МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА АВТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ І УПРАВЛІННЯ

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Сайт. Онлайн новини»

Варіант №10

Студента 1 курсу групи ІС-02

Спеціальності 126 «ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

Керівник **доцент Крамар Ю.М.**

Кількість балів:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Члени комісії \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Київ-2021 рік

**Національний технічний університет України “КПІ ім. Сікорського”**

**Кафедра автоматизованих систем обробки інформації і управління**

ДисциплінаСтруктури даних та алгоритми

**Спеціальність 126 «Інформаційні системи та технології»**

**Курс 1 Група ІС-02 Семестр 2**

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| ***Коваля Кирила Назаровича*** |
| (прізвище, ім’я, по батькові) |

.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. Тема роботи: | | ***Розробка програмного забезпечення з використанням*** | | | |
| ***об’єктно-орієнтованого підходу.*** | | | | | |
|  | | | | | |
|  | |  | | | |
| 2. Строк здачі студентом закінченого проекту (роботи) | | | | | ***18.05.2021*** |
|  | |  | | | |
| 3. Вихідні дані до проекту (роботи): | | | | ***Сайт. Онлайн-новини. Додавання новин*** | |
| ***здійснюється зареєстрованими авторами.*** | | | | | |
|  | | | | | |
|  | | | | | |
|  | |  | | | |
| 4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають | | | | | |
| розробці): | ***1. Визначення класів та застосування всіх відношень*** | | | | |
| ***(реалізація, наслідування: інтерфейс – шаблонний клас-контейнер, класи-нащадки – рубрика новин (розділ сайту), зареєстровані акаунти, класи-нащадки рубрики новин – ігрова, філомова та музична рубрики; асоціація – структура дати; агрегація – класи даних користувача та публікації, що застосовані в класі «Сайт»; композиція – додаткова інформація конкретної публікації)*** | | | | | |
| ***2. Створення об’єктів та використання контейнерів (тип контейнера –вектор)*** | | | | | |
| ***3. Організація роботи з файлами.*** | | | | | |
| ***4. Пошук даних на сайті – за конкретними параметрами*** | | | | | |
| ***5. Функціонал авторизації/реєстрації користувача*** | | | | | |
|  | |  | | | |
| 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): | | | | | |
| ***діаграми класів в нотації UML*** | | | | | |
|  | |  | | | |
| 6. Дата видачі завдання: | | | ***01.04.2021*** | | |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Назва етапів курсової роботи** | **Термін виконання етапів роботи** | **Позначки керівника про виконання завдань** |
| 1. | Отримання теми курсової роботи | 01.04.2021 |  |
| 2. | Підготовка ТЗ | 08.04.2021 |  |
| 3. | Аналіз предметної області | 15.04.2021 |  |
| 4. | Проектування архітектури програмної системи | 21.04.2021 |  |
| 5. | Розробка сценарію роботи програми | 21.04.2021 |  |
| 6. | Узгодження з керівником інтерфейсу користувача | 25.04.2021 |  |
| 7. | Розробка програмного забезпечення | 01.05.2021 |  |
| 8. | Узгодження з керівником плану тестування | 14.05.2021 |  |
| 9. | Тестування програми | 15.05.2021 |  |
| 10. | Підготовка пояснювальної записки | 04.05.2021 |  |
| 11. | Здача курсової роботи на перевірку | 11.05.2021 |  |
| 12. | Захист курсової роботи | 18.05.2021 |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис)

**Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Коваль Кирило Назарович

(підпис) (прізвище ім'я, по батькові)

**«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.**

Зміст

Вступ 5

РОЗДІЛ 1. Аналіз предметної області 6

1.1 ОПИС ФУНКЦІОНАЛУ ПРОГРАМИ 8

1.1.1 Відображення головної сторінки/ігрової рубрики 8

[1.1.2 Завантаження даних сайту з файлів 8](#_Toc8948107)

[1.1.3 Відображення рубрики фільмів 8](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.4 Відображення рубрики музики 8](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.5 Відображення сторінки меню неавторизованого користувача для вибору функцій 9](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.6 Відображення сторінки меню авторизованого користувача для вибору функцій 9](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.7 Введення пункту меню (в цифровому проміжку, що передається у функцію параметром) 10](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.8 Сторінка створення нової новини 10](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.9 Сторінка авторизації 10](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.10 Сторінка реєстрації 10](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.11 Сторінка пошуку за певними параметрами 11](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.12 Метод завантаження новини на сайт 11](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.13 Метод перевірки на існування користувача 11](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.14 Метод підтвердження авторизації користувача 12](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.1.15 Метод підтвердження реєстрації 12](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[1.2 ПРИНЦИП РОБОТИ ПРОГРАМИ 13](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948089)

[1.3 РОЗДІЛ КОДУВАННЯ 23](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948089)

[РОЗДІЛ 2. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 24](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948090)

[2.1 СТРУКТУРА ТА ЛОГІКА ПРОГРАМИ 24](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948091)

[2.2.ОПИС ТА ПРИЗНАЧЕННЯ КЛАСІВ ПРЕДМЕТНОЇ ГАЛУЗІ](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948102) 24

[2.2.1 Клас «iContainer» 24](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[2.2.2 Клас «Data» 24](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[2.2.3 Структура «Date»](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107) 25

[2.2.4 Клас «Publication» 2](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)6

[2.2.5 Клас «Publicationinfo» 2](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)7

[2.2.6 Клас «Rubric» 2](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)7

[2.2.7 Клас «Site» 2](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)8

[2.3 ОПИС ТА ПРИЗНАЧЕННЯ КЛАСІВ ІНТЕРФЕЙСНОГО МОДУЛЮ 29](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948103)

[2.3.1 Клас «MessageType» 30](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[2.3.2 Клас «Alerts» 30](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[2.3.3 Клас «Session» 30](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

[2.3.4 Клас «Menu» 32](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948107)

РОЗДІЛ 3. [ТЕСТУВАННЯ 32](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948105)

[ВИСНОВКИ](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948128) 34

[ДОДАТОК А. Діаграма класів](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948130) 35

[ДОДАТОК Б. ЛІСТИНГ КОДУ](:///C:\Users\Fossa\AppData\Local\Temp\Rar$DRa2116.21121\.docx#_Toc8948131) 36

ВСТУП

Метою курсової роботи з дисципліни «Структури даних та алгоритми» є закріплення на практиці вміння:

- використовувати основні концепції об'єктно-орієнтованого підходу (ООП) — класи, інкапсуляцію, успадкування, поліморфізм, перевантаження методів і операцій, шаблони методів і класів, бібліотек один із найпотужніших інструментаріїв ООП (власних та STL –бібліотек С++);

- поглиблювати та узагальнювати знанння, які студент набув під час вивчення курсу «Структури даних та алгоритми» ;

- застосовувати студентами практичні навички в області створення сучасних програмних продуктів, використовуючи об’єктно-орієнтованих підхід;

- вибирати середовище проектування програмних засобів;

- проектувати, розробляти, налагоджувати та супроводжувати сучасне програмне забезпечення інформаційних та комп’ютерних систем.

Предметна галузь «Сайт. Онлайн-новини» є непоганим завданням на вміння застосовувати об’єктно-орієнтовані навички та вміння в програмуванні. Так, можна по-різному спроектувати класову архітектуру програми, що матиме різні поведінки та реалізації, тож поле для фантазії є неймовірно великим.

За предметною областю необхідно створити програмну реалізацію сайту онлайн новин, які розташовуються за певними рубриками, можливість їх перегляду, пошуку за різними параметрами, та можливість публікації нових новин (лише для авторизованих користувачів).

**ОСНОВНА ЧАСТИНА**

Розділ 1. Аналіз предметної області

Оскільки постає проблема створення ПО для сайту, необхідно передбачити його структуру.

Загально, програма розбита на два модулі – фронтенд (взаємодія з користувачем) та бекенд (логіка сайту, АРІ реалізація для фронтенд модуля).

Наш сайт має складатися з певних рубрик, а саме – «Ігри», «Фільми» та «Музика». На них будуть розміщені новини лише цих тематик, і користувач чітко розумітиме, де що знаходиться.

Далі необхідно подумати про те, що при запуску програми всі розділи будуть пустими – отже треба з цим щось робити. Під час створення сесії перегляду сайту, на нього буде завантажено весь контент, що міститься в попередньо заповненому файлі за певною адресою, та дані зареєстрованих користувачів – також у файлі. Для зручності зчитування даних з файлів треба заповняти їх за принципом маскування – позначка «/\*» казатиме програмі, що з цього місця починається набір даних для зчитування, позначка «\*» - про те, що набір даних для одного об’єкта завершено, та «\*/» - про те, що зчитування даних закінчено. Для зручності треба розташовувати кожне зчитуване поле на окремому рядку.

Для реалізації авторизування користувачів треба зробити клас-контейнер авторизаційних даних. Так, вони будуть інкапсулюватися в головному класі «Сайт», який буде контролювати їх життєвий цикл – приклад застосування композиції. Так, для наприклад логіну користувача на сайт, програмі треба буде звірити передані інтерфейсним модулем введені дані (логін та пароль) на збіжність в контейнері об’єктів користувацьких даних.

Той основний контент, яким буде заповнено сайт, - новини, є окремим класом, який називається «Публікація» (застосування агрегації). Він складається з полів, таких як «Зміст новини», «Дата публікації» та інша інформація про конкретну публікацію.

Ці об’єкти мають бути інкапсульованими в відповідній для них рубриці.

Вся логіка програми, «начинка» предметної області оформлена на кшталт АРІ проекту – об’єктно-орієнтованої програмної реалізації сайту з широким повним функціоналом для роботи з інтерфейсом та взаємодією з користувачем.

