**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**



**Курсова робота**

з предмету «Програмування, частина 2 (Об’єктно-орієнтоване програмування)»

на тему:

«**Базові принципи об’єктно-орієнтованого програмування**»

Завдання роботи: «Кінотеатр»

Виконав:

ст.гр. КІ-15

Мармура В.І.

Прийняв:

Козак Н.Б.

**Львів-2019**

**Анотація**

У цій курсовій роботі було виконано написання програмного забезпечення на мові програмування С++ з використанням основних принципів об’єктно-орієнтованого програмування, таких як одинарне та множинне наслідування, абстрагування, інкапсуляція. Програма являє собою спрощену модель кінотеатру та включає у себе роботу з текстовими файлами, а саме: запис, зчитування, пошук, видалення та редагування вибраних даних, а також сортування на екрані за певною характеристикою. Інтерфейс програми написаний англійською мовою. У реалізації програми було використано реальні та вигадані назви предметів, залів, фільмів тощо.

**Зміст**

Завдання на курсову роботу………………………….………………..……………4

Вступ……………………………………………….….………………….…….…….5

1.Огляд та обгрунтування вибору технологій об'єктно орієнтованого програмування…………………………………………………………………….....6

1.1Визначення ООП………....……………………………………………...……….6

1.2Фундаментальні поняття...………………………………………………………7

1.3Об’єктно-орієнтовані особливості С++...………………………………………8

1.4Теорія про діаграми……………………………………………………………..13

2.Проектування……………………………………………………………………..15

2.1.Аналіз об’єктно-орієнтованої моделі об’єкту та алгоритмів функціонування…………………………………………………………………….15

2.1.1.Аналіз контексту проекту……………………………………………………17

2.1.1.1 Структурний аналіз предметної області.………………………………….17

2.1.1.2 Аналіз функціонального наповнення проекту……………………………18

2.1.1.3 Вибір інтерфейсу проекту та пунктів меню, які він реалізовує…………19

2.1.2 Аналіз об’єктно-орієнтованої моделі проекту……………………………...20

2.1.3 Аналіз алгоритмів функціонування проекту……………………………….21

2.2 Програмна реалізація проекту…………………………………………………25

2.2.1 Програмна реалізація об’єктно-орієнтованої моделі проекту…………….25

2.2.2 Програмна реалізація алгоритмів функціонування проекту………………27

2.2.3 Програмна реалізація інтерфейсу проекту та його пунктів меню………..30

3.Відлагодження……………………………………………………………………32

3.1.Висновки..........…………………………………………………………………35

3.2.Список використаної літератури……….......................………………………36

ДОДАТОК А. Лістинг проекту……………………………………………………36

**Завдання на курсову роботу**

У даній курсовій роботі повинно бути реалізовано на мові С++ таку сферу розваг як кінотеатр. У ній повинні бути використані основні принципи об’єктно-орієнтованого програмування.

**Вступ**

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Однією з таких мов є С++. C++ (Сі-плюс-плюс) — мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом (англ. Bjarne Stroustrup) в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові С. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014. У 1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм таких як відеоігри. С++ суттєво вплинула на інші, популярні сьогодні, мови програмування: С# та Java.

***Метою виконання*** цієї курсової роботи є вивчення об’єктно-орієнтованої мови С++ та практичне застосування здобутих моїх знань у цій сфері.

**1.Огляд та обгрунтування вибору технологій об'єктно-орієнтованого програмування.**

**1.1Визначення ООП**

Об'єктно-орієнтоване програмування — це метод програмування, заснований на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування[[6]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-6). Програмісти спочатку пишуть клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів.

На думку Алана Кея, розробника мови Smalltalk, якого вважають одним з «батьків-засновників» ООП, об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:

* Все є об'єктами.
* Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, при якій один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення — це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії.
* Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів.
* Кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів.
* У класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії.
* Класи організовані у єдину деревоподібну структуру з загальним корінням, яка називається ієрархією успадкування. Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Таким чином, програма являє собою набір об'єктів, що мають стан та поведінку. Об'єкти взаємодіють використовуючи повідомлення. Будується ієрархія об'єктів: програма в цілому — це об'єкт, для виконання своїх функцій вона звертається до об'єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об'єктів програми. Звісно, щоб уникнути безкінечної рекурсії у зверненнях, на якомусь етапі об'єкт трансформує запит у повідомлення до стандартних системних об'єктів, що даються мовою та середовищем програмування. Стійкість та керованість системи забезпечуються за рахунок чіткого розподілення відповідальності об'єктів (за кожну дію відповідає певний об'єкт), однозначного означення інтерфейсів міжоб'єктної взаємодії та повної ізольованості внутрішньої структури об'єкта від зовнішнього середовища (інкапсуляції).

