[i];**

**return \*this;**

**}**

**//деструктор**

**~Katalog()**

**{**

**if(!Catalog) delete[] Catalog;**

**}**

**//функції для запису та зчитування з файлів**

**void WriteToTxt(void)**

**{**

**ofstream file("txt.txt");**

**try**

**{**

**if(file.is\_open())**

**{**

**file << \*this;**

**file.close();**

**cout << "Catalog has been written to txt.txt" << endl;**

**}**

**else**

**{**

**FileError err("File is not opened", "read.txt");**

**throw err;**

**}**

**}**

**catch (FileError err)**

**{**

**cerr << err.what() << endl;**

**return;**

**}**

**}**

**void WriteToBin(void)**

**{**

**ofstream file("bin.txt", ios::binary);**

**try**

**{**

**if(file.is\_open())**

**{**

**file.write((char\*)&items\_count, sizeof(items\_count));**

**for(int i = 0; i < items\_count; i++)**

**file.write((char\*)(&Catalog[i]), sizeof(Archive\_item));**

**file.close();**

**}**

**else**

**{**

**FileError err("File is not opened", "read.txt");**

**throw err;**

**}**

**}**

**catch (FileError err)**

**{**

**cerr << err.what() << endl;**

**return;**

**}**

**}**

**void ScanTxt()**

**{**

**cout << "Catalog will be read from read.txt. Chek it and press any key to continue" << endl;**

**system("pause");**

**ifstream file("read.txt");**

**try**

**{**

**if(file.is\_open())**

**{**

**char\* str = new char[220];**

**while(!file.eof())**

**{**

**file.getline(str,221,'\n');**

**cout << str << endl;**

**}**

**cout << endl;**

**file.close();**

**}**

**else**

**{**

**FileError err("File is not opened", "read.txt");**

**throw err;**

**}**

**}**

**catch (FileError err)**

**{**

**cerr << err.what() << endl;**

**return;**

**}**

**}**

**void ScanBin()**

**{**

**ifstream file("bin.txt", ios::binary);**

**try**

**{**

**if(file.is\_open())**

**{**

**file.read((char\*)&items\_count, sizeof(items\_count));**

**for(int i = 0; i < items\_count; i++)**

**file.read((char\*)(&Catalog[i]), sizeof(Archive\_item));**

**file.close();**

**}**

**else**

**{**

**FileError err("File is not opened", "read.txt");**

**throw err;**

**}**

**}**

**catch (FileError err)**

**{**

**cerr << err.what() << endl;**

**return;**

**}**

**}**

**//методи отримання значень закритих полів**

**int GetArchCount()**

**{**

**return items\_count;**

**}**

**//перевантаження операції індексування**

**Archive\_item& operator [] (int index) const**

**{**

**try**

**{**

**if(index >= 0 && index < items\_count)**

**return Catalog[index];**

**IndexError err("Uncorrect index value");**

**throw err;**

**}**

**catch (IndexError err)**

**{**

**cerr << err.what() << endl;**

**}**

**}**

**//перевантаження операції <<**

**friend ostream& operator << (ostream& stream, const Katalog & ktl)**

**{**

**if(ktl.items\_count == 0) return stream << "Catalog is empty\n";**

**char\* c = new char[220];**

**for(int i = 0; i < 220; i++) c[i] = 0xC4; c[219] = '\0'; c[0] = 0xDA; c[26] = c[37] = c[50] = c[76] = c[87] = c[123] = c[136] = c[171] = 0xC2; c[218] = 0xBF;**

**stream << c << endl << (char)0xB3 << setw(25) << left << "Archive" << (char)0xB3 << setw(10) << left << "Size(arch)" << (char)0xB3 << setw(12) << left << "Date(arch)" << (char)0xB3 << setw(25) << left << "File" << (char)0xB3 << setw(10) << left << "Size"<< (char)0xB3 << setw(35) << left << "Path" << (char)0xB3 << setw(12) << left << "Date(file)" << (char)0xB3 << setw(34) << left << "Author" << (char)0xB3 << setw(46) << left << "Key words" << (char)0xB3 << endl;;**