1.1 ОПИС ФУНКЦІОНАЛУ ПРОГРАМИ

1.1.1. *Відображення головної сторінки/ігрової рубрики:*

Коли: *під час запуску програми*

Ким: *конструктором класу «Session»*

Як: *методом фронтенд класу «Session» - «GamesPage», що виводить в консоль надпис адреси сторінки, привітання користувача, поточний час, інтерфейсні елементи управління, панель рубрик та власне новини ігрової рубрики*

1.1.2. *Завантаження даних сайту з файлів:*

Коли: *під час запуску програми*

Ким: *конструктором класу «Session»*

Як: *методом фронтенд класу «Session» - «loadData»* використовуючи об’єкти потокового класу зчитування з файлів ifstream, створити, заповнити та відправити в контейнери сайту об’єкти новин (зміст, інформація про новину) та користувацьких даних (акаунти – логіни та паролі)

1.1.3. *Відображення рубрики фільмів:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: *користувачем*

Як: *методом фронтенд класу «Session» - «FilmsPage», що виводить в консоль надпис адреси сторінки, привітання користувача, поточний час, інтерфейсні елементи управління, панель рубрик та власне новини рубрики фільмів*

1.1.4. *Відображення рубрики музики:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: *користувачем*

Як: *методом фронтенд класу «Session» - «MusicPage», що виводить в консоль надпис адреси сторінки, привітання користувача, поточний час, інтерфейсні елементи управління, панель рубрик та власне новини рубрики музики*

1.1.5. *Відображення сторінки меню неавторизованого користувача для вибору функцій:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: *неавторизованим* *користувачем*

Як: *методом фронтенд класу «Menu», що міститься в класі «Session» - «start», що виводить в консоль адресу сторінки та список доступних функцій та очікує від користувача вибору певної з них*

1.1.6. *Відображення сторінки меню авторизованого користувача для вибору функцій:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: *авторизованим користувачем*

Як: *методом фронтенд класу «Menu», що міститься в класі «Session» - «start», що виводить в консоль адресу сторінки та список доступних функцій та очікує від користувача вибору певної з них*

1.1.7. *Введення пункту меню (в цифровому проміжку, що передається у функцію параметром):*

Коли: *під час очікування вибору пункту меню користувачем*

Ким: *користувачем*

Як: статичним *методом фронтенд класу «Session» - «GetKeyPressed», що буде чекати від користувача натиснення клавіші на клавіатурі, відповідній числовому діапазону від однієї цифри до іншої, що відповідають доступним пунктам меню (наприклад, від 0 до 3). Поки користувач не натисне одну з цих клавіш, програма чекатиме далі. А коли натисне, функція поверне числове значення, відповідне натиснутій клавіші.*

1.1.8. Сторінка створення нової новини*:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: авторизованим *користувачем*

Як: *методом фронтенд класу «Session – «MakeNewPostPage», що спитає в користувача всі необхідні дані для створення нової публікації та завантажить її на сайт*

1.1.9. Сторінка авторизації*:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: *користувачем*

Як: *методом фронтенд класу «Session – «signIn», що спитає в користувача логін та пароль, і в разі успіху авторизує його.*

1.1.10. Сторінка реєстрації*:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: неавторизованим *користувачем*

Як: *методом фронтенд класу «Session – «signUp», що спитає в користувача логін та пароль, попросить повторити пароль,і в разі унікальності логіну зареєструє та авторизує його.*

1.1.11. Сторінка пошуку за певними параметрами*:*

Коли: *під час натисненням користувачем клавіші відповідній пункту меню*

Ким: *користувачем*

Як: *методом фронтенд класу «Session – «SearchPage», що спитає в користувача бажаний параметр для пошуку (певний період створених новин, пошук за певними тегами, автором / рубрикою) та відобразить знайдені на сайті дані. В разі порожнього пошуку нічого не виведе. Можна натиснути «0», щоб повернутись на головну сторінку*

1.1.12. Метод завантаження новини на сайт*:*

Коли: *під час відправлення інтерфейсним модулем пакету даних про новину*

Ким: *інтерфейсним модулем*

Як: *методом бекенд класу «Site» – «publish»,що створить новину на сайті у відповідній рубриці*

1.1.13. Метод перевірки на існування користувача:

Коли: *під час відправлення запиту інтерфейсним модулем*

Ким: *інтерфейсним модулем*

Як: *методом бекенд класу «Site» – «userExists», який відправить істинну відповідь у разі існування запитуваного логіну серед зареєстрованних акаунтів, і негативну у разі відсутності*

1.1.14. Метод підтвердження авторизації користувача:

Коли: *під час відправлення запиту інтерфейсним модулем*

Ким: *інтерфейсним модулем*

Як: *методом бекенд класу «Site» – «signIn», який відправить істинну відповідь у разі відповідності переданого логіну та паролю певному акаунту на сайті серед зареєстрованних, і негативну у разі відсутності співпадінь*

1.1.15. Метод підтвердження реєстрації:

Коли: *під час відправлення запиту інтерфейсним модулем*

Ким: *інтерфейсним модулем*

Як: *методом бекенд класу «Site» – «signUp», який перевірить логін на унікальність*

1.2 ПРИНЦИП РОБОТИ ПРОГРАМИ

Таким чином, можна сформувати сценарій роботи програми:

Завантаження даних сайту з файлів ⮚ Відображення головної ігрової сторінки ⮚ Можливість вибору:

1.2.1. Меню ⮚ Відображення меню ⮚ Вибір з:

* 1. Переходу на головну сторінку
* 2. Пошук новин за певною рубрикою
* 3. Пошук новин за тегами
* 4. Пошук новин за автором
* 5. Пошук новин за певним періодом
* 6. Авторизація
* 7. Реєстрація
* 0. Завершення програми

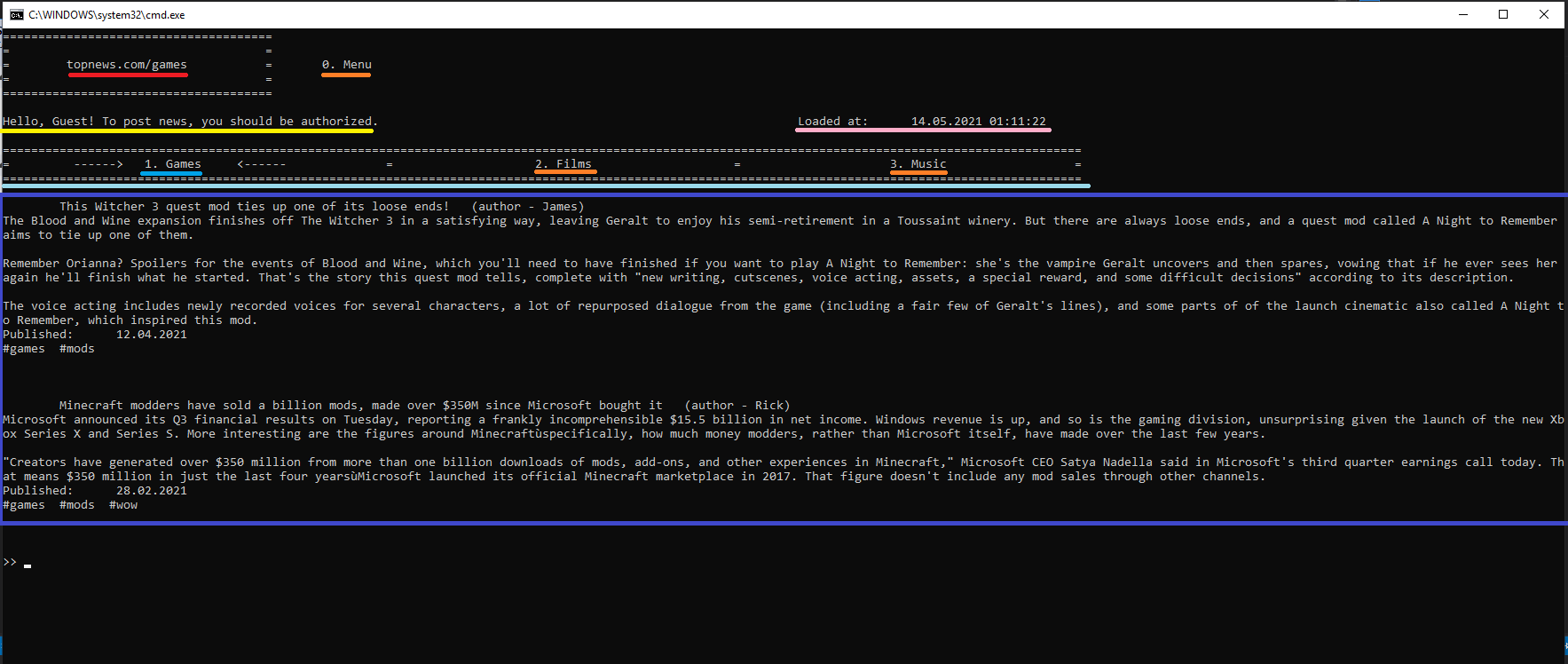
1.2.2. Ігрової сторінки ⮚ Відображення ігрової сторінки

1.2.3. Сторінки рубрики фільмів ⮚ Відображення рубрики фільмів

1.2.4. Сторінки музичної рубрики ⮚ Відображення музичної рубрики

Якщо ж користувач авторизувався/зареєструвався, вікно меню матиме для нього наступний функціонал:

* 1. Перехід на головну сторінку
* 2. Пошук новин за певною рубрикою
* 3. Пошук новин за тегами
* 4. Пошук новин за автором
* 5. Пошук новин за певним періодом
* 6. Публікація новини
* 0. Завершення програми

1.2.0. Опис інтерфейсу:

1.2.0.1. **Адреса сторінки**

1.2.0.2. **Елементи навігації**

1.2.0.3. **Дата, час завантаження сторінки**

1.2.0.4. **Панель рубрик**

1.2.0.5. **Привітання користувача**

1.2.0.6. **Область перегляду новин** Рисунок 1.2.1. Головна сторінка – рубрика ігор

