КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра Автоматизованих Систем Обробки Інформації і Управління

(АСОІУ)

**КУРСОВА РОБОТА**

з дисципліни «Програмування-2. Алгоритми та структури даних»

на тему: «Театральна каса. Розповсюдження квитків»

студента 1 курсу групи ІС-02

спеціальності «Інформаційні системи та технології»

Винника Віталія Валерійовича

Керівник: Вітковська І. І.

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

Дисципліна: Програмування-2. Алгоритми та структури даних

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

Курс I Група ІС-02 Семестр II

**ЗАВДАННЯ**

**на курсову роботу студента**

Винника Віталія Валерійовича

1. Тема проекту (роботи): Театральна каса. Розповсюдження квитків.
2. Строк здачі студентом закінченого проекту (роботи): 20.05.2021.
3. Вихідні дані до проекту (роботи): Тема згідно з варіантом 3, мова програмування C#, технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці): архітектура програми, розробка алгоритмів необхідних для коректної роботи програми.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): Діаграма класів, Use Case діаграма.
6. Дата видачі завдання: 25.03.2021.

**Календарний план**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1 | Отримання теми курсової роботи | 25.03 |  |
| 2 | Підготовка ТЗ | 13.04 |  |
| 3 | Аналіз предметної області | 15.04 |  |
| 4 | Проектування архітектури програмної системи | 18.04 |  |
| 5 | Розробка сценарію роботи програми | 25.04 |  |
| 6 | Узгодження з керівником інтерфейсу користувача |  |  |
| 7 | Розробка програмного забезпечення | 05.05 |  |
| 8 | Узгодження з керівником плану тестування |  |  |
| 9 | Тестування програми | 10.05 |  |
| 10 | Підготовка пояснювальної записки | 17.05 |  |
| 11 | Здача курсової на перевірку | 20.05 |  |
| 12 | Захист курсової роботи | 20.05 |  |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Винник Віталій Валерійович

Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вітковська Ірина Іванівна

**Зміст**

[ВСТУП 3](#_Toc72422546)

[РОЗДІЛ 1. Аналіз предметної області 4](#_Toc72422547)

[1.1. Розробка сценарію роботи програми 4](#_Toc72422548)

[1.2. Функціональні вимоги до програми 5](#_Toc72422549)

[РОЗДІЛ 2. Опис програмного забезпечення 6](#_Toc72422550)

[2.1. Документація 6](#_Toc72422551)

[2.2. Діаграма класів 14](#_Toc72422552)

[РОЗДІЛ 3. Інструкція користувача 15](#_Toc72422553)

[3.1. Робота з програмою 15](#_Toc72422554)

[РОЗДІЛ 4. Тестування ПЗ 23](#_Toc72422555)

[4.1. Алгоритм тестування програми 23](#_Toc72422556)

[ВИСНОВКИ 31](#_Toc72422557)

[Перелік посилань 32](#_Toc72422558)

# ВСТУП

Метою курсової роботи з дисципліни «Структури даних та алгоритми» є покращення уявлення про об’єктно-орієнтований підхід на основі мови С#, а також про всі етапи розробки сучасного програмного забезпечення (ПЗ) та його використання.

Суть завдання варіанту 3 – розробити театральну касу.

Наразі це популярна тема, тому що наше життя неможливо уявити без квитків, тож навички розробки програм такого типу ніколи не будуть зайвими.

За предметною областю необхідно створити програмну реалізацію театральної каси, яка дозволить зручно і легко купувати або бронювати квитки, перед цим дізнавшись, на які вистави вони є.

Тому основним функціоналом має бути:

1. Перегляд вистав, які скоро будуть представлятися.
2. Пошук вистави за різними параметрами.
3. Можливість придбати або забронювати квиток на обрану виставу.

Дана програма має зекономити час глядача, надавши йому можливість придбати квиток на обрану виставу не виходячи з дому.

# РОЗДІЛ 1. Аналіз предметної області

## 1.1. Розробка сценарію роботи програми

Оскільки ПЗ має представляти собою театральну касу, то користувач при запуску програми повинен мати можливість переглянути, які взагалі є вистави і з якої дати розпочинається їх представлення, також йому має бути надана можливість здійснювати їх пошук за різними параметрами. Якщо користувачу сподобалась одна із вистав, він має можливість побачити, з якої дати розпочинається її представлення.

Далі, якщо його все влаштовує, він переходить в касу і вже відповідно, попередньо обравши дату, час та місце, придбати свій бажаний квиток або ж забронювати його та згодом придбати.

Сценарій роботи програми зображений на малюнку:



## 1.2. Функціональні вимоги до програми

Дана програма, при її запуску, повинна надати можливість користувачу ознайомитися з переліком вистав, які будуть представлятися найближчим часом та здійснювати пошук за параметрами серед них або ж відразу перейти в касу та придбати або забронювати квиток на доступні вистави.

Після того як користувач ознайомився з афішею вистав і дізнався, на які вистави і коли є квитки або, можливо, вирішив не скористатися цією можливістю, він переходить в касу і може переглянути чи є якісь вистави сьогодні.

Щоб купити квиток користувач повинен обрати бажану дату, виставу, час її представлення, а також місце в залі, якщо його все влаштовує, то він це підтверджує та вводить наприклад свою електронну адресу, на яку прийде придбаний квиток.

З процедурою бронювання все аналогічно, крім того, що вона можливо потребує ще якісь додаткові дані про , можливо, майбутнього покупця.

Якщо місце обране користувачем вільне, то він придбає або забронює квиток.

