Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп’ютерних наук

Кафедра Системотехніки

**КУРСОВА РОБОТА**

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

з дисципліни Об’єктно-орієнтоване програмування

(назва дисципліни)

тема: Розробка об’єктно-орієнтованої програми «Залікова книжка»

(тема роботи)

Керівник 21.05.20 Решетнік В. М.

(підпис, дата, посада, прізвище, ініціали)

Студент КНТ-20-2 21.05.20 Швець В. Є.

(група, підпис, дата, прізвище, ініціали)

Робота захищена з оцінкою « »

« » 2021 р.

Комісія:

Решетнік В.М.

(підпис, посада, прізвище, ініціали)

Губаренко М.С.

(підпис, посада, прізвище, ініціали)

Чорна О.С.

(підпис, посада, прізвище, ініціали

Харків 2021

Харківський національний університет радіоелектроніки

**Факультет** комп'ютерних наук

**Кафедра** системотехніки

**Освітньо-професійна програма**  Комп'ютерні науки та технології

**Курс**   1  **група** КНТ-20-2 **семестр** 2

**ЗАВДАННЯ**

**на курсову роботу**

**студенту**                     Швець Вадим Євгенович

(прізвище, ім'я, по батькові)

**1. Тема роботи:** Розробка об’єктно-орієнтованої програмної системи «ЗАЛІКОВА КНИЖКА»

**2. Термін здачі студентом закінченої роботи** 15.05.2021

**3. Вихідні дані до проекту:** Розробити об’єктно-орієнтовану програмну систему «Залікова книжка» з ієрархією класів та підтримкою поліморфізму мовою програмування С++. Програмна система виконує функції, які визначені у варіанті завдання: назва предмета, кількість годин для вивчення предмета, дата проведення екзамену, оцінка. Операційна система – Windows 7 або вище, програмне забезпечення: інтегроване середовище MS Visual Studio, редактор UML-діаграм Software Ideas Modeler. Методичне забезпечення – методичні вказівки до курсової роботи.

**4. Зміст пояснювальної записки** **(перелік питань, які підлягають розробці)**: провести аналіз предметної області та визначити сутності, об’єкти, їх атрибути та функції; сформулювати та оформити вимоги до програмної системи; розробити об’єктну модель предметної області та сформувати словник; провести UML-моделювання, розробити діаграму класів (Class Diagram); реалізувати мовою С++ спроектовану програмну систему та описати її; розробити інтерфейс користувача; виконати тестування розробленої програми; доступу до БД; підготувати відповідно до ГОСТ 19.401-78 програмний документ «Текст програми».

**5. Перелік графічного матеріалу:** Схема об’єктної моделі, діаграма класів (Class Diagram), алгоритми, приклади екранних форм.

**6. Дата видачі завдання:** 01.04.2021

**Керівник роботи**   Решетнік Віктор Михайлович

(Підпис) (прізвище, ім'я, по батькові)

**Студент**   Швець Вадим Євгенович

(Підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсового проекту | Термін виконання | Примітка |
| 1 | Видача теми, узгодження і затвердження теми. | 01.04.2021 | виконано |
| 2 | Аналіз предметної області. | 15.04.2021 | виконано |
| 3 | Формулювання вимог до програми, розробка словника системи. | 15.04.2021 | виконано |
| 4 | Розробка діаграми класів. | 20.04.2021 | виконано |
| 5 | Розробка базового класу та похідних класів. | 22.04.2021 | виконано |
| 6 | Розробка інтерфейсу програми у вигляді текстового меню. | 26.04.2021 | виконано |
| 7 | Розробка функцій-членів класів, що призначені для встановлення значень полів класів СТУДЕНТ та ВИКЛАДАЧ. Перевірка працездатності функцій. | 30.04.2021 | виконано |
| 8 | Розробка функцій-членів класів , що призначені для видалення /додавання елементів класу ЕКЗАМЕН. Перевірка працездатності функцій. | 04.05.2021 | виконано |
| 9 | Розробка функцій-членів класу , що призначені для сортування, пошуку, ведення /виведення елементів класу ЗАЛІКОВА КНИЖКА, який містить масив об'єктів класу ЕКЗАМЕН. Перевірка працездатності функцій. | 08.05.2021 | виконано |
| 10 | Розробка головної функції main(). Перевірка працездатності системи. | 12.05.2021 | виконано |
| 11 | Тестування розробленого програмного забезпечення. | 14.05.2021 | виконано |
| 12 | Підготовка пояснювальної записки та її додатків. | 15.05.2021 | виконано |

**Керівник роботи**   Решетнік Віктор Михайлович

(Підпис) (прізвище, ім'я, по батькові)

**Студент**   \_\_ Швець Вадим Євгенович

(Підпис) (прізвище, ім'я, по батькові)

«  04 » квітня 2021 р.

# РЕФЕРАТ

Пояснювальна записка до курсової роботи: 26 с., 6 рис., 0 таблиць, 2 додатки, 5 джерел інформації.

КЛАСС, ПОЛІМОРФНИЙ КЛАСТЕР, ВІРТУАЛЬНА ФУНКЦІЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, ТЕСТУВАННЯ, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Мета роботи — розробка програмної системи «Залікова книжка» з використанням парадигм об’єктно-орієнтованого програмування.

Об’єкт розробки — програмна система «Залікова книжка».

Предмет розробки — інформаційні технології і програмні методи розробки об’єктно-орієнтованої програми.

Розробка використовує об’єктно-орієнтовані можливості мови програмування C++ та інтегрованого середовища розробки MS Visual Studio 2019.

Проведено аналіз предметної області, визначені основні завдання системи, опис вхідної та вихідної інформації, виконана діаграма класів, розроблено програмне забезпечення, що реалізує основні функції системи:

* запис об’єктів до файлу та вивід на консоль;
* встановлення значень полів об’єктів;
* отримання значень полів об’єктів;

Програма забезпечує швидку роботу з базою даних оцінок студентів/учнів.

Сфера застосування — навчальні заклади.

ЗМІСТ

ВСТУП……………………………………………………………………………6

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ………...8

* 1. Аналіз предметної області……………………………………………8
  2. Постановка задачі……………………………………………………..9

1. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ……………………………10
   1. Підсумки об’єктної декомпозиції……………………………………10
   2. Опис класів…………………………………………………………….11
      1. Опис інтерфейсу класу Student……………………………..11
      2. Опис інтерфейсу класу Vykladach………………………….14
      3. Опис інтерфейсу класу Ekzamen……………………………15
2. ПРОГРАМНА РЕАЛІЗАЦІЯ…………………………………………………18
3. ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА…………………………………….19

ВИСНОВОК………………………………………………………………………20

ПЕРЕЛІК ПОСИЛАНЬ…………………………………………………………..21  
Додаток А…………………………………………………………………………22

# ВСТУП

На сьогоднішній день існує багато підходів щодо написання комп’ютерних програм, або так званих парадигм програмування.

Існують такі парадигми програмування:

* процедурне програмування;
* функціональне програмування;
* структурне програмування;
* модульне програмування;
* об’єктно-орієнтоване програмування;

У рамках курсової роботи будемо розглядати об’єктно- орієнтованих підхід (ООП). Порівняно з другими парадигмами програмування, ця парадигма є новою. Історія розвитку ООП починається приблизно з 1960-х років. Активну участь у розвитку цього підходу брав Алан Кей.

Перш ніж ознайомлюватися з основними поняттями об’єктно- орієнтованого підходу нам потрібно зрозуміти що він собою являє.

Об'єктно-орієнтоване програмування (ООП) - це модель програмування, яка базується на твердженні, що програма є сукупністю об'єктів, які взаємодіють між собою. Кожен об'єкт у цій моделі є незалежним, і він може отримувати, обробляти дані та надсилати ці дані іншим об'єктам. У ООП використовуються моделі успадкування, модульності, поліморфізму та інкапсуляції.