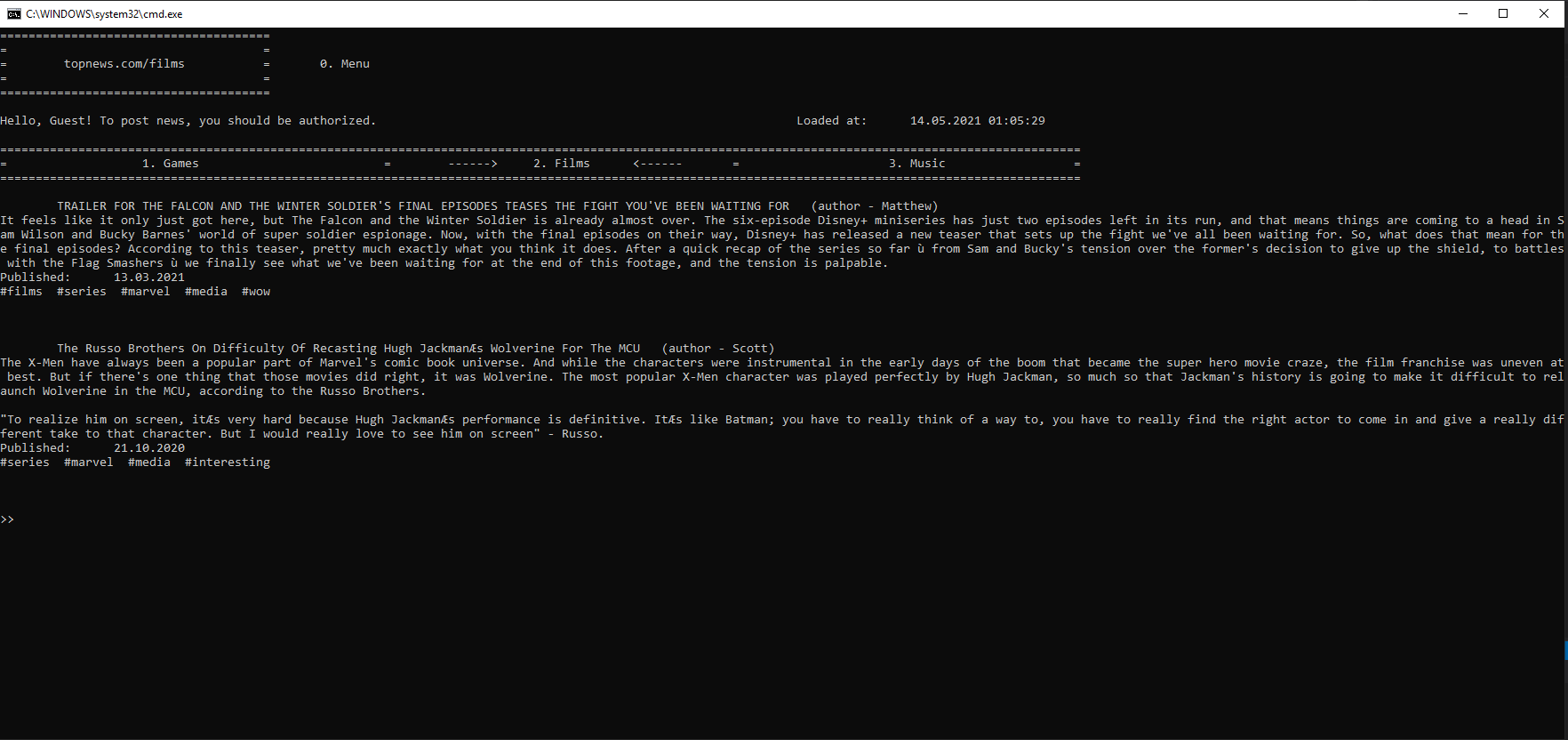
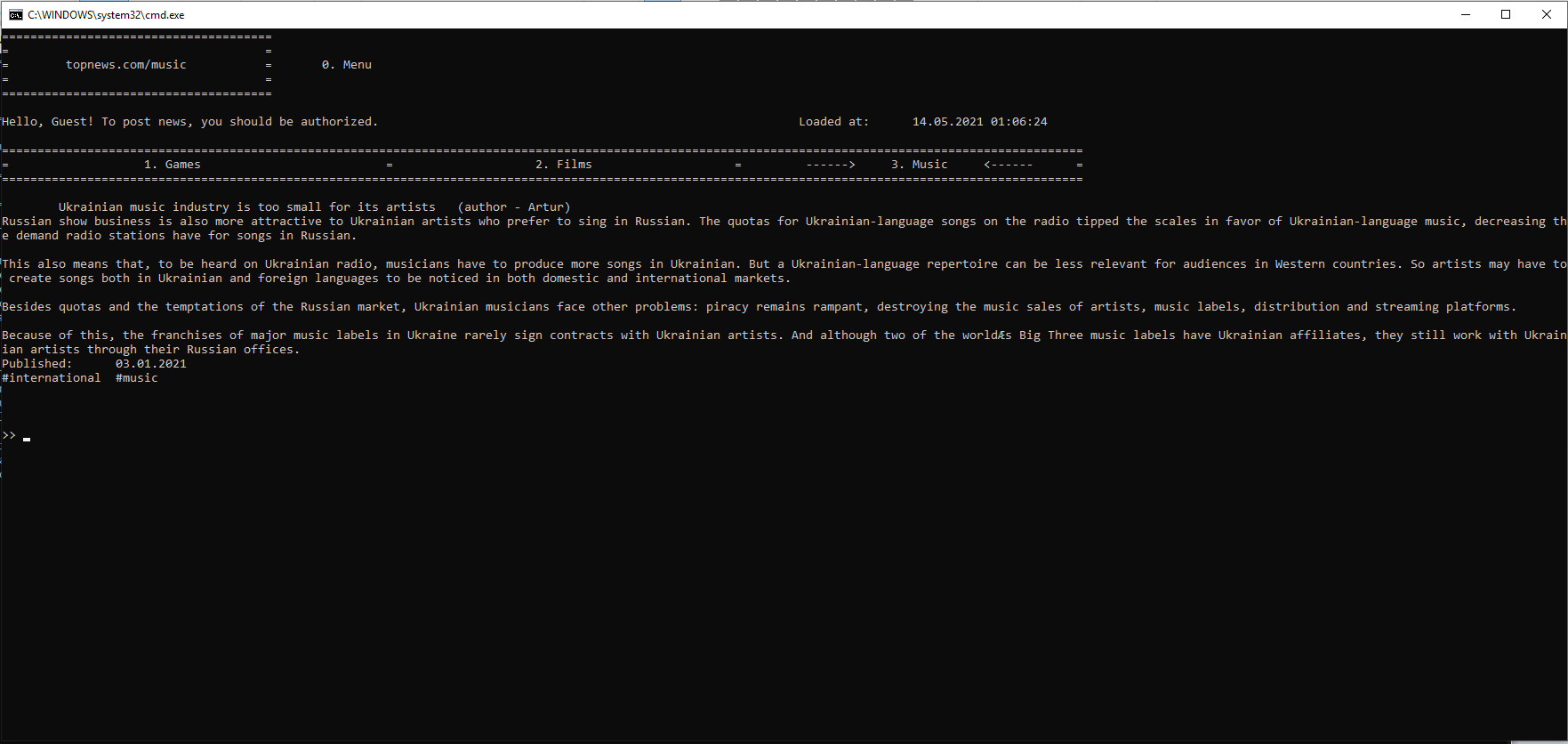


Рисунок 1.2.2. Сторінка рубрики фільмів

Рисунок 1.2.3. Сторінка рубрики музики



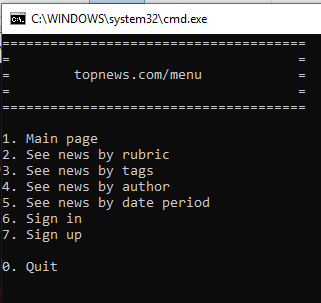


Рисунок 1.2.4. Меню неавторизованого користувача

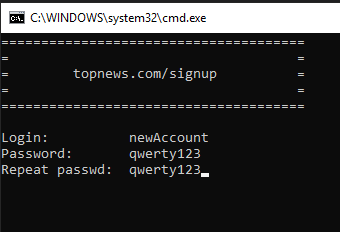


Рисунок 1.2.5. Вікно реєстрації

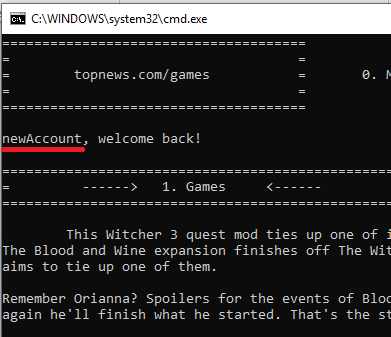
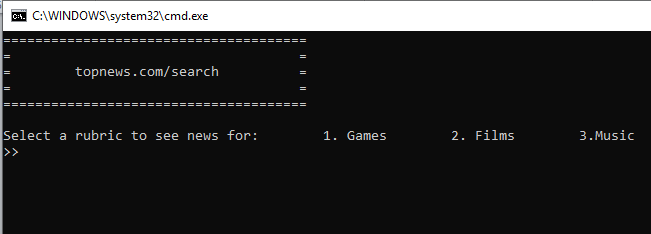
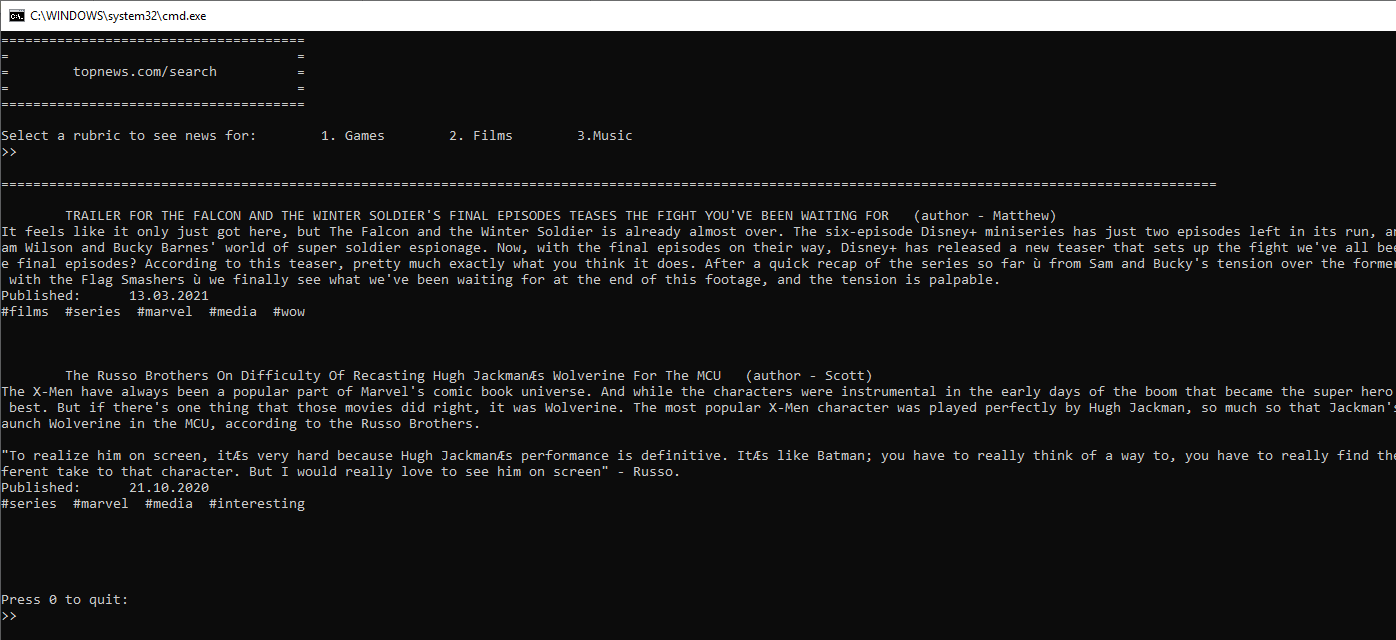