**c[0] = 0xC3; c[26] = c[37] = c[50] = c[76] = c[87] = c[123] = c[136] = c[171] = 0xC5; c[218] = 0xB4;**

**for(int i = 0; i < ktl.items\_count; i++)**

**{**

**stream << c << endl << ktl.Catalog[i];**

**}**

**c[0] = 0xC0; c[26] = c[37] = c[50] = c[76] = c[87] = c[123] = c[136] = c[171] = 0xC1; c[218] = 0xD9;**

**stream << c << endl;**

**return stream;**

**}**

**};**

**#include <iostream>**

**#include <ctime>**

**#include <fstream>**

**#include <iomanip>**

**#include <string.h>**

**#include "classes.hpp"**

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**using namespace std;**

**void menu(void);**

**Text\_doc SetTextDoc();**

**File SetFile();**

**Person SetAuthor();**

**Date SetDate();**

**Archive\_item SetArchive();**

**//Шаблон функції для підрахунку середньої к-ті текстових док.**

**template <typename T>**

**double AvgTextDocCount(Katalog c, T count)**

**{**

**double sum = 0;**

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

**{**

**sum+=c[i].GetCount();**

**}**

**return sum/count;**

**}**

**//Шаблон функції для підрахунку середнього ступеня архівації**

**template <typename T>**

**double AvgArchCoeff(Katalog c, T count)**

**{**

**double sum = 0;**

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

**{**

**sum+=c[i].GetArchCoefficient();**

**}**

**return sum/count;**

**}**

**int main()**

**{**

**menu(); // запуск меню**

**return 0;**

**}**

**void menu(void)**

**{**

**char choise;**

**Katalog catalog;**

**bool Catalog\_created = false, BinEmpty = true;**

**cout << " Menu\n";**

**cout << "1.Create Catalog object\n2.Change \n3.Print Catalog\n4.Write to file\n5.Read from file\n6.Search\n7.Exit\n";**

**do**

**{**

**cout << "\nChoise: ";**

**cin >> choise;**

**flushall();**

**switch (choise)**

**{**

**case ('1'): { // створює та заповнює каталог**

**char c;**

**do**

**{**

**cout << "Create (1 - default, 2 - own) object: ";**

**cin >> c;**

**}**

**while(c!='1' && c!='2');**

**if(c=='2')**

**{**

**int n;**

**cout << "Enter archive count (up to 50): ";**

**while(scanf("%d", &n)!=1 || n <=0 || n > 50)**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**Archive\_item\* ai = new Archive\_item[n];**

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

**{**

**cout << endl << "Creating " << i + 1 << " archive object" << endl;**

**ai[i] = SetArchive();**

**}**

**catalog = Katalog(ai, n);**

**Catalog\_created = true;**

**break;**

**}**

**else**

**{**

**catalog = Katalog();**

**Catalog\_created = true;**

**break;**

**}**

**}**

**case ('2'): { // редагування значень створеного каталогу**

**if(!Catalog\_created)**

**{**

**cout << "Catalog not created" << endl;**

**}**

**else**

**{**

**int n;**

**cout << "Input number of archive : ";**

**while(scanf("%d", &n)!=1 || n < 1 || n > catalog.GetArchCount())**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**catalog[n-1] = SetArchive();**

**}**

**break;**

**}**

**case ('3'): { // вивід у табличному режимі**

**if(Catalog\_created)**

**{**

**cout << catalog;**

**cout << endl << "Average value of text documents in archives = " << AvgTextDocCount(catalog, catalog.GetArchCount());**

**cout << endl << "Average archiviing coefficient = " << AvgArchCoeff(catalog, catalog.GetArchCount());**

**}**

**else cout << "Catalog not created" << endl;**

**break;**

**}**

**case ('4'): { // запис у файл**

**if(Catalog\_created)**

**{**

**char c;**

**do**

**{**

**cout << "Wtite to (1 - text, 2 - binary) file :";**

**cin >> c;**

**}**

**while(c!= '1' && c!= '2');**

**if(c == '1') catalog.WriteToTxt();**

**else**

**{**

**catalog.WriteToBin();**

**BinEmpty = false;**

**}**

**}**

**else cout << "Catalog not created" << endl;**

**break;**

**}**

**case ('5'): { // зчитування з файлу**

**char c;**

**do**

**{**

**cout << "Read from (1 - text, 2 - binary) file :";**

**cin >> c;**

**}**

**while(c!= '1' && c!= '2');**

**if(c == '1') catalog.ScanTxt();**

**else if(!BinEmpty)**

**{**

**catalog.ScanBin();**

**}**

**else cout << "Bin is empty. Press 4 to fill it" << endl;**

**break;**

**}**

**case ('6'): { // пошук в каталозі**

**char c;**

**do**

**{**

**cout << "Search by (1 - text, 2 - numeric) field :";**

**cin >> c;**

**}**

**while(c!= '1' && c!= '2');**

**if(c=='1') // по текстовову полю**

**{**

**char\* search = new char[30];**

**do**

**{**

**cout << "Enter text which you want to find (up to 30 symbols): ";**

**cin >> search;**

**}**

**while(strlen(search) > 30);**

**for(int i = 0; i < catalog.GetArchCount(); i++)**

**{**

**catalog[i].SearchInText(search);**

**}**

**}**

**else**

**{ // по числовому полю**

**double n;**

**cout << "Enter number which you want to find: ";**

**while(scanf("%lf", &n)!=1)**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**for(int i = 0; i < catalog.GetArchCount(); i++)**