Щоб купити заброньований квиток, якщо не вийшов час бронювання, потрібно ввести свої дані, вказані безпосередньо під час самого бронювання.

# РОЗДІЛ 2. Опис програмного забезпечення

### 2.1. Документація

Список класів:

1. Performance – клас, який містить всю необхідну інформацію про виставу:
2. Властивості:

* public Title – отримання назви вистави.
* public Author – отримання автора вистави.
* public Genre – отримання жанру вистави.
* public Year – отримання року вистави.

1. Методи:

* public Performance(string title, string author, string genre, int year) – конструктор з параметрами, що приймає назву вистави, автора, жанр та рік.
* public Equals() – перевизначений метод базового класу object, повертає true, якщо об’єкти однакові між собою, інакше - false.
* public GetHashCode() - перевизначений метод базового класу object, повертає деяке числове значення, яке буде відповідати тільки цьому об’єкту.
* public ToString() - перевизначений метод базового класу object, слугує для отримання рядкового представлення цього об’єкту в даному випадку всю інформацію про виставу.

1. Poster – клас, який містить виставу, дати її показу, а також опис.
2. Поля:

* private description – опис вистави.

1. Властивості:

* public Performance – отримання вистави.
* public Description – отримання опису вистави.
* public Dates – отримання дат показу вистави.

1. Методи:

* public Poster(Performance performance, DateTime startDate) – конструктор з параметрами, що приймає виставу, а також дату початку її показу.
* public AddDate(DateTime date) – метод для додавання дат показу вистави.
* public Equals() – перевизначений метод базового класу object, повертає true, якщо об’єкти однакові між собою, інакше - false.
* public GetHashCode() - перевизначений метод базового класу object, повертає деяке числове значення, яке буде відповідати тільки цьому об’єкту.
* public ToString() - перевизначений метод базового класу object, слугує для отримання рядкового представлення цього об’єкту в даному випадку всю інформацію про виставу.

1. PosterPerformances – клас-контейнер афіш.
2. Поля:

* private posters – список афіш.

1. Властивості:

* public Posters – отримання списку афіш.

1. Методи:

* public PosterPerformances() – конструктор за замовчуванням.
* public AddPoster(Poster poster) – метод для додавання афіші якоїсь вистави, повертає true, якщо вдалось додати.
* public DeletePoster(Poster poster) – метод для видалення афіші якоїсь вистави, повертає true, якщо вдалось видалити.
* public SearchForTitle(string title) – метод, який повертає список афіш вистав із заданою назвою.
* public SearchForAuthor(string author) – метод, який повертає список афіш вистав із заданим автором.
* public SearchForGenre(string genre) – метод, який повертає список афіш вистав із заданим жанром.
* public SearchForYear(int year) – метод, який повертає список афіш вистав із заданим роком.

1. Ticket – клас, який містить основні дані про квиток.
2. Властивості:

* public Title – отримання назви вистави.
* public Status – отримання стану квитка.
* public DateTime – отримання дати і часу.
* public Price – отримання ціни.
* public Row – отримання ряду.
* public Seat – отримання місця.

1. Методи:

* public Ticket(string title, DateTime dateTime, decimal price, int row, int seat) – конструктор з параметрами, що приймає заголовок білету, дату і час, ціну, ряд і місце.

1. Hall – клас, що містить мінімальні необхідну інформацію про зал.
2. Властивості:

* public Name – отримання назви залу.
* public QtyRows – отримання кількості рядів.
* public QtySeatsInRow – отримання кількості місць у кожному ряді.

1. Методи:

* public Hall(string name, int qtyRows, int qtySeatsInRow) – конструктор з параметрами, який приймає назву залу, кількість рядів, кількість місць у рядах.
* public ToString() - перевизначений метод базового класу object, слугує для отримання рядкового представлення цього об’єкту в даному випадку назву залу.

1. ClientInfo – клас, який містить необхідну інформацію про клієнта.
2. Властивості:

* public Name – отримання ім’я.
* public Surname – отримання прізвища.
* public Email – отримання електронної пошти.

1. Методи:

* public ClientInfo(string name, string surname, string email) – конструктор з параметрами, який приймає ім’я, прізвище та електронну пошту користувача.
* public Equals() – перевизначений метод базового класу object, повертає true, якщо об’єкти однакові між собою, інакше - false.
* public GetHashCode() - перевизначений метод базового класу object, повертає деяке числове значення, яке буде відповідати тільки цьому об’єкту.
* public ToString() - перевизначений метод базового класу object, слугує для отримання рядкового представлення цього об’єкту в даному випадку прізвище та ім’я клієнта.

1. Booking – клас, який містить всю інформацію про бронювання.
2. Властивості:

* public ClientInfo – отримання інформації про клієнта.
* public UntilDateTime – отримання кінцевого терміну бронювання.
* public Ticket – отримання квитка.

1. Методи:

* public Booking(ClientInfo clientInfo, Ticket ticket, DateTime untilDateTime) – конструктор з параметрами, що приймає інформацію про клієнта, квиток та термін бронювання.
* public IsActual() – метод, який повертає true, якщо бронь ще досі дійсна, інакше – false.

1. PerformanceTickets – клас, який об’єднує виставу з усією необхідною інформацією для каси.
2. Поля:

* internal bookings – список бронювань.
* internal int availableTickets – кількість доступних квитків.

1. Властивості:

* public Performance – отримання вистави.
* public Tickets – отримання списку квитків.
* public DateTime – отримання дати і часу.
* public Hall – отримання залу.