Рисунок 1.2.6. Тепер ми авторизований користувач

 Рисунок 1.2.7. Пошук новин за рубрикою

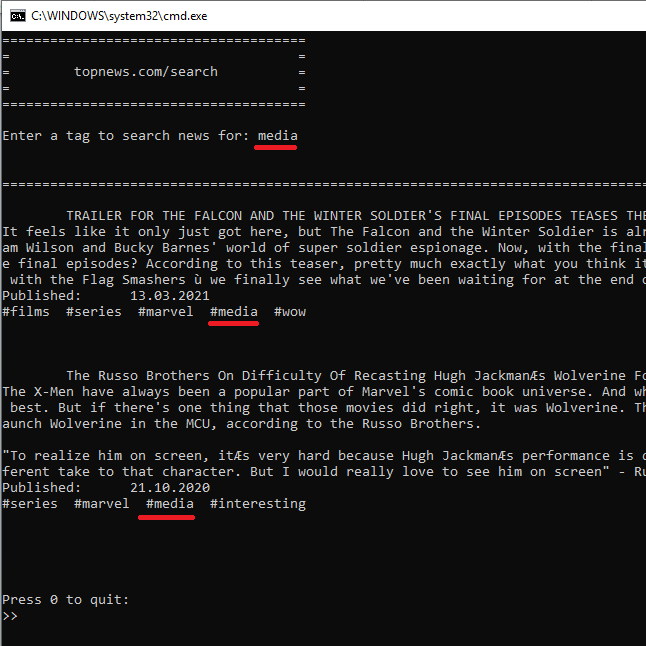
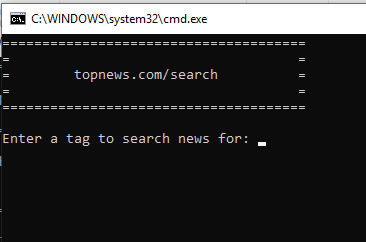


Рисунок 1.2.8., 1.2.9. Пошук новин за тегами

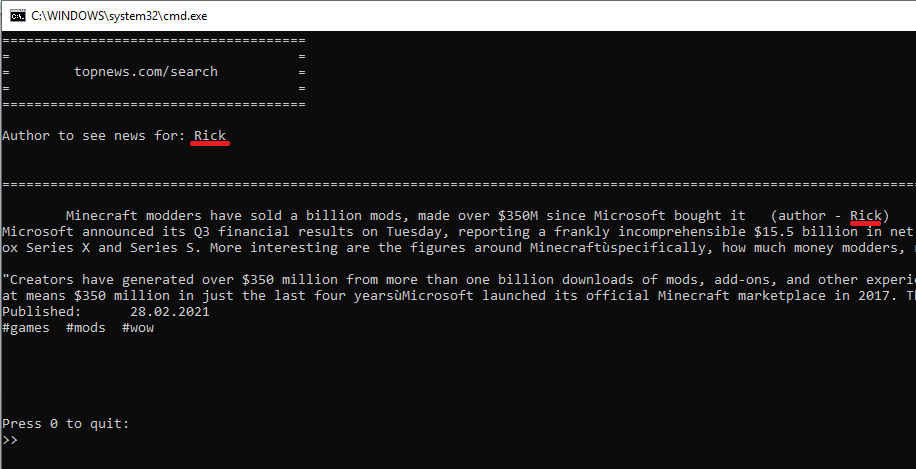
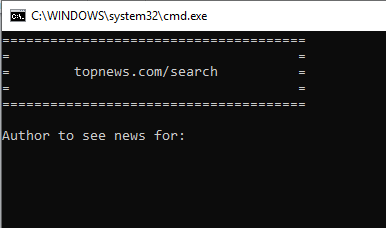


Рисунок 1.2.10., 1.2.11. Пошук новин за автором

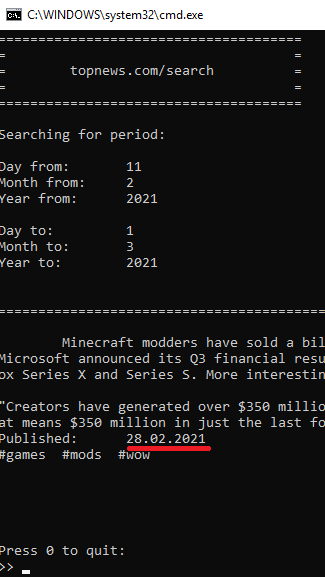


Рисунок 1.2.12. Пошук новин за певним періодом

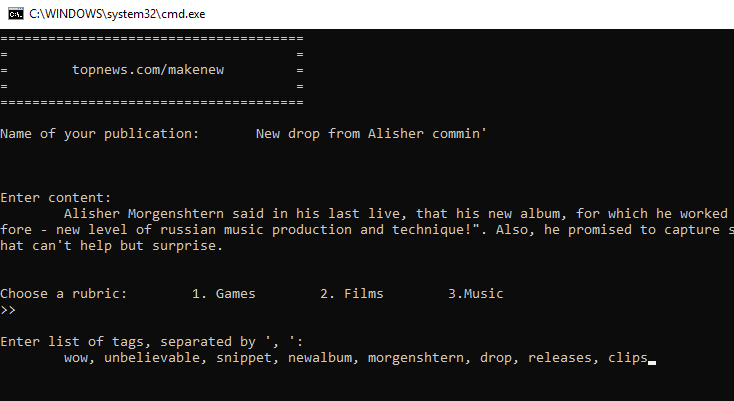


Рисунок 1.2.13., 1.2.14. Публікація новини

1.3. Розділ кодування

Під час створення програмного коду мовою С++ було використано:

1.3.1. Відношення між класами – реалізація, наслідування, асоціація, агрегація, композиція;

1.3.2. Основні концепції об'єктно-орієнтованого підходу (ООП) — класи, інкапсуляцію, успадкування, поліморфізм (статичний – напр. шаблони, динамічний – напр. віртуальні функції), перевантаження методів і операцій, шаблони класів, класів – контейнерів.

1.3.3. Один із найпотужніших інструментаріїв С++ - STL–бібліотеку, а саме такі її об’єкти, як вектори

1.3.4. Кортежі для швидкого й грамотного порівняння об’єктів структури «Date»

1.3.5. Роботу з файлами (бібліотека fstream)

1.3.6 Enum класи

Програму розбито за принципом модульного підходу – на .h заголовки та їх .срр реалізацію.

Лістинг програми знаходиться в Додатку Б.Розділ 2. Опис програмного забезпечення

СТРУКТУРА ТА ЛОГІКА РОБОТИ

2.1. Загальна структура

Діаграма класів наведена на окремому аркуші. (Див. додаток А)

2.2. Опис та призначення класів предметної галузі

2.2.1. Клас «iContainer»

Клас «iContainer» - це шаблонний клас інтерфейс, який містить чисто віртуальний метод “howManyInside”, який повертає кількість об’єктів типу Type в інкапсульованому векторі data. Використовується для зручного створення класів-контейнерів (приклад реалізації).

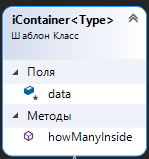


Рисунок 2.1

2.2.2. Клас «Data»

Клас «Data» - це клас авторизаційних даних користувача, що інкапсулює логін та пароль та містить методи перевірки коректності логіну/пароля. Використовується як тип шаблонного класу-контейнеру всіх акаунтів сайту «Accounts» (приклад агрегації), що наслідується від інтерфейсу “iContainer”

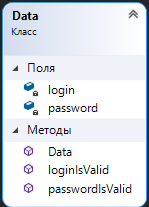


Рисунок 2.2

2.2.3. Структура «Date»

Структура «Date» складається з таких цілочислених полів, як «день», «місяць» та «рік». Використовується впродовж всього коду для зручності роботи з датами (приклад асоціації).

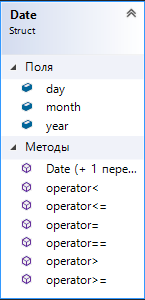


Рисунок 2.3

2.2.4. Клас «Publication»

Клас «Publication» описує публікацію на сайті. Інкапсулює вміст(текст) новини та інформацію про неї у вигляді об’єкта класу «PublicationInfo», що є складовою класу «Publication» (приклад композиції)

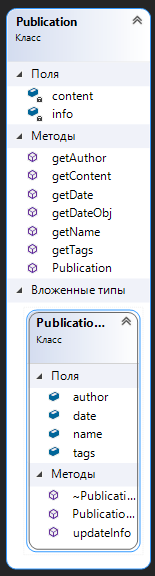


Рисунок 2.4

2.2.5. Клас «PublicationInfo»

Клас «PublicationInfo» описує додаткову інформацію про публікацію на сайті. Інкапсулює автора, дату публікації новини, її назву та набір тегів.

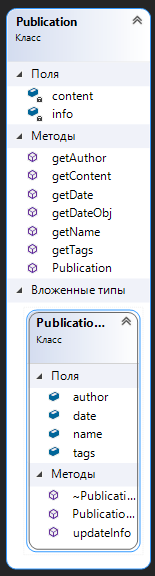


Рисунок 2.5

2.2.6. Клас «Rubric»

Клас-контейнер об’єктів класу “Publication” - «Rubric», що наслідує шаблонний клас-інтерфейс, описує взаємодію з новинами певної рубрики.