**1.2Фундаментальні поняття**

В результаті дослідження Дебори Дж. Армстронг (англ. Deborah J. Armstrong) комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень об'єктно-орієнтованого програмування. До них належить:

Клас

Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості). Наприклад, клас Собака може характеризуватись рисами, притаманними всім собакам, зокрема: порода, колір хутра, здатність гавкати. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Як правило, клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

***Об'єкт***

Окремий екземпляр класу (створюється після запуску програми і ініціалізації полів класу). Клас Собака відповідає всім собакам шляхом опису їхніх спільних рис; об'єкт Сірко є одним окремим собакою, окремим варіантом значень характеристик. Собака має хутро; Сірко має коричнево-біле хутро. Об'єкт Сірко є екземпляром (примірником) класу Собака. Сукупність значень атрибутів окремого об'єкта називається станом. На основі класу Собака можна, також, створити інший об'єкт Дружок, який відрізнятиметься від об'єкта Сірко своїм станом (наприклад кольором хутра). Обидва об'єкта (Сірко і Дружок) є екземплярами класу Собака.

***Метод***

Можливості об'єкта. Оскільки Сірко — Собака, він може гавкати. Тому гавкати() є одним із методів об'єкта Сірко. Він може мати й інші методи, зокрема: місце(), або їсти(). В межах програми, використання методу має впливати лише на один об'єкт; всі Собаки можуть гавкати, але треба щоб гавкав лише один окремий собака.

***Обмін повідомленнями***

«Передача даних від одного процесу іншому, або надсилання викликів методів.»

***Абстрагування***

Спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми. Наприклад Собака Сірко більшу частину часу може розглядатись як Собака, а коли потрібно отримати доступ до інформації специфічної для собак породи коллі — як Коллі і як Тварина (можливо, батьківський клас Собака) при підрахунку тварин Петра.

**1.3Об’єктно-орієнтовані особливості С++**

Сі++ додає до Сі об'єктно-орієнтовані можливості. Він вводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляція, поліморфізм, успадкування.

### Проблеми старого підходу

В мові C основним способом організації даних були структури. Структура складається з набору полів, які ніяк не захищені. Якщо елементи структури мають змінну довжину, їх представляють у вигляді вказівників. Виділення і звільнення пам'яті під ці вказівники робляться вручну. Так, наприклад, одновимірний масив змінної довжини в мові C з перевіркою меж може представлятися таким чином:

**struct** Array {

double\* val;

int len;

};

void FreeArray (**const** **struct** Array\*);

void AllocArray (**const** **struct** Array\*, int len);

double Elem (**const** **struct** Array\*, int i);

void ChangeElem (**const** **struct** Array\*, int i, double x);

Така реалізація небезпечна і неефективна з багатьох причин:

* Необхідно викликати FreeArray і AllocArray. Програміст може забути викликати одну з цих функцій, або викликати її дуже рано/запізно, або двічі, або з вказівником на неправильний масив. Все це приводить до помилок, що важко виявити.
* Функції Elem і ChangeElem повільні.
* Немає ніякого способу перешкодити програмістам створювати і інші функції для роботи із структурою Array. Ці функції можуть робити з полями len і val будь-що.
* Немає ніякого способу перешкодити програмістам безпосередньо міняти поля len і val.
* Присвоєння об'єктів типу struct Array приведе до того, що їхні поля val указуватимуть на одну і ту ж область пам'яті. Немає ніякого способу ні заборонити присвоєння, ні змінити таку поведінку.

Мова Сі++, використовуючи ООП, усуває всі ці проблеми.