Основне поняття ООП - це об’єкт. Об’єкт можна визначити як сукупність даних (характеристик об’єкта) та методів роботи з ними. Класи використовуються для класифікації об'єктів в ООП. Клас служить моделлю для створення об'єкта, тобто об'єкт є не що інше, як копією класу.

Кожен об'єкт має процедури та функції (що він може зробити, наприклад, завантажити файл, показати зображення тощо), які використовуються для роботи з даними об'єкта. Ці процедури та функції називаються методами.

Існування ООП можливе завдяки трьом основним парадигмам, на яких базується ООП:

* Інкапсуляція – один з трьох основних механізмів ООП. Йдеться про те, що об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді методів. А також про те, що доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи).
* Наслідування. механізм утворення нових класів на основі використання вже існуючих. При цьому властивості та функціональність батьківського класу переходять до класу нащадку.
* Поліморфізм – властивість, яка дозволяє одне і те саме ім’я використовувати для вирішення декількох технічно різних задач, тобто основною метою поліморфізму є використання одного імені для задання загальних класу дій.

Отримавши базові знання з ООП, стало зрозуміло, що такий підхід зручно застосовувати в ситуаціях, коли маємо справу з реальними об’єктами та різними абстрактними об’єктами. Об'єктно-орієнтований підхід, порівняно з іншими, справді полегшує життя програмісту і дає нам можливість реалізовувати масивні програми, над якими можуть працювати декілька програмістів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

* 1. Аналіз предметної області

У даній курсовій роботі перед нами постала задача розробки програми, яка реалізує залікову книгу, надаючи користувачеві змогу ввести інформації про залікову книжку, створити список екзаменів.

Залікова книга – документ, у якому містяться відомості про оцінки студента, його середній бал, захисти курсових робіт, здачі екзаменів тощо.

Для реалізації заданого проекту використано об’єктно-орієнтований підхід, для того щоб мати змогу визначити класи, об’єкти та зв’язки між ними. Такий підхід обраний невипадково, так як у нашому випадку потрібно описати книгу або залікову книжку, яка містить низку даних, які потрібні нам для їх виведення та здійснення операції над ними. Не важко здогадатися, що в залежності від книги чи залікової книжки ця інформація буде різна, тому нам потрібно розроблювати окремі об’єкти інформація яких відрізнялася б.

Проект міг би бути корисним для навчальних закладів, так як може здійснювати функціонал, корисний для таких установ. Що відходять від паперового ведення розрахунків, та переходять до альтернативного – електронного. І для цього потрібно використовувати таке програмне забезпечення, яке буде працювати стабільно та повністю задовольняти усі потреби, що можуть виникнути під час роботи.

За завданням потрібно створити ієрархію класів: Студент (Student), із атрибутами: номер группи, прізвище викладача. Викладач (Vykladach) з атрибутами: прізвище викладача, назва посади. Екзамент (Ekzamen) з атрибутами успадкованих вчителя та студента, а також такі додадкові атрибути: назва предмета, кількість годин, дата проведення екзамену, оцінка. Та реалізувати клас-контейнер Залікова книжка (ZalikovaKnyzhka), який містить масив об’єктів класу Екзамен із класом ітератором (iterator), за допомогою якого виконується управління індексацією масива об’єктів у класі-контейнері, а саме збільшення та зменшення індексу на деяке число та порівняння між собою елементів масиву за індексом за допомогою перевантажених операцій +,-,++,--,<,>,==,!=, оператор потокового виведення (<<) для виводу елементу масиву та оператор \* для отримання елементу масиву.

1.2 Постановка задачі

Для цієї курсової роботи постала задача розробити стабільну об’єктно-орієнтовану програму з використанням набутих навичок за курс об’єктно-орієнтованого програмування. Мінімальний набір задач, які потрібно було виконати, було наведено у завданні до курсової роботи.

Для розробки об’єктно-орієнтованої програми треба реалізувати сутності:

* Студент: прізвище, назва группи;
* Викладач: прізвище, назва посади;
* Екзамен: прізвище студента, назва группи, прізвище викладача, назва посади.
* Залікова книжка: книга із показниками усіх екзаменів, що в ній записані.

Об’єктну модель сутностей можна побачити на рисунку 1.1

У ході аналізу предметної області та тестуванні вже розроблених методів були сформульовані деякі додаткові вимоги до програми, такі як:

* реалізувати механізм відловлювання виключень;
* стабільність програми. Тобто розроблена програма має працювати з будь-яким набором вхідних даних і при цьому не повинні викликатися критичні помилки, що ведуть до екстреного завершення роботи;
* повинна бути реалізована коректна робота з кирилицею;
* програма має містити інтерфейс користувача для тестування основних функцій програми;
* програмний код має бути прокоментований;
* в програмі повинна бути використана бібліотека STL;
* повинен бути реалізований клас-контейнер, та ітератор для роботи з масивом підприємств, які користуються електролічильниками.

Усі розроблені об’єкти ведуть себе стабільно і не викликають критичну помилку у програмі, навіть за умови введення некоректних даних. Найімовірніший процес у якому розроблені об’єкти можуть брати участь — це використовування їх для роботи з базою даних екзаменів.

2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Підсумки об’єктної декомпозиції

Після проведення аналізу предметної області та складеного словника іменників області було прийнято рішення розділити програму на 4 класи, у яких виділені лише основні характеристики об’єктів, абстрагуючись від зайвого (рис. 2.7). Та клас-контейнер ZalikovaKnyzhka для роботи з масивом екзаменів:

* Student — клас базовий, що має поле prizv, що відповідає за значення прізвища студента і поле gruppa, що відповідає за значення группи студента;
* Vykladach — клас базовий, що має поле prizv, що відповідає за значення прізвища викладача і поле posada, що відповідає за значення посади викладача;
* Ekzamen — клас, що наслідує поля та методи класів Student та Vykladach та додатково містить поля nazvaPredmetu, kilkistGod, dataProv, otsinka;
* ZalikovaKnyzhka — клас-контейнер для масиву об’єктів класу Ekzamen, має поля \*array, n, що відповідають за зберігання масиву об’єктів Ekzamen, кількості елементів в масиві.

Архітектурний стиль програми було вирішено робити консольним з використанням консольного інтерфейсу для демонстрування основних функцій. Вхідні дані можуть вводитися користувачем через консоль з використанням клавіатури у певному порядку або з типізованого файлу. У якості носія для зберігання даних виступають текстові файли, так як з ними легко працювати і їх можливості повністю задовольняють поставленій задачі. Для користувача дані відображаються на консолі з вказанням того, які саме дані відображаються.

2.2 Опис класів

2.2.1 Опис класу Student

Клас студента у нашому проекті описує саме клас Student. Цей клас являється базовим для класу Ekzamen.

Тепер перейдемо до детальнішого опису цього класу. Клас Student містить захищені та паблік поля.

Серед захищених описано такі поля як:

* gruppa –– зберігає в собі значення типу string назви групи;
* prizvyshe\_stud –– зберігає в собі значення типу string прізвища студента;

Для зберігання змінних було обрано захищені поля для того, щоб при успападкуванні клас нащадок мав доступ до цих полів.