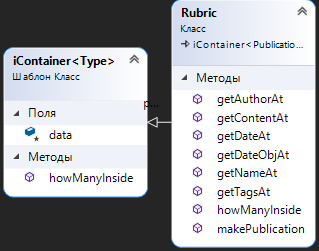


Рисунок 2.6

2.2.7. Клас «Site»

Головний клас бекенду – “Site” – інкапсулює об’єкти ігрового, фільмового та музичного класу рубрики, та об’єкт класу «Accounts». Містить методи взаємодії з сайтом: реєстрації, авторизації, перегляду новин, публікації нової новини та ін.

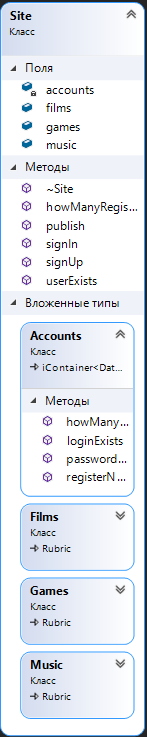


Рисунок 2.7

2.3. Опис та призначення класів інтерфейсного модулю

2.3.1. Enum - клас «MessageType»

Перерахування всіх кодів повідомлень, що можуть бути виведені на екран.

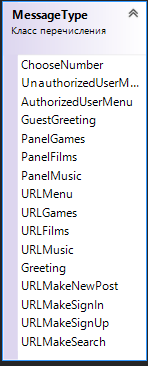


Рисунок 2.8

2.3.2. Клас «Alerts»

Клас з одним перевантаженим методом виводу повідомлень на екран (залежачи від переданого параметра MessageType)

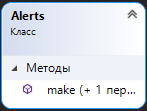


Рисунок 2.9

2.3.3. Клас «Session»

Головний клас фронтенду, відповідає за інтерфейс та взаємодію з користувачем. Інкапсулює об’єкт класу «Menu», що знаходиться всередині класу “Session”(композиція), об’єкт класу «Site» для взаємодії з ним, та поля authorised (bool) та user (string), що відповідають за поточного користувача.

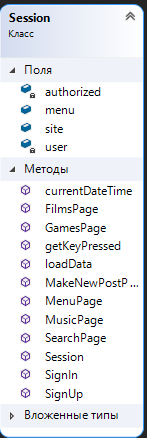


Рисунок 2.10

2.3.4. Клас «Menu»

Клас, інкапсульований в класі “Session”. Виводить меню для авторизованого/неавторизованого користувача.

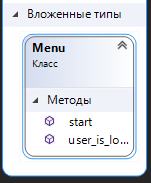


Рисунок 2.12

Розділ 3. ТЕСТУВАННЯ

На рисунках 1.2.4 – 1.2.14 зображено результати тестування усього функціоналу програми:

3.1. Тестування функції авторизації користувача (Рисунок 1.2.5, 1.2.6)

Дії: *обрати в меню пункт «Sign in» («Sign up»), ввести логін та пароль (повторити пароль)*

Очікуваний результат: *доступ до меню авторизованого користувача, привітання нашого імені*

Результат: *успішно*

3.2. Тестування функції пошуку новин за рубрикою (Рисунок 1.2.7)

Дії: *обрати в меню пункт «Search by rubric», обрати одну з рубрик (натисненням клавіші 1, 2 або 3)*

Очікуваний результат: *виведення на екран новин відповідної рубрики*

Результат: *успішно*

3.3. Тестування функції пошуку новин за автором (Рисунок 1.2.10., 1.2.11)

Дії: *обрати в меню пункт «Search by author», ввести бажаного автора*

Очікуваний результат: *виведення на екран новин відповідного автора*

Результат: *успішно*

3.4. Тестування функції пошуку новин за тегами (Рисунок 1.2.8., 1.2.9)

Дії: *обрати в меню пункт «Search by tags», ввести бажану тематику*

Очікуваний результат: *виведення на екран новин відповідної тематики*

Результат: *успішно*

3.5. Тестування функції пошуку новин за певним періодом (Рисунок 1.2.12)

Дії: *обрати в меню пункт «Search by period», ввести бажаний період*

Очікуваний результат: *виведення на екран новин відповідного періоду*

Результат: *успішно*

3.6. Тестування функції публікації новини (Рисунок 1.2.13., 1.2.14)

Дії: *обрати в меню пункт «Post a new», ввести необхідні дані*

Очікуваний результат: *збереження новини, виведення на екран*

Результат: *успішно*

3.7. Тестування функції навігації по сайту(перегляд новин різних рубрик, перехід до меню) (Рисунки 1.2.1 - 1.2.4)

Дії: *натиснути клавішу бажаного елементу керування вікном (0-3)*

Очікуваний результат: *перехід на обрану сторінку*

Результат: *успішно*

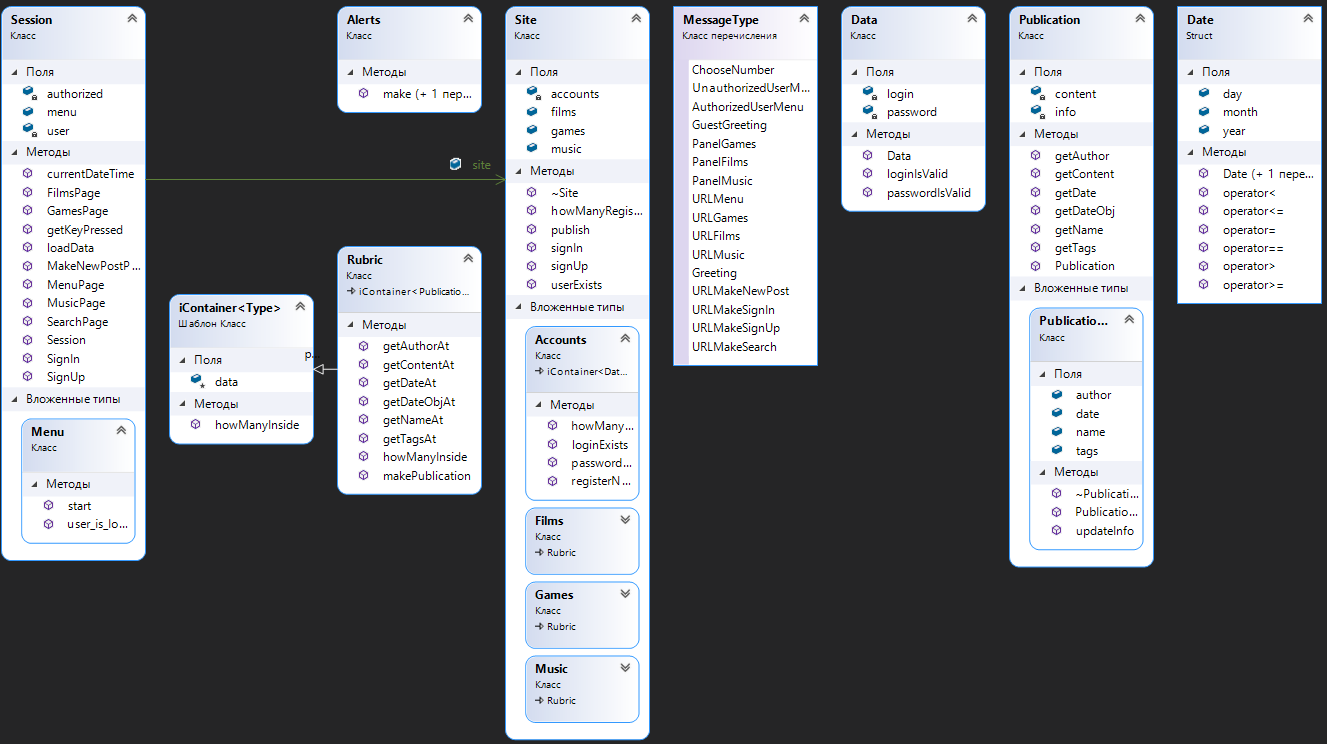
**ВИСНОВКИ**

У ході виконання курсової роботи було використано основні концепції об'єктно-орієнтованого підходу, шаблони методів і класів, STL –бібліотек С++; поглибено та узагальнено знанння, які було набуто під час вивчення курсу «Структури даних та алгоритми» ; застосовано практичні навички в області створення сучасних програмних продуктів, використовуючи об’єктно-орієнтованих підхід; обрано середовище проектування програмних засобів (Visual Studio); спроектовано, розроблено, налагоджено та супроводжено сучасне програмне забезпечення інформаційних та комп’ютерних систем.

Створено модель класів для предметної галузі «Сайт онлайн новин» з повним функціоналом, який вимагався, а саме: додавання новин здійснюється зареєстрованими авторами,  всі новини розташовуються за певними рубриками. Для новин формується набір тегів, за якими зручно шукати новини зазначеної тематики. Новини можуть вносити на сайт лише зареєстровані автори. Можливо передивлятися новини за рубрикою, тематикою, певного автора та за конкретний період.

Тестування додатку проводилось безпосередньо у меню користувача. На підставі отриманих результатів можна отримати висновок що програма працює коректно.

Реалізація програмного коду відбувалась у Microsoft Visual Studio 2019 на мові С++.