1. Методи:

* public PerformanceTickets(Performance performance, Hall hall, List<Ticket> tickets, DateTime dateTime) – конструктор з параметрами, що приймає виставу, зал, список білетів та дату з часом.
* public Seats() – метод, який повертає булеву матрицю місць.
* public GetTicketPrice(int row, int seat) – метод, який повертає вартість квитка залежно від місця і ряду.
* internal UpdateTickets() – метод, який оновлює статуси квитків, у яких вийшов термін броні.

1. BoxOffice – головний клас логічної частини ПЗ, клас театральна каса:
2. Поля:

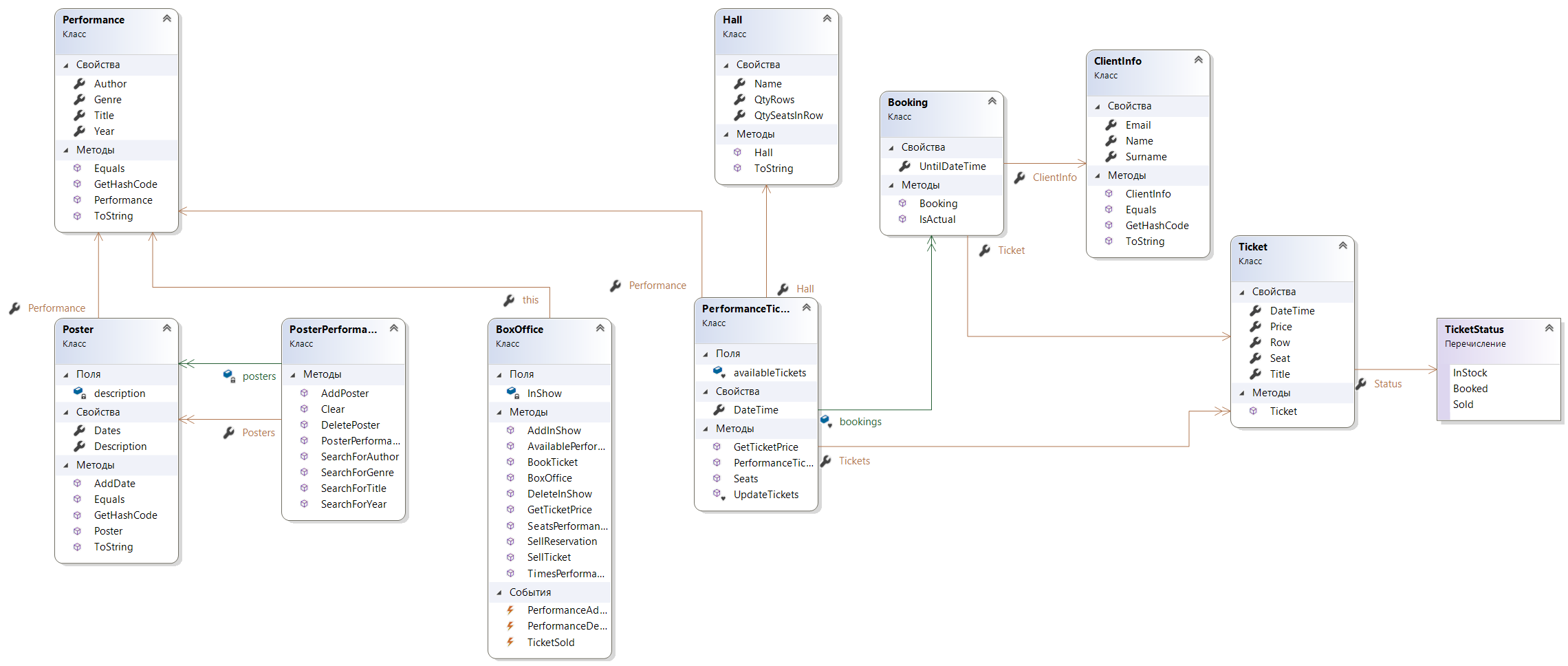
* private InShow – словник, у якому кожній даті відповідає список об’єктів PerformanceTickets.
* public event PerformanceAdded – подія, яка стається при додаванні квитків на виставу.
* public event PerformanceDeleted – подія, яка стається при повному видаленні всіх квитків на виставу.
* public event TicketSold – подія, яка стається при покупці квитка.

1. Методи:

* public BoxOffice() – конструктор за замовчуванням.
* public AddInShow(Performance performance, Hall hall, DateTime dateTime, decimal Maxprice) – метод для додавання квитків у продаж.
* public DeleteInShow(Performance performance, DateTime dateTime) – метод для видалення квитків з продажу.
* public SellTicket(Performance performance, DateTime dateTime, int row, int seat, string communicationMedium) – метод продажу квитка, повертає true, якщо квиток був вільний та успішно придбався, інакше false.
* public BookTicket(Performance performance, DateTime dateTime, int row, int seat, ClientInfo clientInfo,DateTime untilDateTime) – метод бронювання квитка, повертає true, якщо квиток був вільний та успішно забронювався, інакше false.
* public SellReservation(ClientInfo clientInfo) – метод для покупки заброньованого квитка, повертає true, якщо вдалося купити квиток.
* public AvailablePerformances(DateTime date) – метод, що повертає список вистав, на які продаються квитки на певну дату.
* public TimesPerformance(DateTime date, Performance performance) – метод, що повертає список часу представлення даної вистави за якоюсь датою.
* public SeatsPerformance(DateTime dateTime, Performance performance) – метод, який повертає булеву матрицю місць.
* public GetTicketPrice(Performance performance, DateTime dateTime, int row, int seat) – метод для отримання ціни квитка.