Також наш клас має відкриті поля у яких знаходяться методи, перевантаження операторів, конструктори та деструктори нашого класу, а саме:

* Student(); –– конструктор за замовчуванням;
* Student(string gruppa, string prizvyshe); –– конструктор з параметрами;
* Student(const Student& student); –– конструктор копіювання;
* ~Student(); –– деструктор;
* void OutputStudent(); –– метод для виводу полів даних студента;
* void InputStudent(); –– метод для вводу полів даних студента;
* virtual void OutputKlaster(); –– метод, за допомогою якого далі буде протестовано роботу поліморфічного кластера;
* Student& operator=(const Student& other); –– перевантаження операції присвоєння;
* friend ostream& operator<<(ostream& out, const Student& student); –– перевантаження операції виведення;
* friend istream& operator>>(istream& in, Student& student); –– перевантаження операції вводу;
* void set\_gruppa(string set\_gruppa); –– метод для встановлення значення групи;
* void set\_prizvyshe\_st(string set\_prizvyshe); –– метод для встановлення значення прізвища студента;
* string get\_gruppa(); –– метод для отримання значення групи;
* string get\_prizvyshe\_st(); –– метод для отримання значення прізвища;

Table

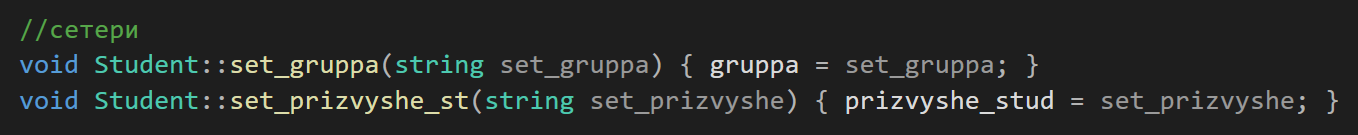
Description automatically generated

Рисунок 2.1 – Діаграма класу Student

Для кожного класу у якого є захищені або приватні поля ми мусимо написати сетери та гетери для цих полів. У нашого класу Student є по два сетери та гетери. Сетери та гетери або, так звані, функції доступу являють собою короткі функції, задачею яких є отримання або встановлення значень закритих полів.

Сетери реалізовуються таким чином:

* Користувач встановлює в параметри функції сетера значення змінної, яку він хоче змінити;
* У процесі виконання функції сетера значення змінної-члена класу змінюється на значення, яке передав до параметра функції користувач.

Рисунок 2.2 – Приклад сеттерів у программі

Наступним етапом написання програми було створення конструкторів.

В конструкторі за замовчуванням у нас усім об’єктам, які створені через цей конструктор присвоюється одна й та ж інформація. Нижче на рисунку 2.3 буде наведено фрагмент коду, в якому можна буде побачити яка інформація присвоюється конкретним полям.

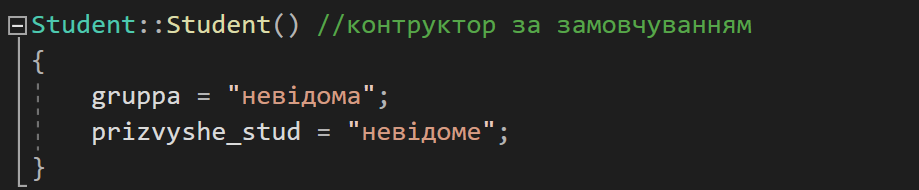


Рисунок 2.3 – Приклад конструктора за замовчуванням

Далі в классі було описано констуктори:

* Копіювання – в конструктор копіювання передається лише один об’єкт – це посилання на об’єкт класу, з якого ми будемо копіювати інформацію в поточний об’єкт.
* З параметрами ­­– конструктор з параметрами, який буде викликатися, якщо ми надамо значення.

Далі були створені методи класу. У методах InputStudent(), OutputStudent() немає майже нічого особливого. Вони розроблені для введення та виведення інформації про студента.

У класі Student були перевантажені операції введення, виведення полів данних і присвоєння об’єктів.

Нижче на рисунку 2.4 описано реалізацію перевантаженої операції присвоєння.

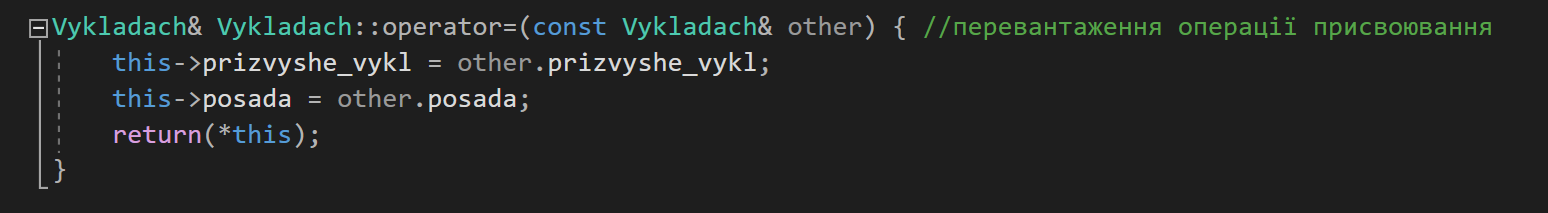


Рисунок 2.4 – Приклад перевантаження операції присвоєння

2.2.2 Опис класу Vykladach

Клас студента у нашому проекті описує саме клас Student. Цей клас також являється базовим для класу Ekzamen.

Тепер перейдемо до детальнішого опису цього класу. Клас Vykladach містить захищені та публічні поля.

Усі поля і методи, що описано в класі описано в діаграмі класу (рис. 2.5)

Серед захищених описано такі поля як:

* posada –– зберігає в собі значення типу string посади викладача;
* prizvyshe\_vykl –– зберігає в собі значення типу string прізвища студента;

Для зберігання змінних було обрано захищені поля для того, щоб при успападкуванні клас нащадок мав доступ до цих полів.

Також наш клас має відкриті поля у яких знаходяться методи, перевантаження операторів, конструктори та деструктори нашого класу, а саме:

* Vykladach(); –– конструктор за замовчуванням;
* Vykladach(string prizv, string pos); –– конструктор з параметрами;
* Vykladach(const Vykladach& vykladach); –– конструктор копіювання;
* ~Vykladach(); –– деструктор;
* void OutputVykladach(); –– метод для виводу полів даних викладача;
* void InputVykladach(); –– метод для вводу полів даних викладача;
* Vykladach& operator=(const Vykladach& other); –– перевантаження операції присвоєння;
* friend ostream& operator<<(ostream& out, const Vykladach& vykladach); –– перевантаження операції виведення;
* friend istream& operator>>(istream& in, Vykladach& vykladach); –– перевантаження операції вводу;
* void set\_posada(string set\_posada); –– метод для встановлення значення посади;
* void set\_prizvyshe\_vykl(string set\_prizvyshe); –– метод для встановлення значення прізвища викладача;
* string get\_posada(); –– метод для отримання значення посади;
* string get\_prizvyshe\_vykl(); –– метод для отримання значення прізвища викладача;

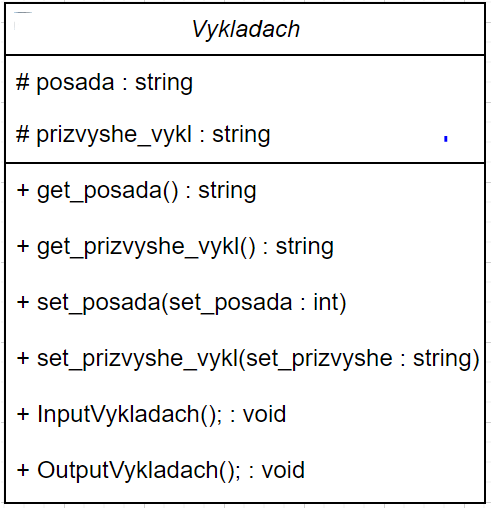


Рисунок 2.5 – Діаграма класу Vykladach

Повну реалізацію наведено в додатку А.

2.2.3 Опис класу Ekzamen

Клас «Ekzamen» (рис. 2.6) є нащадком класу «Student» і «Vykladach». Має два власних захищених поля:

* назва предмета;
* кількість годин для вивчення предмета;
* дата проведення екзамену;
* оцінка

Цей клас має методи встановлення та отримання значень його полів, функцію виводу на консоль, перевантаження операцій вводу, виводу і встановлення значень полів, також такі конструктори, як конструктор за замовчуванням, конструктор з параметрами і конструктор копіювання. Всі методи в классі мають публічний модифікатор доступу.