ДОДАТОК А. Діаграма класів

ДОДАТОК Б. Лістинг коду

source.cpp

#include "Backend.h"

#include "Frontend.h"

#include "Header.h"

using namespace std;

int main()

{

setConsoleWindow();

Session session;

return 0;

}

Header.h

#pragma once

void setConsoleWindow();

Header.cpp

#include "Header.h"

#include <windows.h>

void setConsoleWindow()

{

HANDLE out\_handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD crd = { 10, 10 };

SMALL\_RECT src = { 0, 0, crd.X - 1, crd.Y - 1 };

SetConsoleWindowInfo(out\_handle, true, &src);

system("mode con cols=220 lines=50");

SetConsoleScreenBufferSize(out\_handle, crd);

}

Alerts.h

#pragma once

#include "MessageType.h"

#include <string>

class Alerts

{

public:

static void make(MessageType code);

static void make(MessageType code, std::string name);

};

Alerts.cpp

#include "Alerts.h"

#include <iostream>

using namespace std;

void Alerts::make(MessageType code)

{

if (static\_cast<int>(code) < 0 || static\_cast<int>(code) > 15)

cout << "\nOops, there is no such text message.\n";

else

{

switch (code)

{

case MessageType::ChooseNumber:

cout << "Choose number.\n";

break;

case MessageType::UnauthorizedUserMenu:

cout << "1. Main page\n";

cout << "2. See news by rubric\n";

cout << "3. See news by tags\n";

cout << "4. See news by author\n";

cout << "5. See news by date period\n";

cout << "6. Sign in\n";

cout << "7. Sign up\n";

cout << "\n0. Quit\n";

break;

case MessageType::AuthorizedUserMenu:

cout << "1. Main page\n";

cout << "2. See news by rubric\n";

cout << "3. See news by tags\n";

cout << "4. See news by author\n";

cout << "5. See news by date period\n";

cout << "6. Post a new\n";

cout << "\n0. Quit\n";

break;

case MessageType::GuestGreeting:

cout << "Hello, Guest! To post news, you should be authorized.\t\t\t\t\t\t\t\t";

break;

case MessageType::PanelGames:

cout << "========================================================================================================================================================\n";

cout << "= ------> 1. Games <------ = 2. Films = 3. Music =\n";

cout << "========================================================================================================================================================\n";

break;

case MessageType::PanelFilms:

cout << "========================================================================================================================================================\n";

cout << "= 1. Games = ------> 2. Films <------ = 3. Music =\n";

cout << "========================================================================================================================================================\n";

break;

case MessageType::PanelMusic:

cout << "========================================================================================================================================================\n";

cout << "= 1. Games = 2. Films = ------> 3. Music <------ =\n";

cout << "========================================================================================================================================================\n";

break;

case MessageType::URLMenu:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/menu =\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::URLGames:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/games = 0. Menu\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::URLFilms:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/films = 0. Menu\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::URLMusic:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/music = 0. Menu\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::Greeting:

cout << "Welcome back!\t\t\t\t\t\t\t\t\t\t\t\t\t";

break;

case MessageType::URLMakeNewPost:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/makenew =\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::URLMakeSignIn:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/signin =\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::URLMakeSignUp:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/signup =\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

case MessageType::URLMakeSearch:

cout << "======================================\n";

cout << "= =\n";

cout << "= topnews.com/search =\n";

cout << "= =\n";

cout << "======================================\n\n";

break;

}

}

}

void Alerts::make(MessageType code, string name)

{

if (static\_cast<int>(code) != 11)

std::cout << "\nOops, there is no such text message.\n";

else

{

switch (code)

{

case MessageType::Greeting:

std::cout << name << ", welcome back!\t\t\t\t\t\t\t\t\t\t\t\t\t";

break;

}

}

}

Backend.h

#pragma once

#include "iContainer.h"

#include "Data.h"

#include "Date.h"

#include "Publication.h"

#include "Rubric.h"

#include <vector>

class Site

{

private:

class Accounts : public iContainer <Data>

{

public:

int howManyInside()

{

return data.size();

}

bool registerNew(std::string login, std::string password) //вернет 0, если такой аккаунт уже зарегистрирован, 1 - если успешно

{

for (int i = 0; i < (int)data.size(); i++)

if (data.at(i).loginIsValid(login))

return 0;

Data newAccount(login, password);

data.push\_back(newAccount);

return 1;

}

bool loginExists(std::string login)

{

if (login == "")

return 0;

else

{

for (int i = 0; i < (int)data.size(); i++)

{

if (data.at(i).loginIsValid(login))

return 1;

}

return 0;

}

}

bool passwordCheck(std::string login, std::string password)

{

if (!loginExists(login) || login == "" || password == "")

return 0;

else

{

for (int i = 0; i < (int)data.size(); i++)

{

if (data.at(i).loginIsValid(login))

{

if (data.at(i).passwordIsValid(password))

return 1;

else

return 0;

}

}

return 0;

}

}

};

class Games : public Rubric

{};

class Films : public Rubric

{};

class Music : public Rubric

{};

Accounts accounts;

public:

Games\* games = new Games();

Films\* films = new Films();

Music\* music = new Music();

int howManyRegistered();

bool signUp(std::string login, std::string password);

bool signIn(std::string login, std::string password);

bool userExists(std::string login);

bool publish(std::string \_name, std::string \_content, std::string \_author, std::string \_rubric, std::vector<std::string> \_tags, Date \_date);

~Site();

};

Backend.cpp

#include "Backend.h"

using namespace std;

int Site::howManyRegistered()

{

return accounts.howManyInside();

}

bool Site::signUp(string login, string password)

{

return accounts.registerNew(login, password);

}

bool Site::signIn(string login, string password)

{

if (accounts.loginExists(login) && accounts.passwordCheck(login, password))

return 1;

return 0;

}

bool Site::userExists(string login)

{

if (accounts.loginExists(login))

return 1;

return 0;

}

bool Site::publish(string \_name, string \_content, string \_author, string \_rubric, vector<string> \_tags, Date \_date)

{

if (\_rubric == "Games")

{

games->makePublication(\_name, \_content, \_author, \_tags, \_date);

return 1;

}

if (\_rubric == "Films")

{

films->makePublication(\_name, \_content, \_author, \_tags, \_date);

return 1;

}

if (\_rubric == "Music")

{

music->makePublication(\_name, \_content, \_author, \_tags, \_date);

return 1;

}

return 0;

}

Site::~Site()

{

delete games, films, music;

}

Data.h

#pragma once

#include <string>

class Data

{

public:

Data(std::string \_login, std::string \_password);

bool loginIsValid(std::string input);

bool passwordIsValid(std::string input);

private:

std::string login;

std::string password;

};

Data.cpp

#include "Data.h"

using namespace std;

Data::Data(string \_login, string \_password)

{

login = \_login;

password = \_password;

}

bool Data::loginIsValid(string input)

{

if (input == login)

return 1;

else

return 0;

}

bool Data::passwordIsValid(string input)

{

if (input == password)

return 1;

else

return 0;

}

Date.h

#pragma once

struct Date

{

public:

int day;

int month;

int year;

Date();

Date(int \_day, int \_month, int \_year);

void operator ()(int \_day, int \_month, int \_year);

void operator =(const Date& other);

bool operator >(const Date& right);

bool operator <(const Date& right);

bool operator >=(const Date& right);

bool operator <=(const Date& right);

bool operator ==(const Date& right);

};

Date.cpp

#include "Date.h"

#include <tuple>

#include <stdexcept>

using namespace std;

Date::Date()

{

day = 1;

month = 1;

year = 0;

}

Date::Date(int \_day, int \_month, int \_year)

{

try

{

if (\_day < 1 || \_day > 31 || \_month < 1 || \_month > 12)

throw invalid\_argument("Incorrect date format");

day = \_day;

month = \_month;

year = \_year;

}

catch (invalid\_argument e)

{

day = 1;

month = 1;

year = 0;

}

}

void Date::operator ()(int \_day, int \_month, int \_year)

{

day = \_day;

month = \_month;

year = \_year;

}

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

{

try

{

if (other.day < 1 || other.day > 31 || other.month < 1 || other.month > 12)

throw invalid\_argument("Incorrect date format");

this->day = other.day;

this->month = other.month;

this->year = other.year;

}

catch (invalid\_argument e)

{

}

}

bool Date::operator >(const Date& right)

{

return tie(this->year, this->month, this->day) > tie(right.year, right.month, right.day);

}

bool Date::operator <(const Date& right)

{

return tie(this->year, this->month, this->day) < tie(right.year, right.month, right.day);

}

bool Date::operator >=(const Date& right)

{

return tie(this->year, this->month, this->day) >= tie(right.year, right.month, right.day);

}

bool Date::operator <=(const Date& right)

{

return tie(this->year, this->month, this->day) <= tie(right.year, right.month, right.day);

}

bool Date::operator ==(const Date& right)

{

return tie(this->year, this->month, this->day) == tie(right.year, right.month, right.day);

}Frontend.h

#pragma once

#include "Alerts.h"

#include "Backend.h"

class Session

{

private:

class Menu

{

public:

int start()

{

system("cls");

Alerts::make(MessageType::URLMenu); //link

Alerts::make(MessageType::UnauthorizedUserMenu); //start\_menu

int answer = getKeyPressed(0, 7);

return answer;

}

int user\_is\_logged\_in()

{

system("cls");

Alerts::make(MessageType::URLMenu); //link

Alerts::make(MessageType::AuthorizedUserMenu); //user\_is\_logged\_in\_menu

int answer = getKeyPressed(0, 6);

return answer;

}

};

bool authorized = false; //авторизован пользователь или нет

std::string user = "";

public:

Menu menu;

Site site;

Session();

void MenuPage();

void GamesPage();

void FilmsPage();

void MusicPage();

void MakeNewPostPage();

void SignIn();

void SignUp();

void SearchPage(std::string what);

static int getKeyPressed(int from, int to);

void loadData();

const std::string currentDateTime();

};

Frontend.cpp

#include "Frontend.h"

#include <vector>

#include <string>

#include <iostream>

#include <fstream>

#include <conio.h>

#include <windows.h>

#include <time.h>

using namespace std;

Session::Session() //начальная страница - рубрика игр

{

loadData();

GamesPage();

}

void Session::MenuPage()

{

if (authorized)

{

int answer = menu.user\_is\_logged\_in();

switch (answer)

{

case 0:

exit(0);

break;

case 1:

GamesPage();

break;

case 2:

SearchPage("rubric");

break;

case 3:

SearchPage("tags");

break;

case 4:

SearchPage("author");

break;

case 5:

SearchPage("period");

break;

case 6:

MakeNewPostPage();

break;

}

}

else

{

int answer = menu.start();

switch (answer)

{

case 0:

exit(0);

break;

case 1:

GamesPage();

break;

case 2:

SearchPage("rubric");

break;

case 3:

SearchPage("tags");

break;

case 4:

SearchPage("author");

break;

case 5:

SearchPage("period");

break;

case 6:

SignIn();

break;

case 7:

SignUp();

break;

}

}

}

void Session::GamesPage()

{

system("cls");

Alerts::make(MessageType::URLGames); //link

if (authorized)

Alerts::make(MessageType::Greeting, user); //приветствие авторизованного пользователя

else

Alerts::make(MessageType::GuestGreeting); //приветствие гостя

cout << "Loaded at:\t" << currentDateTime() << "\n\n";