1. Індексатор:
2. public Performance this[DateTime date, int index] – повертає виставу за вказаною датою та індексом.

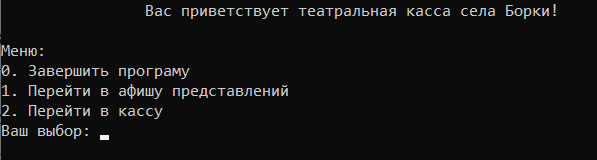
### 2.2. Діаграма класів



# РОЗДІЛ 3. Інструкція користувача

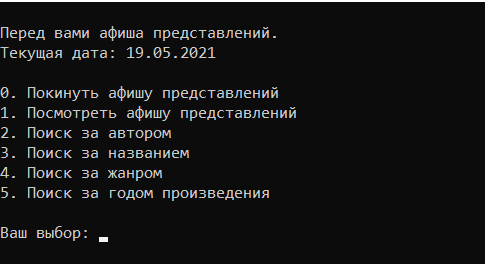
## 3.1. Робота з програмою

Програма починається з привітання та меню взаємодії з користувачем.

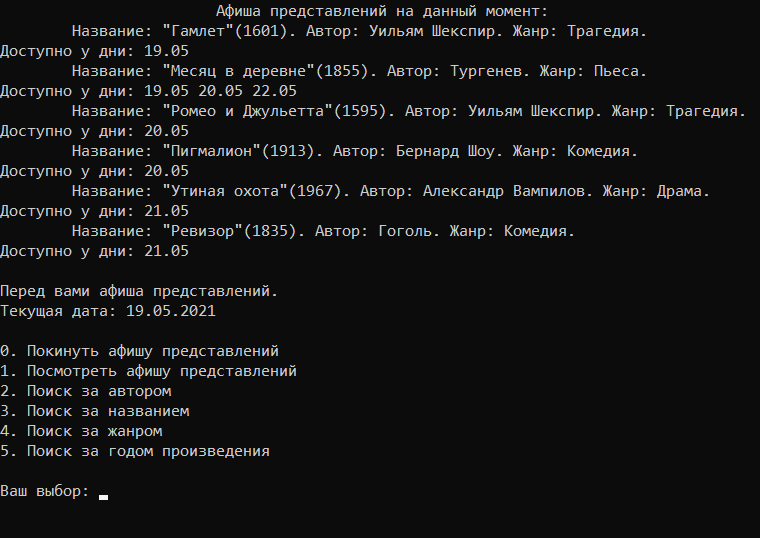


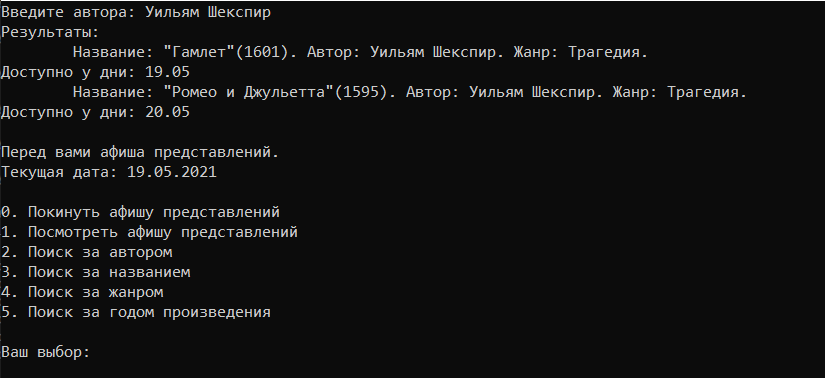
Користувач може обрати будь-який із пунктів меню, але гарною практикою перед вибором бажаної вистави глянути афішу вистав на даний момент, тому перейдемо до неї.

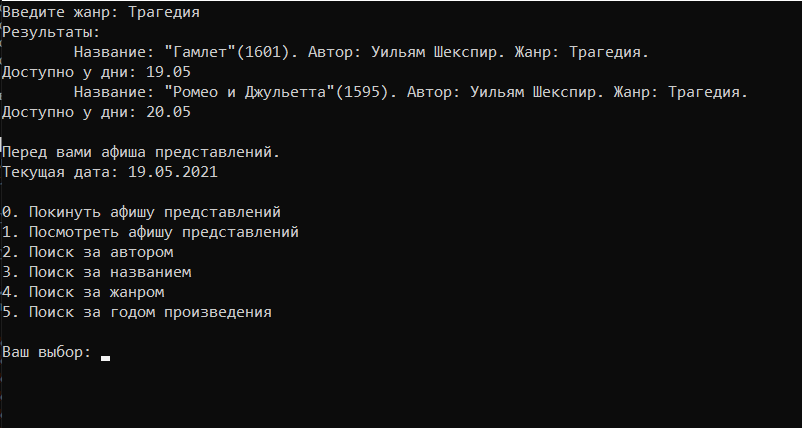
Після переходу висвічується меню взаємодії користувача з афішею.