Перевантаження операції встановлення значень полів () дає змогу задати дані для екземпляру класу, який створено за допомогою конструктору за умовчанням.

Операція = дозволяє призначати значення полів з одного екземпляру на інший.

Перевантажено потокові операції введення з консолі та виведення на консоль. Вони надають можливість задати дані екземпляру класу та вивести екземпляр на консоль.

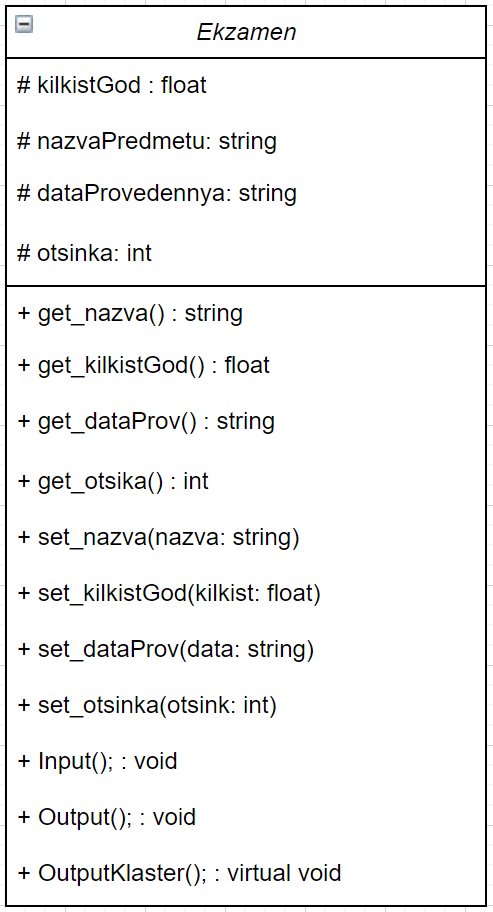


Рисунок 2.6 – Інтерфейс класу «Ekzamen»

Diagram

Description automatically generated

Рисунок 2.7 – Діаграма классів

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Перш за все, було вивчено предметну область та продумано класи, їх методи та алгоритми цих методів. Наступним було розроблено чотири хедерних файли: “Student.h”, “Vykladach.h”, “Ekzamen.h”, “ZalikovaKnyzhka.h”. Всі ці файли використані для опису інферфейсів класів, де описані їх змінні-члени класу та методи.

Також було розроблено створено шість .cpp файлів: “MainFile.cpp”, “Student.cpp”, “Vykladach.cpp”, “Ekzamen.cpp”, “ZalikovaKnyzhka.cpp”. У файлі “MainFile.cpp” виконується головна функція main(), де проводиться створення об’єктів та перевірка їх методів та перевантажень, а всі інші файли описують реалізацію методів класів.

Наступним кроком було створено інтерфейси клас “ZalikovaKnyzhka.h”, розроблений клас-контейнер з включеним класом-ітератором та були описані їхні методи, перевантаження операторів введення та виведення об’єктів, присвоєння та операцію для встановлень значень полів.

І останнім було створення меню у головній функції main(), для того щоб користувачеві було зручно користуватися програмою.

Загалом, було спроектовано та реалізовано класи. Програмний код наведений у додатку А. Забезпечена стійкість програми, тобто програма не втрачає працездатність за жодних, навіть некоректних, дій користувача. Користувач не має змогу встановити значення полів класів такими, які не передбачені під час розробки цих класів. Функціональність кожного з розроблених класів задовольняє усі початкові вимоги.

В програмi полiморфiчний кластер реалізован на класах Student та Ekzamen з вiртуальним методом OutputKlaster(), який виводить поточне значення лічильника. Вiн переписується класом Ekzamen. В iнтерфейсi програми кластер реалiзується вказивником типу Student. При виборi використання вiртуального методу OutputKlaster() у екземпляра класа Student, то вказiвнику присвоюється екземпляр класу DigitalCounter та викликається сам метод. А при виборi використання вiртуального методу OutputKlaster() у екземпляра класа Ekzamen, то вказiвнику присвоюється екземпляр класу Ekzamen та викликається сам метод. Реалізацію цих методів можна побачити на рисунках 3.1 та 3.2

4 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

Програма розбита на три пункти головного меню:

* Повна перевірка класів (викладач, студент, екзамен);
* Демонстрація роботи поліморфічного класу;
* Робота з контейнером 'Залікова книжка'.

В головному меню програми користувач може вибрати, з якою інформацією працювати та, за бажанням, закрити програму.

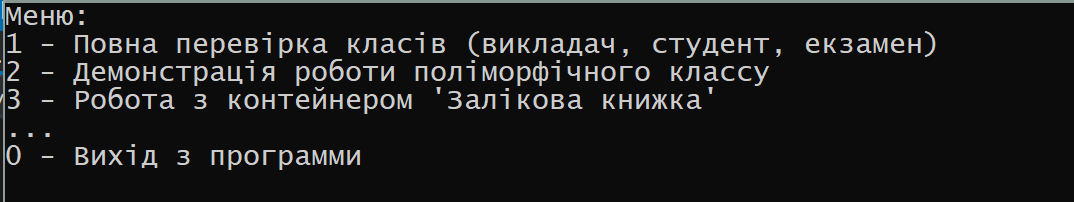


Рисунок 4.1 – Головне меню програми

Після вибору відповідного пункту перед користувачем з’являється меню повної перевірки класів, демонстрація роботи поліморфічного кластеру, або меню роботи з контейнером «Залікова книжка». Вигляд цих вкладок меню подано на рисунках 4.2. та 4.3.

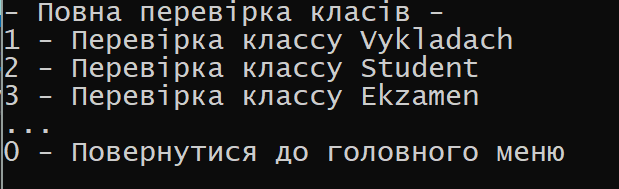


Рисунок 4.2 – Меню перевірки класів

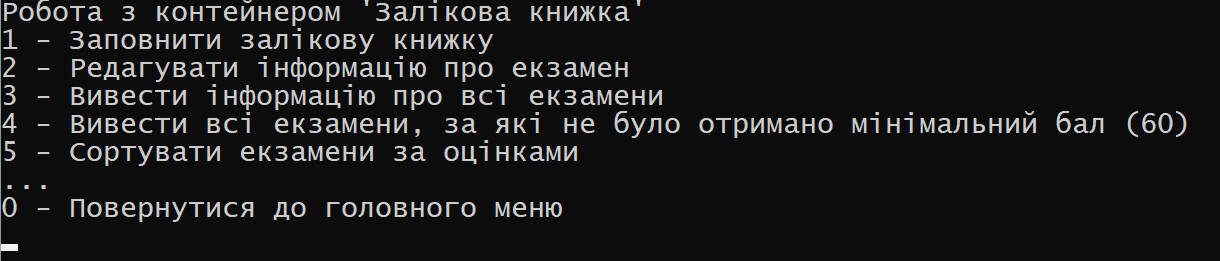


Рисунок 4.3 – Меню роботи з контейнерним класом

# ВИСНОВКИ

Під час написання курсової роботи було проаналізовано залікові книги, веб сайти, де описується ведення такої книги та декілька інших сайтів. З аналізу предметної області сформовані вимоги до програми.

Так як ООП програми можна розглядати як сукупність об’єктів, де об’єкт здатний отримувати повідомлення, обробляти дані та надсилати повідомлення іншим об’єктам, після проведення об’єктної декомпозиції з виділенням основних характеристик об’єктів, програма була розділена на 3 класи та клас-контейнер для зручної роботи з масивом екзаменів.