Alerts::make(MessageType::PanelGames); //панель рубрик

cout << "\n";

int maxPostsOnPage = 3; //максимальное количество одновременно отображаемых постов

int postsToPrint = site.games->howManyInside() > maxPostsOnPage ? maxPostsOnPage : site.games->howManyInside();

for (int id = 0; id < postsToPrint; id++)

{

cout << "\t" << site.games->getNameAt(id) << " (author - " << site.games->getAuthorAt(id) << ")\n";

cout << site.games->getContentAt(id) << "\n";

cout << "Published:\t" << site.games->getDateAt(id) << "\n";

for (int i = 0; i < (int)site.games->getTagsAt(id).size(); i++)

cout << "#" << site.games->getTagsAt(id).at(i) << " ";

cout << "\n\n\n\n";

}

cout << ">> ";

int answer = getKeyPressed(0, 3);

switch (answer)

{

case 0:

MenuPage();

break;

case 1:

GamesPage();

break;

case 2:

FilmsPage();

break;

case 3:

MusicPage();

break;

}

}

void Session::FilmsPage() //отобразить раздел новостей о фильмах

{

system("cls");

Alerts::make(MessageType::URLFilms); //link

if (authorized)

Alerts::make(MessageType::Greeting, user); //приветствие авторизованного пользователя

else

Alerts::make(MessageType::GuestGreeting); //приветствие гостя

cout << "Loaded at:\t" << currentDateTime() << "\n\n";

Alerts::make(MessageType::PanelFilms); //панель рубрик

cout << "\n";

int maxPostsOnPage = 3; //максимальное количество одновременно отображаемых постов

int postsToPrint = site.films->howManyInside() > maxPostsOnPage ? maxPostsOnPage : site.films->howManyInside();

for (int id = 0; id < postsToPrint; id++)

{

cout << "\t" << site.films->getNameAt(id) << " (author - " << site.films->getAuthorAt(id) << ")\n";

cout << site.films->getContentAt(id) << "\n";

cout << "Published:\t" << site.films->getDateAt(id) << "\n";

for (int i = 0; i < (int)site.films->getTagsAt(id).size(); i++)

cout << "#" << site.films->getTagsAt(id).at(i) << " ";

cout << "\n\n\n\n";

}

cout << ">> ";

int answer = getKeyPressed(0, 3);

switch (answer)

{

case 0:

MenuPage();

break;

case 1:

GamesPage();

break;

case 2:

FilmsPage();

break;

case 3:

MusicPage();

break;

}

}

void Session::MusicPage() //отобразить раздел новостей о музыке

{

system("cls");

Alerts::make(MessageType::URLMusic); //link

if (authorized)

Alerts::make(MessageType::Greeting, user); //приветствие авторизованного пользователя

else

Alerts::make(MessageType::GuestGreeting); //приветствие гостя

cout << "Loaded at:\t" << currentDateTime() << "\n\n";

Alerts::make(MessageType::PanelMusic); //панель рубрик

cout << "\n";

int maxPostsOnPage = 3; //максимальное количество одновременно отображаемых постов

int postsToPrint = site.music->howManyInside() > maxPostsOnPage ? maxPostsOnPage : site.music->howManyInside();

for (int id = 0; id < postsToPrint; id++)

{

cout << "\t" << site.music->getNameAt(id) << " (author - " << site.music->getAuthorAt(id) << ")\n";

cout << site.music->getContentAt(id) << "\n";

cout << "Published:\t" << site.music->getDateAt(id) << "\n";

for (int i = 0; i < (int)site.music->getTagsAt(id).size(); i++)

cout << "#" << site.music->getTagsAt(id).at(i) << " ";

cout << "\n\n\n\n";

}

cout << ">> ";

int answer = getKeyPressed(0, 3);

switch (answer)

{

case 0:

MenuPage();

break;

case 1:

GamesPage();

break;

case 2:

FilmsPage();

break;

case 3:

MusicPage();

break;

}

}

void Session::MakeNewPostPage()

{

system("cls");

Alerts::make(MessageType::URLMakeNewPost); //link

string name = "unknown", content = "unknown", author = "unknown", rubric = "Games";

vector<string> tags;

int day = 1, month = 1, year = 0;

string d = currentDateTime();

string day\_(""), month\_(""), year\_("");

day\_ += d.substr(0, 2);

month\_ += d.substr(3, 2);

year\_ += d.substr(6, 4);

day = stoi(day\_);

month = stoi(month\_);

year = stoi(year\_);

Date date(day, month, year);

author = user;

cout << "Name of your publication:\t";

cin.clear();

cin.ignore(8196, '\n');

getline(cin, name);

cout << "\n\nEnter content:\n\t";

getline(cin, content);

cout << "\n\nChoose a rubric:\t1. Games\t2. Films\t3.Music\n>> ";

int answer = getKeyPressed(1, 3);

switch (answer)

{

case 1:

rubric = "Games";

break;

case 2:

rubric = "Films";

break;

case 3:

rubric = "Music";

break;

}

cout << "\n\nEnter list of tags, separated by ', ':\n\t";

string temp(""), one\_tag("");

getline(cin, temp);

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

{

if (temp[i] == ',')

{

tags.push\_back(one\_tag);

one\_tag = "";

continue;

}

if (temp[i] == ' ')

continue;

if (i == temp.size() - 1)

{

one\_tag += temp[i];

tags.push\_back(one\_tag);

}

one\_tag += temp[i];

}

site.publish(name, content, author, rubric, tags, date);

GamesPage();

}

void Session::SignIn()

{

system("cls");

Alerts::make(MessageType::URLMakeSignIn); //link

string login(""), password("");

cout << "Login (or enter 0 to quit):\t";

cin >> login;

if(login == "0")

GamesPage();

cout << "Password:\t\t\t";

cin >> password;

if (site.signIn(login, password))

{

authorized = 1;

user = login;

GamesPage();

}

else

SignIn();

}

void Session::SignUp()

{

system("cls");

Alerts::make(MessageType::URLMakeSignUp); //link

string login(""), password(""), password1("");

cout << "Login:\t\t";

cin >> login;

cout << "Password:\t";

cin >> password;

cout << "Repeat passwd:\t";

cin >> password1;

if (password != password1)

SignUp();

if (site.signUp(login, password))

{

authorized = 1;

user = login;

GamesPage();

}

else

SignUp();

}

void Session::SearchPage(string what)

{

system("cls");

Alerts::make(MessageType::URLMakeSearch); //link

if (what == "rubric")

{

cout << "Select a rubric to see news for:\t1. Games\t2. Films\t3.Music\n>> ";

int answer = getKeyPressed(1, 3);

cout << "\n\n========================================================================================================================================================\n\n";

Rubric\* rubrics[] = { site.games, site.films, site.music };

--answer;

int maxPostsOnPage = 3; //максимальное количество одновременно отображаемых постов

int postsToPrint = rubrics[answer]->howManyInside() > maxPostsOnPage ? maxPostsOnPage : rubrics[answer]->howManyInside();

for (int id = 0; id < postsToPrint; id++)

{

cout << "\t" << rubrics[answer]->getNameAt(id) << " (author - " << rubrics[answer]->getAuthorAt(id) << ")\n";

cout << rubrics[answer]->getContentAt(id) << "\n";

cout << "Published:\t" << rubrics[answer]->getDateAt(id) << "\n";

for (int i = 0; i < (int)rubrics[answer]->getTagsAt(id).size(); i++)

cout << "#" << rubrics[answer]->getTagsAt(id).at(i) << " ";

cout << "\n\n\n\n";

}

cout << "\n\nPress 0 to quit:\n";

cout << ">> ";

answer = getKeyPressed(0, 0);

GamesPage();

}

if (what == "tags")

{

cin.clear();

string key("");

cout << "Enter a tag to search news for:\t";

cin >> key;

cout << "\n\n========================================================================================================================================================\n\n";

Rubric\* rubrics[] = { site.games, site.films, site.music }; //динамический полиморфизм

for (int z = 0; z < 3; z++)

{

for (int i = 0; i < rubrics[z]->howManyInside(); i++)

{

bool match(0);

for (size\_t j = 0; j < rubrics[z]->getTagsAt(i).size(); j++)

{

if (rubrics[z]->getTagsAt(i).at(j) == key)

{

match = 1;

break;

}

}

if (match)

{

cout << "\t" << rubrics[z]->getNameAt(i) << " (author - " << rubrics[z]->getAuthorAt(i) << ")\n";

cout << rubrics[z]->getContentAt(i) << "\n";

cout << "Published:\t" << rubrics[z]->getDateAt(i) << "\n";

for (int x = 0; x < (int)rubrics[z]->getTagsAt(i).size(); x++)

cout << "#" << rubrics[z]->getTagsAt(i).at(x) << " ";

cout << "\n\n\n\n";

}

}

}

cout << "\n\nPress 0 to quit:\n";

cout << ">> ";

int answer = getKeyPressed(0, 0);

GamesPage();

}

if (what == "author")

{

cin.clear();

string key("");

cout << "Author to see news for:\t";

cin >> key;

cout << "\n\n========================================================================================================================================================\n\n";

Rubric\* rubrics[] = { site.games, site.films, site.music }; //динамический полиморфизм

for (int z = 0; z < 3; z++)

for (int i = 0; i < rubrics[z]->howManyInside(); i++)

{

if (rubrics[z]->getAuthorAt(i) == key)

{

cout << "\t" << rubrics[z]->getNameAt(i) << " (author - " << rubrics[z]->getAuthorAt(i) << ")\n";

cout << rubrics[z]->getContentAt(i) << "\n";

cout << "Published:\t" << rubrics[z]->getDateAt(i) << "\n";

for (int x = 0; x < (int)rubrics[z]->getTagsAt(i).size(); x++)

cout << "#" << rubrics[z]->getTagsAt(i).at(x) << " ";

cout << "\n\n\n\n";

}

}

cout << "\n\nPress 0 to quit:\n";

cout << ">> ";

int answer = getKeyPressed(0, 0);

GamesPage();

}

if (what == "period")

{

cin.clear();

int dayF(1), monthF(1), yearF(0);

int dayTo(1), monthTo(1), yearTo(0);

cout << "Searching for period:\n\n";

try

{

cout << "Day from:\t";

cin >> dayF;

if (dayF > 31 || dayF < 1)

throw exception("Bad day format.");