**Інкапсуляція**

Основним способом організації інформації в С++ є класи. На відміну від типу, структура (struct) мови С, що складається тільки з полів, клас (class) С++ складається з полів і функцій-членів або методів (англ. member functions). Поля бувають публічними (public), захищеними (protected) і приватними (private). У С++ тип структура аналогічний типу клас, відмінність в тому, що за умовчанням поля і функції-члени у структури публічні, а у класу — приватні.

З публічними полями можна робити зовні класу все, що завгодно. До захищених і приватних полів не можна звертатися ззовні класу, щоб не порушити цілісність даних класу. Спроба такого звернення викличе помилку компіляції. До таких полів можуть звертатися тільки функції-члени класу (а також так звані функції-друзі і функції-члени класів-друзів; про поняття друзів в C++ дивись нижче.) Поза тілом функцій-членів (а також друзів) захищені і власні поля недоступні навіть для читання. Такий захист полів називається інкапсуляциєю.

Використовуючи інкапсуляцію, автор класу може захистити свої дані від некоректного використання. Крім того, вона замислювалася для полегшення сумісної розробки класів. Малося на увазі, що зміна способу зберігання даних, якщо вони оголошені як захищені або приватні, не вимагає відповідних змін в класах, які використовують змінений клас. Наприклад, якщо в старій версії класу дані зберігалися у вигляді лінійного списку, а в новій версії — у вигляді дерева, ті класи, які були написані до зміни формату зберігання даних, переписувати не буде потрібно, якщо дані були приватними або захищеними (у останньому випадку — якщо використовуючі класи не були класами-нащадками), оскільки жоден з них цих класів не міг би безпосередньо звертатися до даних, а тільки через стандартні функції, які в новій версії мають вже коректно працювати з новим форматом даних. Навіть оператор доступу operator [] може бути визначений як така стандартна функція.

Функції-члени, як і поля, можуть бути публічними, захищеними і приватними. Публічні функції може викликати будь-хто, а захищені і власні — тільки функції-члени і друзі.Використовуючи інкапсуляцію, структуру Array з попереднього розділу можна переписати таким чином:

**class** **Array**

{**public**:

void Alloc(int new\_len);

void Free();

**inline** double Elem(int i);

**inline** void ChangeElem(int i, double x);

**protected**:

int len;

double\* val;

};

void Array::Alloc(int new\_len)

{**if** (len>0) Free(); len=new\_len; val=**new** double[new\_len];}

void Array::Free() {**delete** [] val; len=0;}

**inline** double Array::Elem(int i)

{assert(i>=0 && i<len ); **return** val[i];}

**inline** void Array::ChangeElem(int i, double x)

{assert(i>=0 && i<len); val[i]=x;}

Array a;

a.Alloc(10);

a.ChangeElem(3, 2.78);

double b = a.Elem(3);

a.Free();

**Успадкування**

Для створення класів з доданою функціональністю вводять успадкування. Клас-нащадок має поля і функції-члени базового класу, але не має права звертатися до приватних (private) полів і функцій базового класу. У цьому і полягає різниця між приватними і захищеними членами.

Клас-нащадок може додавати свої поля і функції або перевизначати функції базового класу.

За умовчанням, конструктор нащадка без параметрів викликає конструктор базового класу, а потім конструктори доданих елементів. Деструктор працює в зворотному порядку. Інші конструктори доводиться визначати щоразу наново. На щастя, це можна зробити викликом конструктора базового класу.

**class** **ArrayWithAdd** : **public** Array {

ArrayWithAdd(int n) : Array(n) {}

ArrayWithAdd() : Array() {}

ArrayWithAdd(**const** Array& a) : Array(a) {}

void Add(**const** Array& a);

};

Нащадок  — це більш ніж базовий клас, тому він може використовуватися скрізь, де використовується базовий клас, але не навпаки.

Успадкування буває публічним, захищеним і приватним. При публічному успадкуванні, публічні і захищені члени базового класу зберігають свій статус, а до приватних не можуть звертатися навіть функції-члени нащадка. Захищене успадкування відрізняється тим, що при нім публічні члени базового класу є захищеними членами нащадка. При приватному успадкуванні, до жодного члена базового класу навіть функції-члени нащадка права звертатися не мають. Як правило, публічне успадкування зустрічається значно частіше за інші.