Визначений архітектурний стиль програми — консольний з використанням консольного інтерфейсу для демонстрування основних функцій.

Проведений детальний опис усіх методів класів. За допомогою UML діаграми продемонстровані та описані відношення між класами.

У ході програмної реалізації поставлені вимоги до програми були виконані. А саме:

* реалізований механізм відловлювання виключень;
* стабільність програми;
* реалізована коректна робота з кирилицею;
* програма містить інтерфейс користувача для тестування основних функцій програми;
* програмний код прокоментований;
* у програмі використана бібліотека STL;
* реалізований клас-контейнер, та ітератор для роботи з масивом екзаменів.
* Особлива увага приділена стабільності роботи програми. За результатами тестувань див. додаток Б.
* Розроблене програмне забезпечення можна використовувати при роботі з базою користувачі електролічильників.
* Для подальшого вдосконалення програми можна виконати наступне:
* зробити графічний інтерфейс;
* вирішити усі припущення, які були зроблені для спрощення програми, див. пункт 1.1.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Гріс Д. Наука програмування. - Г .: Мир. 1994.

2. Шилдт, Герберт. C++: руководство для начинающих, 2-е издание. :Пер. с англ./ Г.Шилдт. – М.: Издательский дом "Вильямс", 2005. – 672 с.

3. Грицюк Ю. ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ С++ / Ю. Грицюк, Т. Рак. – Львів: Вид-во ЛДУ БЖД, 2011. – 402 с.

4. Кравець П.О. Об'єктно-орієнтоване програмування: навч. посібник / П.О. Кравець. – Львів: Львівська політехніка, 2012. – 624 с.

5. Прата С. Язык программирования С++. Лекции и упражнения, 6-е изд.: Пер. с англ. М.: ООО "Изд. дом Вильямс", 2012. – 1248 с.

Додаток А

Міністерство освіти і науки України

Затверджую

Керівник курсової роботи,

Доцент кафедри системотехніки,

кандидат технічних наук,

старший науковий співробітник

\_\_\_\_\_\_\_\_\_\_\_\_В.М.Решетнік

(Підпис, дата)

ПРОГРАМНА СИСТЕМА «ЗАЛІКОВА КНИЖКА»

Текст програми

Аркуш затвердження

ГЮИК.508100.162 – 01 12 01 - ЛУ

Студент групи КНТ-20-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Швець Вадим Евгенович

(підпис, дата)

2021

Міністерство освіти і науки України

ПРОГРАМНА СИСТЕМА «ЗАЛІКОВА КНИЖКА»

Текст програми

ГЮИК.508100.162 9– 01 12 01

Аркушів 29

2021

Файл «Student.h»

#include <iostream>

#include <string>

using namespace std;

class Student

{

protected:

string gruppa;

string prizvyshe\_stud;

public:

string get\_gruppa();

string get\_prizvyshe\_st();

void set\_gruppa(string set\_gruppa);

void set\_prizvyshe\_st(string set\_prizvyshe);

Student(); //конструктор за замовчуванням

Student(string gruppa, string prizvyshe); //констуктор з параметрами

Student(const Student& student); //конструктор копіювання

//деструктори

~Student();

//методи

void OutputStudent();

void InputStudent();

virtual void OutputKlaster();

//перегрузки операцій

Student& operator=(const Student& other);

friend ostream& operator<<(ostream& out, const Student& student);

friend istream& operator>>(istream& in, Student& student);

};

Файл «Student.cpp»

#include "Student.h";

using namespace std;

//гетери

string Student::get\_gruppa() { return gruppa; }

string Student::get\_prizvyshe\_st() { return prizvyshe\_stud; }

//сетери

void Student::set\_gruppa(string set\_gruppa) {

gruppa = set\_gruppa;

}

void Student::set\_prizvyshe\_st(string set\_prizvyshe) { prizvyshe\_stud = set\_prizvyshe; }

Student::Student() //контруктор за замовчуванням

{

gruppa = "невідома";

prizvyshe\_stud = "невідоме";

}

Student::Student(string g, string p) //контруктор з параметрами

{

gruppa = g;

prizvyshe\_stud = p;

}

Student::Student(const Student& student) //контруктор копіювання

{

gruppa = student.gruppa;

prizvyshe\_stud = student.prizvyshe\_stud;

}

Student::~Student() //деструктор

{

}

void Student::OutputStudent() { //метод для виводу інформації про студента

cout << " Група: " << get\_gruppa() << endl;

cout << " Прізвище студента: " << get\_prizvyshe\_st() << endl;

}

void Student::InputStudent() { //метод для вводу полів данних студента

cout << " Група: ";

string grp;

cin >> grp;

set\_gruppa(grp);

cout << " Прізвище студента: ";

string przv;

cin >> przv;

set\_prizvyshe\_st(przv);

}

void Student::OutputKlaster() {

cout << " Прізвище студента: " << get\_prizvyshe\_st() << endl;

}

Student& Student::operator=(const Student& other) { //перевантаження операції присвоювання

this->gruppa = other.gruppa;

this->prizvyshe\_stud = other.prizvyshe\_stud;

return(\*this);

}

ostream& operator<<(ostream& out, const Student& student) { //перевантаження операції виводу

out << " Група: " << student.gruppa << endl;

out << " Прізвище студента: " << student.prizvyshe\_stud << endl;

return out;

}

istream& operator>>(istream& in, Student& student) { //перевантаження операції вводу

cout << "Група: ";

in >> student.gruppa;

cout << "Прізвище студента: ";

in >> student.prizvyshe\_stud;

return in;

}

Файл «Vykladach.h»

#include <iostream>

#include <string>

using namespace std;

class Vykladach {

protected:

string prizvyshe\_vykl;

string posada;

public:

//геттери

string get\_prizvyshe\_vykl();

string get\_posada();

//сеттери

void set\_prizvyshe\_vykl(string set\_prizvyshe);

void set\_posada(string set\_posada);

void OutputVykladach(); //метод для виводу інформації про викладача

void InputVykladach(); //метод для вводу полів данних викладача

Vykladach(); //конструктор за замовчуванням

Vykladach(string prizv, string pos); //конструктор з параметрами

Vykladach(const Vykladach& vykladach); //конструктор копіювання

//деструктори

~Vykladach();

//перегрузки операцій

Vykladach& operator=(const Vykladach& other);

friend ostream& operator<<(ostream& out, const Vykladach& vykladach);

friend istream& operator>>(istream& in, Vykladach& vykladach);

};

Файл «Vykladach.cpp»

#include "Vykladach.h";

using namespace std;

//гетери

string Vykladach::get\_prizvyshe\_vykl() { return prizvyshe\_vykl; }

string Vykladach::get\_posada() { return posada; }

//сетери

void Vykladach::set\_prizvyshe\_vykl(string set\_prizvyshe) { prizvyshe\_vykl = set\_prizvyshe; }

void Vykladach::set\_posada(string set\_posada) { posada = set\_posada; }

void Vykladach::OutputVykladach() { //метод для виводу інформації про викладача

cout << " Прізвище викладача: " << get\_prizvyshe\_vykl() << endl;

cout << " Посада: " << get\_posada() << endl;

}

void Vykladach::InputVykladach() { //метод для вводу полів данних викладача

cout << " Прізвище викладача: ";

string prizv;

cin >> prizv;

set\_prizvyshe\_vykl(prizv);

cout << " Посада: ";

string pos;

cin >> pos;

set\_posada(pos);

}

Vykladach::Vykladach() //контруктор за замовчуванням

{

prizvyshe\_vykl = "невідомо";

posada = "невідомо";

}

Vykladach::Vykladach(string prizv, string pos) //контруктор з параметрами

{

prizvyshe\_vykl = prizv;

posada = pos;

}

Vykladach::Vykladach(const Vykladach& vykladach) { //конструктор копіювання

prizvyshe\_vykl = vykladach.prizvyshe\_vykl;

posada = vykladach.posada;