Користувач може глянути афішу вистав на даний момент та здійснити пошук вистав за певними параметрами:

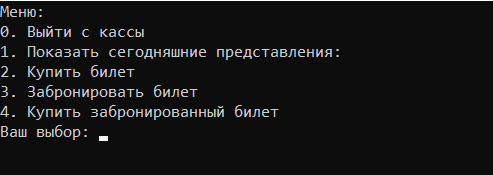




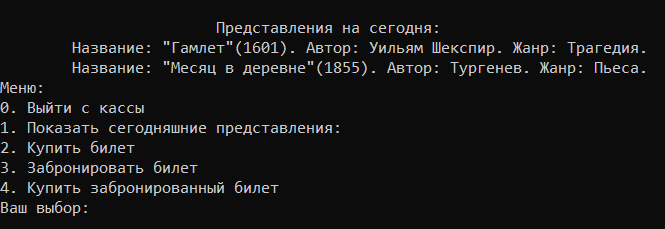


А також за іншими параметрами.

Дізнавшись про вистави, які найближчим часом будуть представлятися, користувач може перейти в касу, перед ним буде висвітлене меню взаємодії з касою:

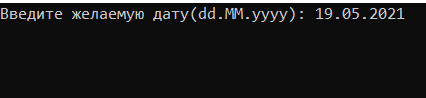


Якщо користувач хоче просто спонтанно піти сьогодні в театр, то він може подивитися чи є якісь вистави, що його зацікавлять, обравши пункт 1.

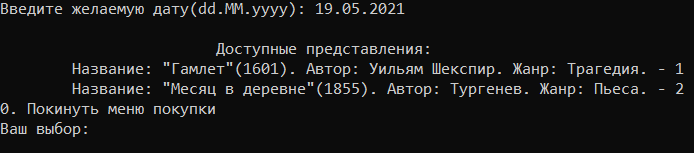


Щоб придбати квиток, користувачу потрібно обрати пункт 2:

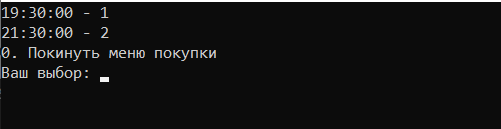
Програма попросить ввести його бажану дату:



Після цього перед ним з’являться вистави, на які є квитки сьогодні:



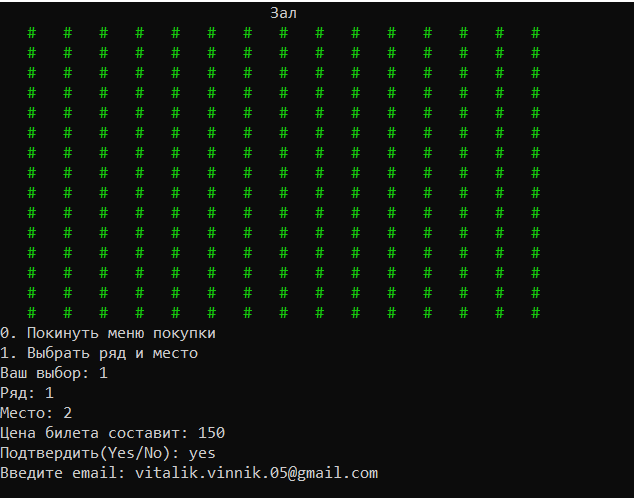
Оберемо наприклад першу виставу



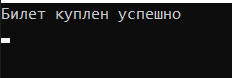
Нам пропонують обрати час, на який запланований її показ, після його вибору, ми потрапляємо в зал:



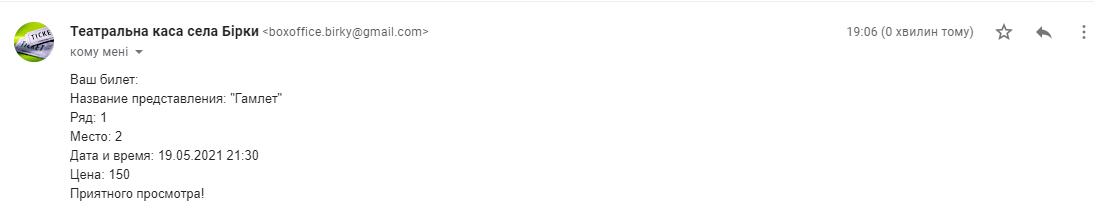
Зробимо покупку квитка, попереднього обравши ряд, місце та ввівши додаткову інформацію:



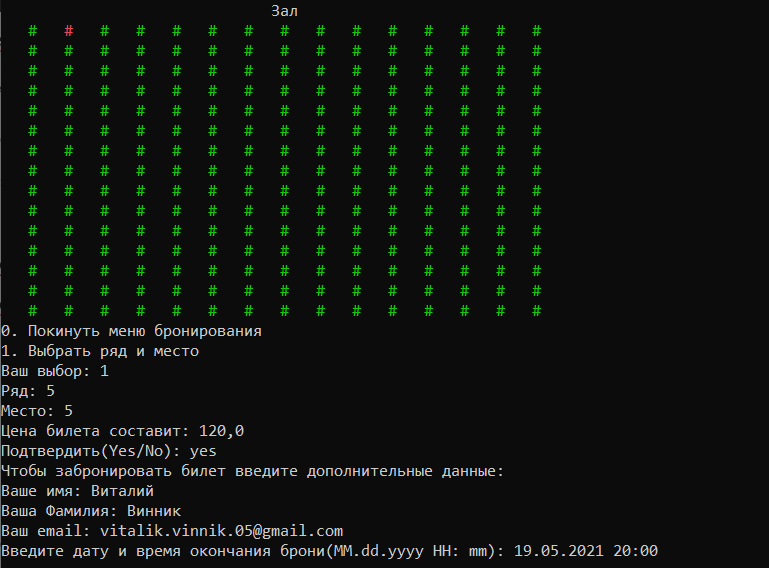
Якщо все пройшло успішно програма повідомить нас про це:

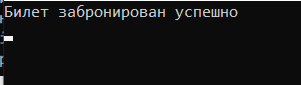


Та пришле білет на вказану електронну пошту:

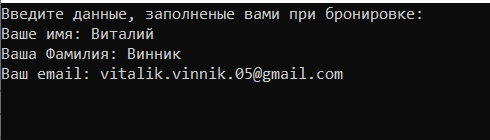


Процес бронювання квитка, майже аналогічний, тільки потрібно ввести додаткову інформацію та термін бронювання:





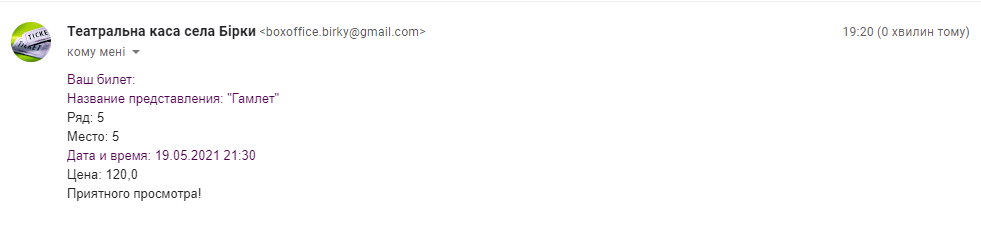
Щоб викупити заброньований білет, потрібно обрати пункт 4 та ввести дані, що заповнювались при бронюванні:



Якщо термін броні ще не вийшов, то білет успішно придбається:



Та прийде на вказану користувачем електронну пошту:

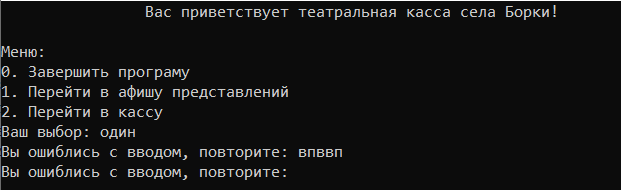


# РОЗДІЛ 4. Тестування ПЗ

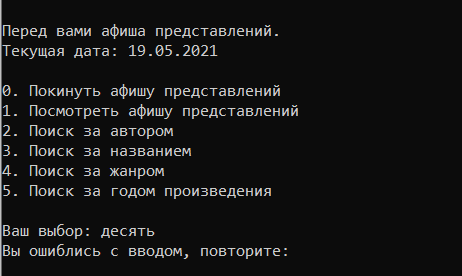
## 4.1. Алгоритм тестування програми

1. Тестування основного меню користувача.
2. Введення некоректних даних при виборі пункту меню.
3. Тестування меню афіші вистав.
4. Введення некоректних даних при виборі пункту меню.
5. Тестування пошуку вистав за параметрами.
6. Тестування меню театральної каси.
7. Введення некоректних даних при виборі пункту меню.
8. Тестування покупки квитка.
9. Введення некоректних даних при покупці квитка.
10. Купування вже проданого або заброньованого квитка.
11. Тестування бронювання квитка.
12. Введення некоректних даних при бронюванні квитка.
13. Бронювання вже проданого або заброньованого квитка.
14. Тестування покупки заброньованого квитка.
15. Введення даних користувача, який не бронював квиток.

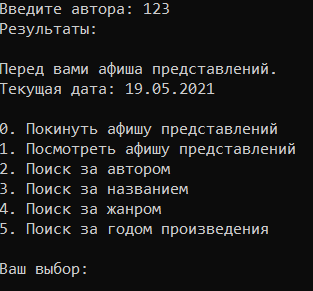
Пункт 1.а.



Пункт 2.а.

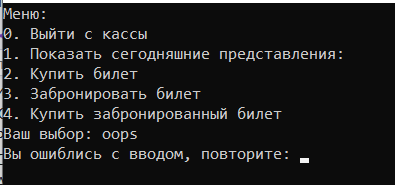


Пункт 2.b.



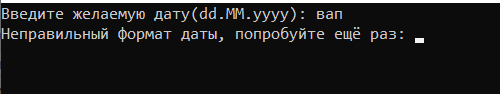
Аналогічно з іншими однотипними методами.

Пункт 3.1.

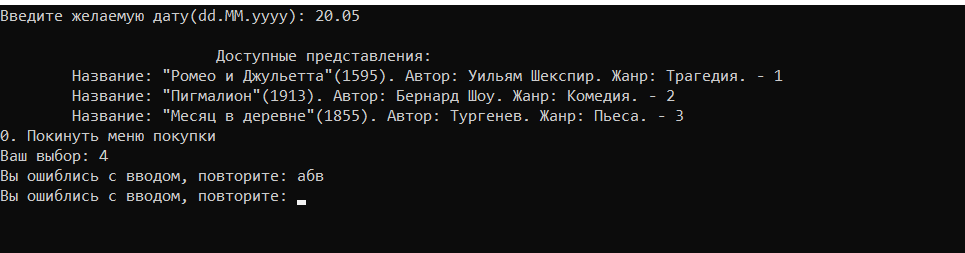


Пункт 3.2.а.