}

catch (exception e)

{

SearchPage("period");

}

try

{

cout << "Month from:\t";

cin >> monthF;

if (monthF > 12 || monthF < 1)

throw exception("Bad month format.");

}

catch (exception e)

{

SearchPage("period");

}

try

{

cout << "Year from:\t";

cin >> yearF;

if (yearF > 2222 || yearF < 1)

throw exception("Bad year format.");

}

catch (exception e)

{

SearchPage("period");

}

try

{

cout << "Day from:\t";

cin >> dayTo;

if (dayTo > 31 || dayTo < 1)

throw exception("Bad day format.");

}

catch (exception e)

{

SearchPage("period");

}

try

{

cout << "Month from:\t";

cin >> monthTo;

if (monthTo > 12 || monthTo < 1)

throw exception("Bad month format.");

}

catch (exception e)

{

SearchPage("period");

}

try

{

cout << "Year from:\t";

cin >> yearTo;

if (yearTo > 2222 || yearTo < 1)

throw exception("Bad year format.");

}

catch (exception e)

{

SearchPage("period");

}

cout << "\n\n========================================================================================================================================================\n\n";

Date from, to;

try

{

from(dayF, monthF, yearF);

to(dayTo, monthTo, yearTo);

if (from > to)

throw exception("Date TO cannot be less than FROM");

}

catch(exception e)

{

SearchPage("period");

}

Rubric\* rubrics[] = { site.games, site.films, site.music }; //динамический полиморфизм

for (int z = 0; z < 3; z++)

for (int i = 0; i < rubrics[z]->howManyInside(); i++)

{

if (rubrics[z]->getDateObjAt(i) >= from && rubrics[z]->getDateObjAt(i) <= to)

{

cout << "\t" << rubrics[z]->getNameAt(i) << " (author - " << rubrics[z]->getAuthorAt(i) << ")\n";

cout << rubrics[z]->getContentAt(i) << "\n";

cout << "Published:\t" << rubrics[z]->getDateAt(i) << "\n";

for (int x = 0; x < (int)rubrics[z]->getTagsAt(i).size(); x++)

cout << "#" << rubrics[z]->getTagsAt(i).at(x) << " ";

cout << "\n\n\n\n";

}

}

cout << "\n\nPress 0 to quit:\n";

cout << ">> ";

int answer = getKeyPressed(0, 0);

GamesPage();

}

GamesPage();

}

int Session::getKeyPressed(int from, int to)

{

int result = -1;

while (result < from || result > to)

{

char keyIsPressed;

keyIsPressed = \_getch();

if (keyIsPressed == '1')

result = 1;

if (keyIsPressed == '2')

result = 2;

if (keyIsPressed == '3')

result = 3;

if (keyIsPressed == '4')

result = 4;

if (keyIsPressed == '5')

result = 5;

if (keyIsPressed == '6')

result = 6;

if (keyIsPressed == '7')

result = 7;

if (keyIsPressed == '8')

result = 8;

if (keyIsPressed == '9')

result = 9;

if (keyIsPressed == '0')

result = 0;

}

return result;

}

void Session::loadData() //подгрузить содержимое сайта из файлов

{

ifstream file("C:\\Users\\Fossa\\source\\repos\\kursach2021oop\\kursach2021oop\\resources\\publications\\content.txt");

bool workingZone = false; //находится ли каретка в нужной зоне для считывания

string name;

string author;

string content;

string rubric;

string tags\_row;

vector <string> tags;

int day = 0, month = 0, year = 0;

if (file.is\_open())

{

cout << "File opened.\n\n";

while (!file.eof())

{

string s;

getline(file, s);

if (s == "")

continue;

else if (s == "/\*")

workingZone = true;

else if (s == "\*/")

break;

else if (s == "\*") //новый пост

{

name = "";

author = "";

content = "";

rubric = "";

tags.clear();

day = 0;

month = 0;

year = 0;

}

else if (workingZone) //считывание одной новости

{

name = s;

getline(file, s);

author = s;

getline(file, s);

content = s;

//тут может быть случай если в тексте присутствуют абзацы - смотрим на пустые строки

//ОБЯЗАТЕЛЬНО после контента в файле должно находиться название одной из 3 рубрик

getline(file, s);

while (true)

{

if (s == "Games" || s == "Films" || s == "Music")

break;

else if (s == "")

content += "\n\n";

else

content += s;

getline(file, s);

}

rubric = s;

getline(file, s);

tags\_row = s;

file >> day >> month >> year;

Date date(day, month, year);

string tag\_buffer = ""; //сюда побуквенно выделяем каждый тег из строки

for (int i = 0; i < (int)tags\_row.size(); i++)

{

if (tags\_row.substr(i, 1) != " ")

tag\_buffer += tags\_row.substr(i, 1);

else

{

tags.push\_back(tag\_buffer);

tag\_buffer = "";

}

}

//тест = смотрим что насобирали

cout << "That's what i've catched:\n\n";

cout << name << endl;

cout << author << endl;

cout << content << endl;

cout << rubric << endl;

for (int i = 0; i < (int)tags.size(); i++)

cout << "#" << tags.at(i) << " ";

cout << endl;

cout << day << endl;

cout << month << endl;

cout << year << endl << endl;

if (!site.publish(name, content, author, rubric, tags, date))

cout << "Something get wrong :(\n\n";

else

cout << "Published.\n\n";

}

}

file.close();

}

else

cout << "Error: cant find content file.\n\n" << endl;

file.open("C:\\Users\\Fossa\\source\\repos\\kursach2021oop\\kursach2021oop\\resources\\accounts\\login\_data.txt");

while (!file.eof())

{

string login(""), password("");

getline(file, login);

getline(file, password);

site.signUp(login, password);

getline(file, login);

}

file.close();

}

const string Session::currentDateTime()

{

time\_t now = time(0);

struct tm tstruct;

char buf[80];

errno\_t err = localtime\_s(&tstruct, &now);

err = asctime\_s(buf, 26, &tstruct);

strftime(buf, sizeof(buf), "%d.%m.%Y %X", &tstruct);

return buf;

}iContainer.h

#pragma once

#include <vector>

template <class Type> //(статический полиморфизм) шаблон класса-контейнера для хранения различных обьектов

class iContainer //(динамический полиморфизм) интерфейс для создания классов-контейнеров

{

protected:

std::vector<Type> data;

public:

virtual int howManyInside() = 0; //возвращает количество хранимых обьектов

};

MessageType.h

enum class MessageType

{

ChooseNumber, UnauthorizedUserMenu, AuthorizedUserMenu, GuestGreeting, PanelGames, PanelFilms,

PanelMusic, URLMenu, URLGames, URLFilms, URLMusic, Greeting, URLMakeNewPost, URLMakeSignIn,

URLMakeSignUp, URLMakeSearch

};

Publication.h

#pragma once

#include <string>

#include <vector>

#include "Date.h"

class Publication

{

private:

class PublicationInfo //композиция

{

public:

std::string name;

std::string author;

std::vector<std::string> tags;

Date date;

PublicationInfo()

{

name = "unknown";

author = "unknown";

}

void updateInfo(std::string \_name, std::string \_author, std::vector<std::string> \_tags, Date \_date)

{

name = \_name;

author = \_author;

tags.swap(\_tags);

date = \_date;

}

~PublicationInfo()

{

tags.clear();

}

};

PublicationInfo info;

std::string content;

public:

Publication(std::string \_name, std::string \_content, std::string \_author, std::vector<std::string> \_tags, Date \_date);

std::string getName();

std::string getAuthor();

std::vector<std::string> getTags();

std::string getDate();

std::string getContent();

Date getDateObj();

};

Publication.cpp

#include "Publication.h"

using namespace std;

Publication::Publication(string \_name, string \_content, string \_author, vector<string> \_tags, Date \_date)

{

info.updateInfo(\_name, \_author, \_tags, \_date);

content = \_content;

}

string Publication::getName()

{

return info.name;

}

string Publication::getAuthor()

{

return info.author;

}

vector<string> Publication::getTags()

{

return info.tags;

}

string Publication::getDate()

{

string result = "";

if (info.date.day < 10)

result += ("0" + to\_string(info.date.day));

else

result += to\_string(info.date.day);

result += ".";

if (info.date.month < 10)

result += ("0" + to\_string(info.date.month));

else

result += to\_string(info.date.month);

result += ("." + to\_string(info.date.year));

return result;

}

string Publication::getContent()

{

return content;

}

Date Publication::getDateObj()

{

return info.date;

}

Rubric.h

#pragma once

#include "Publication.h"

#include "iContainer.h"

class Rubric : public iContainer <Publication>

{

public:

void makePublication(std::string \_name, std::string \_content, std::string \_author, std::vector<std::string> \_tags, Date \_date);

int howManyInside();

std::string getContentAt(int id);

std::string getNameAt(int id);

std::string getAuthorAt(int id);

std::vector <std::string> getTagsAt(int id);

std::string getDateAt(int id);

Date getDateObjAt(int id);

};

Rubric.cpp

#include "Rubric.h"

using namespace std;

void Rubric::makePublication(string \_name, string \_content, string \_author, vector<string> \_tags, Date \_date)

{

Publication newPost(\_name, \_content, \_author, \_tags, \_date);

data.push\_back(newPost);

}

int Rubric::howManyInside()

{

return data.size();

}

string Rubric::getContentAt(int id)

{

string result = "";

if (id >= 0 && id < (int)data.size())

{

result += data.at(id).getContent();

}

return result;

}

string Rubric::getNameAt(int id)

{

string result = "";

if (id >= 0 && id < (int)data.size())

{

result += data.at(id).getName();

}

return result;

}

string Rubric::getAuthorAt(int id)

{

string result = "";

if (id >= 0 && id < (int)data.size())

{

result += data.at(id).getAuthor();

}

return result;

}

vector <string> Rubric::getTagsAt(int id)

{

vector <string> result;

if (id >= 0 && id < (int)data.size())

{

result = data.at(id).getTags();

}

return result;

}

string Rubric::getDateAt(int id)

{

string result = "";

if (id >= 0 && id < (int)data.size())

{

result += data.at(id).getDate();

}

return result;

}

Date Rubric::getDateObjAt(int id)

{

return data.at(id).getDateObj();

}