}

Vykladach::~Vykladach() //деструктор

{

}

Vykladach& Vykladach::operator=(const Vykladach& other) { //перевантаження операції присвоювання

this->prizvyshe\_vykl = other.prizvyshe\_vykl;

this->posada = other.posada;

return(\*this);

}

ostream& operator<<(ostream& out, const Vykladach& vykladach) { //перевантаження операції виводу

out << " Прізвище викладача: " << vykladach.prizvyshe\_vykl << endl;

out << " Посада: " << vykladach.posada << endl;

return out;

}

istream& operator>>(istream& in, Vykladach& vykladach) { //перевантаження операції вводу

cout << "Прізвище викладача: ";

in >> vykladach.prizvyshe\_vykl;

cout << "Посада: ";

in >> vykladach.posada;

return in;

}

Файл «Ekzamen.h»

#include "Student.h"

#include "Vykladach.h"

using namespace std;

class Ekzamen : public Student, public Vykladach {

protected:

string nazvaPredmetu;

float kilkistGod;

string dataProvedennya;

int otsinka;

public:

string get\_nazva();

float get\_kilkistGod();

string get\_dataProv();

int get\_otsinka();

void set\_nazva(string nazva);

void set\_kilkistGod(float kilkist);

void set\_dataProv(string data);

void set\_otsinka(int otsink);

Ekzamen();

Ekzamen(string prizvStud, string grp, string prizvVykl, string pos, string nazvaP, float kilGod, string data, int ots);

Ekzamen(const Ekzamen& ekz);

~Ekzamen();

void Output();

void Input();

void OutputKlaster()override;

friend ostream& operator<<(ostream& out, const Ekzamen& ekz); //перевантаження операції виводу

friend istream& operator>>(istream& in, Ekzamen& ekz); //перевантаження операції вводу

};

Файл «Ekzamen.cpp»

#include "Ekzamen.h"

using namespace std;

string Ekzamen::get\_nazva() { return nazvaPredmetu; }

float Ekzamen::get\_kilkistGod() { return kilkistGod; }

string Ekzamen::get\_dataProv() { return dataProvedennya; }

int Ekzamen::get\_otsinka() { return otsinka; }

void Ekzamen::set\_nazva(string nazva) { nazvaPredmetu = nazva; }

void Ekzamen::set\_kilkistGod(float kilkist) { kilkistGod = kilkist; }

void Ekzamen::set\_dataProv(string data) { dataProvedennya = data; }

void Ekzamen::set\_otsinka(int otsink) { otsinka = otsink; }

Ekzamen::Ekzamen() {

prizvyshe\_stud = "невідомо";

gruppa = "невідома";

prizvyshe\_vykl = "невідомо";

posada = "невідома";

nazvaPredmetu = "невідома";

kilkistGod = 0;

dataProvedennya = "01.01.1970";

otsinka = 0;

}

Ekzamen::Ekzamen(const Ekzamen& ekz) {

prizvyshe\_stud = ekz.prizvyshe\_stud;

gruppa = ekz.gruppa;

prizvyshe\_vykl = ekz.prizvyshe\_vykl;

posada = ekz.posada;

nazvaPredmetu = ekz.nazvaPredmetu;

kilkistGod = ekz.kilkistGod;

dataProvedennya = ekz.dataProvedennya;

otsinka = ekz.otsinka;

}

Ekzamen::Ekzamen(string prizvStud, string grp, string prizvVykl, string pos, string nazvaP, float kilGod, string data, int ots) {

prizvyshe\_stud = prizvStud;

gruppa = grp;

prizvyshe\_vykl = prizvVykl;

posada = pos;

nazvaPredmetu = nazvaP;

kilkistGod = kilGod;

dataProvedennya = data;

otsinka = ots;

}

Ekzamen::~Ekzamen() {

}

void Ekzamen::Output() {

OutputStudent();

OutputVykladach();

cout << " Назва предмету: " << get\_nazva() << endl;

cout << " Кулькість годин: " << get\_kilkistGod() << endl;

cout << " Дата проведення: " << get\_dataProv() << endl;

cout << " Оцінка: " << get\_otsinka() << endl;

}

void Ekzamen::Input() {

cout << "Прізвище студента: ";

cin >> prizvyshe\_stud;

cout << "Група: ";

cin >> gruppa;

cout << "Прізвище викладача: ";

cin >> prizvyshe\_vykl;

cout << "Посада: ";

cin >> posada;

cout << "Назва предмету: ";

cin >> nazvaPredmetu;

cout << "Кулькість годин: ";

cin >> kilkistGod;

cout << "Дата проведення: ";

cin >> dataProvedennya;

cout << "Оцінка: ";

cin >> otsinka;

}

void Ekzamen::OutputKlaster() {

cout << " Прізвище студента: " << get\_prizvyshe\_st() << endl;

cout << " Назва предмету: " << get\_nazva() << endl;

}

ostream& operator<<(ostream& out, const Ekzamen& ekz) { //перевантаження операції виводу

out << " Прізвище студента: " << ekz.prizvyshe\_stud << endl;

out << " Група: " << ekz.gruppa << endl;

out << " Прізвище викладача: " << ekz.prizvyshe\_vykl << endl;

out << " Посада: " << ekz.posada << endl;

out << " Назва предмету: " << ekz.nazvaPredmetu << endl;

out << " Кулькість годин: " << ekz.kilkistGod << endl;

out << " Дата проведення: " << ekz.dataProvedennya << endl;

out << " Оцінка: " << ekz.otsinka << endl;

return out;

}

istream& operator>>(istream& in, Ekzamen& ekz) { //перевантаження операції вводу

cout << " Прізвище студента: ";

in >> ekz.prizvyshe\_stud;

cout << " Група: ";

in >> ekz.gruppa;

cout << " Прізвище викладача: ";

in >> ekz.prizvyshe\_vykl;

cout << " Посада: ";

in >> ekz.posada;

cout << "Назва предмету: ";

in >> ekz.nazvaPredmetu;

cout << "Кулькість годин: ";

in >> ekz.kilkistGod;

cout << "Дата проведення: ";

in >> ekz.dataProvedennya;

cout << "Оцінка: ";

in >> ekz.otsinka;

return in;

}

Файл «ZalikovaKnyzhka.h»

#include <iostream>

#include "Ekzamen.h"

using namespace std;

class ZalikovaKnyzhka {

private:

int n;

Ekzamen \*array;

public:

ZalikovaKnyzhka();

ZalikovaKnyzhka(int count);

~ZalikovaKnyzhka();

class Iterator {

Ekzamen\* ekzamen;

public:

Iterator(Ekzamen\* ekz);

Ekzamen& get();

bool operator>(const Iterator& it);

bool operator<(const Iterator& it);

Ekzamen& operator+(int n);

Ekzamen& operator-(int n);

Ekzamen& operator++(int);

Ekzamen& operator--(int);

Ekzamen& operator++();

Ekzamen& operator--();

bool operator!=(const Iterator& a);

bool operator==(const Iterator& a);

Ekzamen& operator\*();

Iterator& operator+=(int n);

Iterator& operator-=(int n);

};

Iterator beg();

Iterator end();

void Erase();

Ekzamen& operator[](int index);

int getLength();

void Resize(int newLength);

void InsertBefore(Ekzamen& value, int index);

void InsertAtTheEnd(Ekzamen& value);

void InsertAtTheBeginning(Ekzamen& value);

void Remove(int index);

void Sorting();

void OutputKnyzhka();

};

Файл «ZalikovaKnyzhka.cpp»

#include "ZalikovaKnyzhka.h";

#include <cassert>;

ZalikovaKnyzhka::ZalikovaKnyzhka() {

n = 0;

array = nullptr;

}

ZalikovaKnyzhka::ZalikovaKnyzhka(int count) : n(count) {

assert(count >= 0);

if (count > 0)