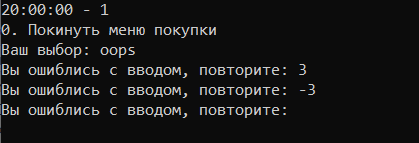
Вибір дати:



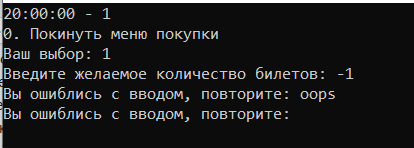
Вибір вистави:



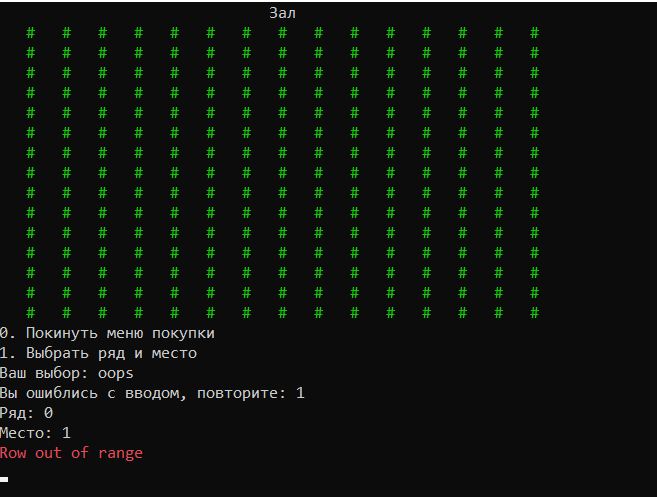
Вибір часу вистави:



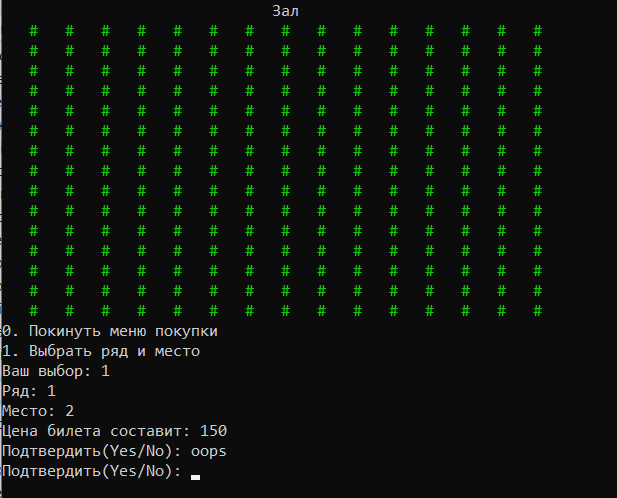
Вибір кількості квитків:



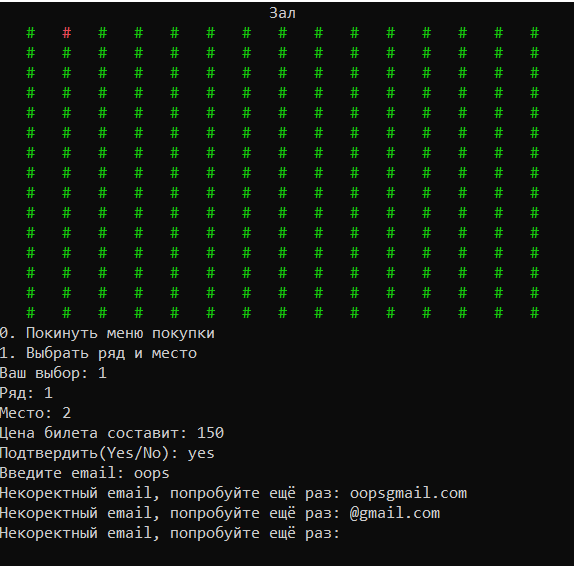
Вибір ряду і місця:



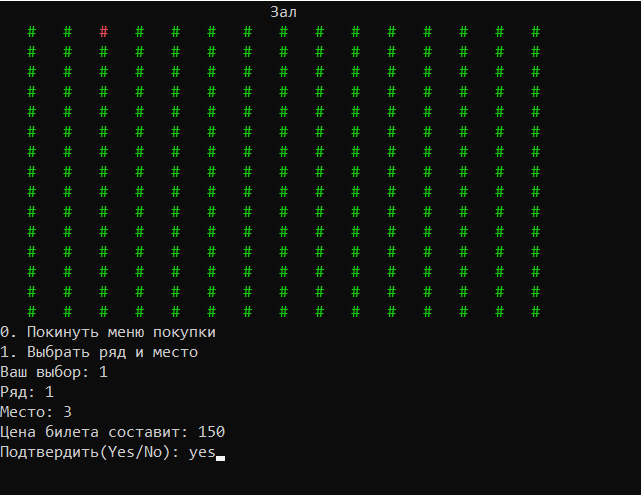
Некоректне підтвердження операції:

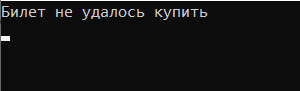


Введення некоректної адреси електронної пошти:



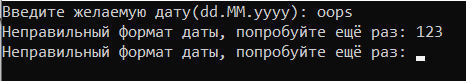
3.2.b.



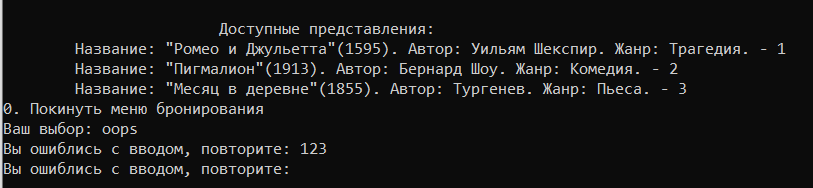


3.3.a.