**{**

**catalog[i].SearchInNumbers(n);**

**}**

**}**

**break;**

**}**

**case ('7'): return; // вихід із програми**

**default: {cout << "Uncorrect value" << endl; continue;}**

**}**

**}**

**while(1);**

**}**

**File SetFile() // створює, заповнює та повертає Файл**

**{**

**File f;**

**f.SetFileName();**

**f.SetFileExtention();**

**f.SetFileCreateDate();**

**f.SetSize();**

**return f;**

**}**

**Person SetAuthor() // створює, заповнює та повертає Автора**

**{**

**Person author;**

**author.SetName();**

**author.SetSurame();**

**author.SetBirthday();**

**return author;**

**}**

**Date SetDate() // створює, заповнює та повертає Дату**

**{**

**Date date;**

**date.SetDay();**

**date.SetMonth();**

**date.SetYear();**

**return date;**

**}**

**Text\_doc SetTextDoc() // створює, заповнює та повертає Текстовий документ**

**{**

**Text\_doc t;**

**t.SetFileName();**

**t.SetFileExtention();**

**t.SetFileCreateDate();**

**t.SetSize();**

**t.SetPath();**

**t.SetAuthor();**

**t.SetKeyWords();**

**return t;**

**}**

**Archive\_item SetArchive() // створює, заповнює та повертає Елемент архіву**

**{**

**Archive\_item ai;**

**File f = SetFile();**

**int doc\_count;**

**cout << "Enter count of text documets (up to 50): ";**

**while(scanf("%d", &doc\_count)!=1 || doc\_count <= 0 || doc\_count > 50);**

**{**

**cout << "Uncorrect value. Try again: ";**

**rewind(stdin);**

**}**

**Text\_doc\* td = new Text\_doc[doc\_count];**

**for(int i = 0; i < doc\_count; i++)**

**{**

**cout << endl << "Creating " << i + 1 << " Text document object" << endl;**

**td[i] = SetTextDoc();**

**}**

**ai = Archive\_item(f,td,doc\_count);**

**return ai;**

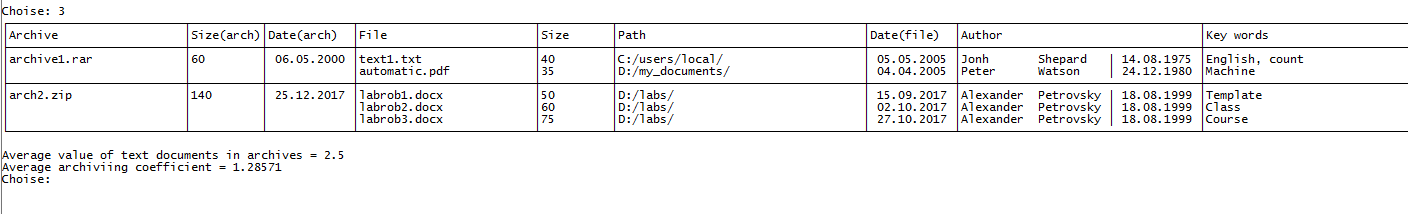
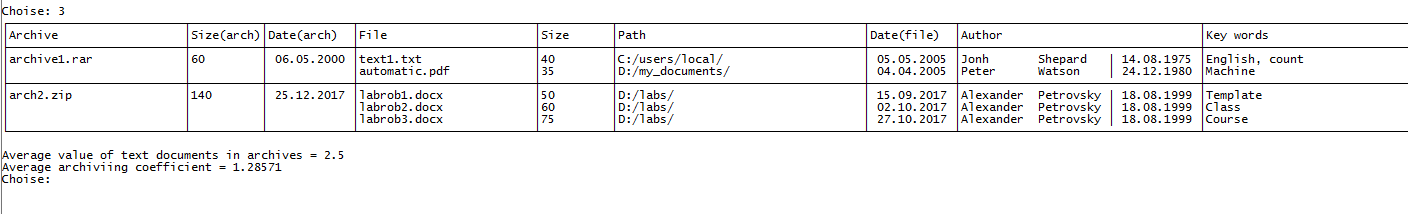
**}**

# ДОДАТОК 2. ОПИС ПРОГРАМИ

Меню програми пропонує користувачеві 7 пунктів, серед яких створення каталогу, зміна значення, вивід у табличному режимі, запис у текстовий або бінарний файл, зчитування із текстового або бінарного файлу, пошук за текстовими або числовими полями та вихід.



Програма рахує та виводить значення середньої кількості текстових документів та середнє значення ступеня архівації при кожному виводі всього каталогу у табличному вигляді.

Користувач сам контролює ввід даних під час заповнення каталогу або заповнення відбувається значеннями за замовчуванням.