{

array = new Ekzamen[count];

}

else {

array = nullptr;

}

}

ZalikovaKnyzhka::~ZalikovaKnyzhka() {

delete[] array;

}

ZalikovaKnyzhka::Iterator::Iterator(Ekzamen\* ekz) { ekzamen = ekz; };

Ekzamen& ZalikovaKnyzhka::Iterator::get() { return \*ekzamen; }

bool ZalikovaKnyzhka::Iterator::operator>(const Iterator& it) { return ekzamen > it.ekzamen; }

bool ZalikovaKnyzhka::Iterator::operator<(const Iterator& it) { return ekzamen < it.ekzamen; }

Ekzamen& ZalikovaKnyzhka::Iterator::operator+(int n) { return\*(ekzamen + n); }

Ekzamen& ZalikovaKnyzhka::Iterator::operator-(int n) { return\*(ekzamen - n); }

Ekzamen& ZalikovaKnyzhka::Iterator::operator++(int) { return\*(ekzamen++); }

Ekzamen& ZalikovaKnyzhka::Iterator::operator--(int) { return\*(ekzamen--); }

Ekzamen& ZalikovaKnyzhka::Iterator::operator++() { return\*++ekzamen; }

Ekzamen& ZalikovaKnyzhka::Iterator::operator--() { return\*--ekzamen; }

bool ZalikovaKnyzhka::Iterator::operator!=(const Iterator& a) { return ekzamen != a.ekzamen; }

bool ZalikovaKnyzhka::Iterator::operator==(const Iterator& a) { return ekzamen == a.ekzamen; }

Ekzamen& ZalikovaKnyzhka::Iterator::operator\*() { return \*ekzamen; }

ZalikovaKnyzhka::Iterator& ZalikovaKnyzhka::Iterator::operator+=(int n) { ekzamen += n; return \*this; }

ZalikovaKnyzhka::Iterator& ZalikovaKnyzhka::Iterator::operator-=(int n) { ekzamen -= n; return \*this; }

ZalikovaKnyzhka::Iterator ZalikovaKnyzhka::beg() { return Iterator(array); }

ZalikovaKnyzhka::Iterator ZalikovaKnyzhka::end() { return Iterator(array + n); }

void ZalikovaKnyzhka::Erase() {

delete[] array;

array = nullptr;

n = 0;

}

Ekzamen& ZalikovaKnyzhka::operator[](int index) {

assert(index >= 0 && index < n);

return array[index];

}

int ZalikovaKnyzhka::getLength() { return n; }

void ZalikovaKnyzhka::Resize(int newLength) { //зміна розміру массима. всі існуючі елементи видаляються

if (newLength == n)

{

return;

}

if (newLength <= 0)

{

Erase();

return;

}

array = new Ekzamen[newLength];

n = newLength;

}

//фунція дозволяє додавати елемент перед елементом з індексом, що може ввести користувач

void ZalikovaKnyzhka::InsertBefore(Ekzamen& value, int index) {

assert(index >= 0 && index <= n); //перевірка коректності переданого індексу

Ekzamen\* data = new Ekzamen[n + 1]; //створюємо новий массив на один елемент більше, ніж старий

//копіюємо всі елементи аж до index'y

for (int before = 0; before < n; before++)

{

data[before] = array[before];

}

//вставляємо наш новий елемент в новий массив

data[index] = value;

//копіюємо всі значення після вставленого елемента

for (int after = index; after < n; after++)

{

data[after + 1] = array[after];

}

//видаляємо старий массив та використовуємо замість нього новий

delete[] array;

array = data;

n++;

}

void ZalikovaKnyzhka::InsertAtTheEnd(Ekzamen& value) {

InsertBefore(value, n);

}

void ZalikovaKnyzhka::InsertAtTheBeginning(Ekzamen& value) {

InsertBefore(value, 0);

}

void ZalikovaKnyzhka::Sorting() { //метод для сортування масиву

Ekzamen tmp;

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

for (int j = i + 1; j < n; j++) {

if (array[i].get\_otsinka() > array[j].get\_otsinka()) {

tmp = array[i];

array[i] = array[j];

array[j] = tmp;

}

}

}

}

void ZalikovaKnyzhka::OutputKnyzhka() {

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

{

cout << "- - - - - - - - - -" << endl;

cout << array[i] << endl;

cout << "- - - - - - - - - -" << endl;

}

}

Файл «MainFile.cpp»

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include "ZalikovaKnyzhka.h"

void Underline() {

cout << "- - - - - - - - - - - - -" << endl;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choise, perevirka, kontainer;

int countEkzams;

cout << "Введіть кількість екзаменів (елементів контейнеру): ";

cin >> countEkzams;

ZalikovaKnyzhka array(countEkzams);

cout << endl;

do {

system("cls");

cout << "Меню: " << endl;

cout << "1 - Повна перевірка класів (викладач, студент, екзамен)" << endl;

cout << "2 - Демонстрація роботи поліморфічного классу" << endl;

cout << "3 - Робота з контейнером 'Залікова книжка'" << endl;

cout << "...\n0 - Вихід з программи" << endl;

cout << endl;

cin >> choise;

switch (choise)

{

case 1: {

do {

system("cls");

cout << "- Повна перевірка класів -" << endl;

cout << "1 - Перевірка классу Vykladach" << endl;

cout << "2 - Перевірка классу Student" << endl;

cout << "3 - Перевірка классу Ekzamen" << endl;

cout << "..." << "\n0 - Повернутися до головного меню" << endl;

cout << endl;

cin >> perevirka;

switch (perevirka)

{

case 1: {

system("cls");

cout << "Тестування всьго классу Vykladach: " << endl;

Vykladach v1("Шевченко", "професор"); //создание экземпляра класса конструктором с параметрами

Vykladach v2; //создание экземпляра класса стандартным конструктором

Underline();

cout << "Створення екземпляру класу конструктором з параметрами" << endl;

cout << " !Виведення працює через метод классу" << endl;

v1.OutputVykladach();

Underline();

cout << "Створення екземпляру класу конструктором за замовчуванням" << endl;

v2.OutputVykladach();

Underline();

cout << "|Перевірка гетерів|" << endl;

cout << "Прізвище викладача 1: " << v1.get\_prizvyshe\_vykl() << endl;

cout << "Посада викладача 1: " << v1.get\_posada() << endl;

Underline();

cout << "|Перевірка сетерів|" << endl;

cout << "Заміна прізвища викладача 1 на будь-яке введене: ";

string prizv;

cin >> prizv;

v1.set\_prizvyshe\_vykl(prizv);

cout << "Заміна посади викладача 1 на 'доцент': " << endl;

v1.set\_posada("доцент");

cout << "\n Оновленна інформація студента1" << endl;

v1.OutputVykladach();

Underline();

cout << "|Перевірка перевантажених операцій|" << endl;

cout << " Oператор присвоювання: v2 = v1" << endl;

v2 = v1;

cout << "\nВиведення 1-го та 2-го викладача перевантаженим оператором виводу: " << endl;

cout << v1;

cout << v2;

cout << "\nСтворення 3-го викладача та заповнення його полів перевантаженим оператором вводу: " << endl;

Vykladach v3;

cin >> v3;

Underline();

cout << "\nСтворимо студента 4 та перевіремо метод для вводу його полів" << endl;

Vykladach v4;

v4.InputVykladach();

Underline();

cout << "\nВиведення 1-го, 2-го, 3-го та 4-го викладачів: " << endl << v1 << endl

<< v2 << endl << v3 << endl << v4 << endl;

system("pause");

break;

}

case 2: {

system("cls");

cout << "Тестування всьго классу Student: " << endl;

Student st1("КНТ-20-2", "Швець"); //создание экземпляра класса конструктором с параметрами