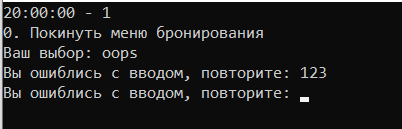
Вибір дати:



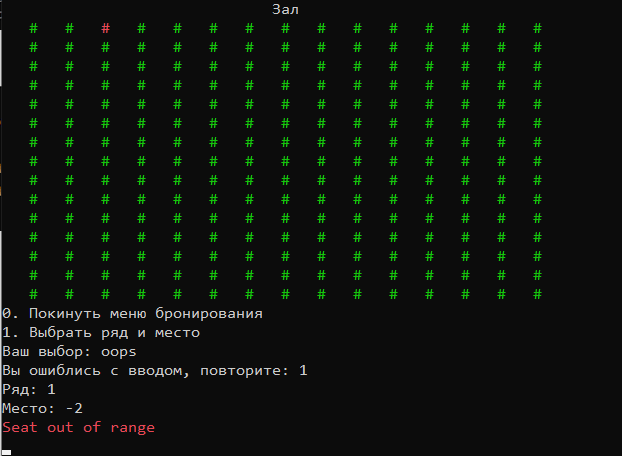
Вибір вистави:



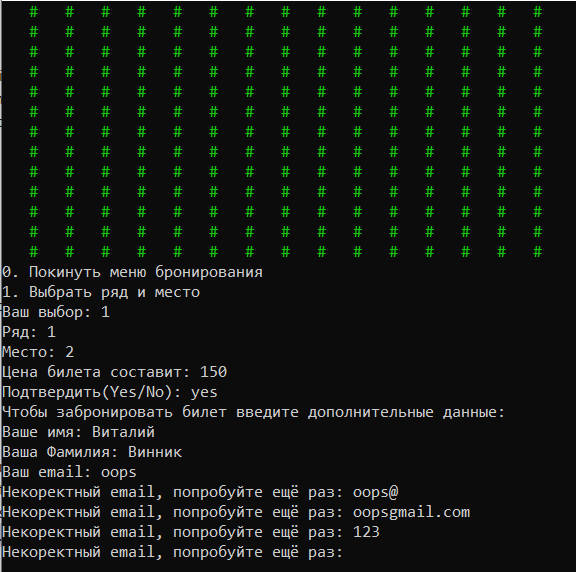
Вибір часу вистави:



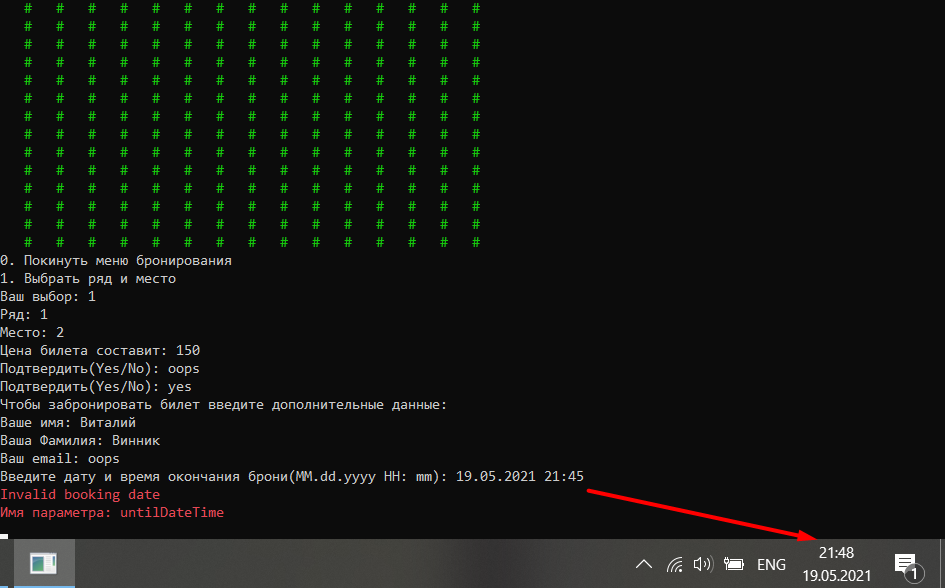
Вибір ряду і місця:



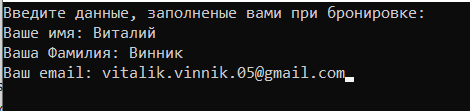
Спроба ввести неправильну електронну адресу:

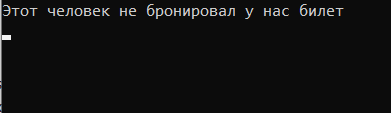


Спроба ввести прострочену дату і час бронювання:



Пункт 3.4.а.





# ВИСНОВКИ

Головною задачею було реалізувати програму, що забезпечить коректну взаємодію користувача з театральною касою.

Під час тестування програми не було виявлено ніяких критичних ситуацій, а всі помилки, які могли виникнути, зазвичай вони виникали через введення некоректних даних користувачем, успішно відловлені.

У процесі виконання були узагальнені знання з дисципліни «Програмування-2. Алгоритми та структури даних»

ПЗ реалізоване в середовищі Visual Studio 2019 Enterprise на мові C#.

# Перелік посилань

1. Програмний код [Електронний ресурс] – [github.com](https://github.com/Vitaliy-126/LabsOOP/tree/master/Coursework).
2. [Електронний ресурс] – [metanit.com](http://metanit.com).
3. Microsoft C# documentation [Електронний ресурс] – [docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/csharp/).
4. Герберт Шилдт - C# 4.0: полное руководство.