Student st2; //создание экземпляра класса стандартным конструктором

Underline();

cout << "Створення екземпляру класу конструктором з параметрами" << endl;

cout << " !Виведення працює через метод классу" << endl;

st1.OutputStudent();

Underline();

cout << "Створення екземпляру класу конструктором за замовчуванням" << endl;

st2.OutputStudent();

Underline();

cout << "|Перевірка гетерів|" << endl;

cout << "Прізвище студента 1: " << st1.get\_prizvyshe\_st() << endl;

cout << "Група студента 1: " << st1.get\_gruppa() << endl;

Underline();

cout << "|Перевірка сетерів|" << endl;

cout << "Заміна прізвища студента 1 на будь-яке введене: ";

string prizv;

cin >> prizv;

st1.set\_prizvyshe\_st(prizv);

cout << "Заміна групи студента 1 на 'КНТ-20-3': " << endl;

st1.set\_gruppa("КНТ-20-3");

cout << "\n Оновленна інформація студента1" << endl;

st1.OutputStudent();

Underline();

cout << "|Перевірка перевантажених операцій|" << endl;

cout << " Oператор присвоювання: st2 = st1" << endl;

st2 = st1;

cout << "\nВиведення 1-го та 2-го студента перевантаженим оператором виводу: " << endl;

cout << st1;

cout << st2;

cout << "\nСтворення 3-го студента та заповнення його полів перевантаженим оператором вводу: " << endl;

Student st3;

cin >> st3;

Underline();

cout << "\nСтворимо студента 4 та перевіремо метод для вводу його полів" << endl;

Student st4;

st4.InputStudent();

Underline();

cout << "\nВиведення 1-го, 2-го, 3-го та 4-го студентів: " << endl << st1 << endl << st2 << endl << st3 << endl << st4 << endl;

system("pause");

break;

}

case 3: {

cout << "Тестування всьго классу Ekzamen: " << endl;

//создание экземпляра класса конструктором с параметрами

Ekzamen ekz1("Труфанов", "КНТ-20-5", "Ізмаїлов", "професор", "МатАн", 50, "01.06.2021", 69);

Ekzamen ekz2; //создание экземпляра класса стандартным конструктором

cout << "Створення екземпляру класу конструктором з параметрами" << endl;

cout << " !Виведення працює через метод классу" << endl;

ekz1.Output();

cout << endl;

ekz2.Output();

Underline();

cout << "|Перевірка гетерів на класі, створеному конструктором за замовчуванням|" << endl;

cout << "Прізвище студента: " << ekz2.get\_prizvyshe\_st() << endl;

cout << "Група: " << ekz2.get\_gruppa() << endl;

cout << "Прізвище викладача: " << ekz2.get\_prizvyshe\_vykl() << endl;

cout << "Посада: " << ekz2.get\_posada() << endl;

cout << "Назва предмету: " << ekz2.get\_nazva() << endl;

cout << "Кулькість годин: " << ekz2.get\_kilkistGod() << endl;

cout << "Дата проведення: " << ekz2.get\_dataProv() << endl;

cout << "Оцінка: " << ekz2.get\_otsinka() << endl;

Underline();

cout << "|Перевірка сетерів на класі, створеному конструктором за замовчуванням|" << endl;

cout << "Заміна прізвища студента на будь-яке введене: ";

string prizv;

cin >> prizv;

ekz2.set\_prizvyshe\_st(prizv);

cout << "Заміна дати проведення на '21.06.2021': " << endl;

ekz2.set\_dataProv("21.06.2021");

cout << "Заміна назви предмету на будь-яку введену: ";

string nazva;

cin >> nazva;

ekz2.set\_nazva(nazva);

cout << "Заміна оцінки за екзамен на 78: " << endl;

ekz2.set\_otsinka(78);

cout << "\n Оновленна інформація екзамену" << endl;

ekz2.Output();

Underline();

cout << "|Перевірка перевантажених операцій|" << endl;

cout << " Oператор присвоювання: ekz3 = ekz2" << endl;

Ekzamen ekz3;

ekz3 = ekz2;

cout << "\nВиведення 2-го та 3-го екзаменів перевантаженим оператором виводу: " << endl;

cout << ekz2;

cout << ekz3;

cout << "\nСтворення 4-го екзамену та заповнення його полів перевантаженим оператором вводу: " << endl;

Ekzamen ekz4;

cin >> ekz4;

Underline();

cout << "\nЗамінемо дані 2-го екзамену за допомогою методу вводу" << endl;

ekz2.Input();

Underline();

cout << "\nВиведення 1-го, 2-го, 3-го та 4-го екхаменів: " << endl << ekz1 << endl << ekz2 << endl << ekz3 << endl << ekz4 << endl;

system("pause");

break;

}

case 0: {

continue;

}

default: {

cout << "Неіснуючий пункт меню!" << endl;

system("pause");

break;

}

}

} while (perevirka != 0);

case 0: {

break;

}

default: {

cout << "Неіснуючий пункт меню! " << endl;

system("pause");

break;

}

}

case 2: {

system("cls");

cout << "Тестування поліморфічного кластера:" << endl;

cout << "\nСтворюємо студента с прізвищем Шевченко" << endl;

Student st;

st.set\_prizvyshe\_st("Шевченко");

Underline();

st.OutputKlaster();

Underline();

cout << "Створюємо екзамен з параметрами конструктора за замовчуванням" << endl;

Ekzamen ekz;

Underline();

ekz.OutputKlaster();

Underline();

cout << "\n Створюємо посилання на клас студенту і виводимо його поля.\n Потім присвоюємо посиланню клас екзамен і ще раз виводимо його поля.\n" << endl;

Student\* stud = &st;

Underline();

stud->OutputKlaster();

stud = &ekz;

Underline();

stud->OutputKlaster();

cout << endl;

system("pause");

break;

}

case 3: {

do

{

system("cls");

cout << "Робота з контейнером 'Залікова книжка'" << endl;

cout << "1 - Заповнити залікову книжку" << endl;

cout << "2 - Редагувати інформацію про екзамен" << endl;

cout << "3 - Вивести інформацію про всі екзамени" << endl;

cout << "4 - Вивести всі екзамени, за які не було отримано мінімальний бал (60)" << endl;

cout << "5 - Сортувати екзамени за оцінками" << endl;

cout << "...\n0 - Повернутися до головного меню" << endl;

cin >> kontainer;

switch (kontainer)

{

case 1: {

cout << "Створити екзамен у залікову книжку: " << endl;

for (int i = 0; i < countEkzams; i++)

{

cout << "\nЕкзамен №" << i + 1 << endl;

cin >> array[i];

}

system("pause");

break;

}

case 2: {

cout << "Редагувати інформацію про екзамен: " << endl;

cout << "\n Редагувати екзамент №";

int edit;

cin >> edit;

array[edit].Input();

system("pause");

break;

}

case 3: {

cout << "Вивести інформацію про всі екзамени: " << endl;

for (int i = 0; i < countEkzams; i++)

{

cout << "\nЕкзамен №" << i << endl;

cout << array[i];

}

system("pause");

break;

}

case 4: {

cout << "Вивести всі екзамени, за які не було отримано мінімальний бал (60): " << endl;

for (int i = 0; i < countEkzams; i++)

{

if (array[i].get\_otsinka() < 60)

{

cout << "\nЕкзамен №" << i << endl;

cout << array[i];

}

}

system("pause");

break;

}

case 5: {

cout << "Сортувати екзамени за оцінками: " << endl;

array.Sorting();

for (int i = 0; i < countEkzams; i++)

{

cout << "\nЕкзамен №" << i << endl;

cout << array[i];

}

system("pause");

break;

}

case 0: {

break;

}

default: {

cout << "Неіснуючий пункт меню!" << endl;

system("pause");

continue;

}

}

} while (kontainer != 0);

}

}

} while (choise != 0);

cin.get();

return 0;

}