Клас може бути нащадком декількох класів. Це називається множинним успадкуванням. Такий клас володіє полями і функціями-членами всіх його предків. Наприклад, клас FlyingCat може бути нащадком класів Cat і FlyingAnimal.

**class** **Cat** {

...

void Purr();

...

};

**class** **FlyingAnimal** {

...

void Fly();

...

};

**class** **FlyingCat** : **public** Cat, **public** FlyingAnimal {

...

PurrAndFly() {Purr(); Fly();}

...

};

**Поліморфізм**

Поліморфізмом в програмуванні називається перевизначення нащадком функцій-членів базового класу, наприклад

**class** **Figure** {

...

void Draw() **const**;

...

};

**class** **Square** : **public** Figure {

...

void Draw() **const**;

...

};

**class** **Circle** : **public** Figure {

...

void Draw() **const**;

...

};

В даному прикладі, яка з функцій буде викликана — Circle::Draw(), Square::Draw() або Figure::Draw(), визначається під час компіляції. Наприклад, якщо написати

Figure\* x = **new** Circle(0,0,5);

x->Draw();

то буде викликане Figure::Draw(), оскільки x — об'єкт класу Figure. Такий поліморфізм називається статичним.

Але в C++ є і динамічний поліморфізм, коли функція, що викликається, визначається під час виконання. Для цього функції-члени повинні бути віртуальними.

**class** **Figure** {

...

**virtual** void Draw() **const**;

...

};

**class** **Square** : **public** Figure {

...

**virtual** void Draw() **const**;

...

};

**class** **Circle** : **public** Figure {

...

**virtual** void Draw() **const**;

...

};

Figure\* figures[10];

figures[0] = **new** Square(1, 2, 10);

figures[1] = **new** Circle(3, 5, 8);

…

**for** (int i = 0; i < 10; i++)

figures[i]->Draw();

У цьому разі для кожного елементу буде викликана Square::Draw() або Circle::Draw() залежно від виду фігури.

Чисто віртуальною функцією називається функція-член, яка не визначена в базовому класі, а тільки в нащадках:

**class** **Figure** {

...

**virtual** void Draw() **const** = 0;

);

**Прототипно-орієнтоване програмування**

Не всі з перелічених вище концепцій присутні в усіх об'єктно-орієнтованих мовах програмування. Зокрема, в прототипно-орієнтованому програмуванні не використовуються класи. Як наслідок, зовсім інша, але аналогічна термінологія використовується для визначення об'єкта та екземпляра в цих мовах.

#### **Агрегація**

[KP-UML-Aggregation-20060420.svg](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:KP-UML-Aggregation-20060420.svg)

Агрегація — проста асоціація між двома класами, яка відображає структурне відношення між рівноправними сутностями, коли обидва класи знаходяться на одному концептуальному рівні, і жоден з них не важливіший за решту. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має вищий ранг (ціле) і складається з декількох менших за рангом (частин). Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і її зображуєть як просту асоціацію з незафарбованим ромбом з боку «цілого».

Графічно агрегація представлена порожнім ромбом на блоці класу, і лінією, яка проведена від цього ромба до класу, що міститься в ньому.

#### **Композиція**

Композиція — більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено.

Графічно представляється як і агрегація, але з зафарбованим ромбиком.

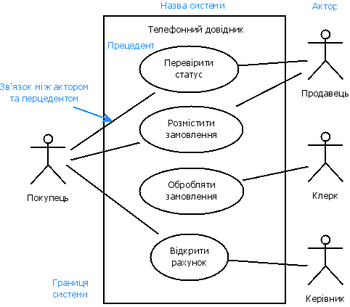
#### **Відмінності між композицією і агрегацією**

Відносини між класом Виш і класами Студент і Факультет злегка відрізняються один від одного, хоча обидва є відносинами агрегування. В виші може бути будь-яка кількість студентів (включаючи нуль), і кожен студент може навчатися в одному або декількох вишах; виш може складатися з одного або декількох факультетів, але кожен факультет належить одному і лише одному вишу. Відношення між класами Виш і Факультет називають композицією, так як при знищенні моделі Виш моделі факультетів, що належать цьому ВНЗ, також повинні бути знищені. Студент і Виш пов'язані агрегацією тому, що Студента не можна видалити при знищенні Вишу.

***1.4Теорія про діаграми***

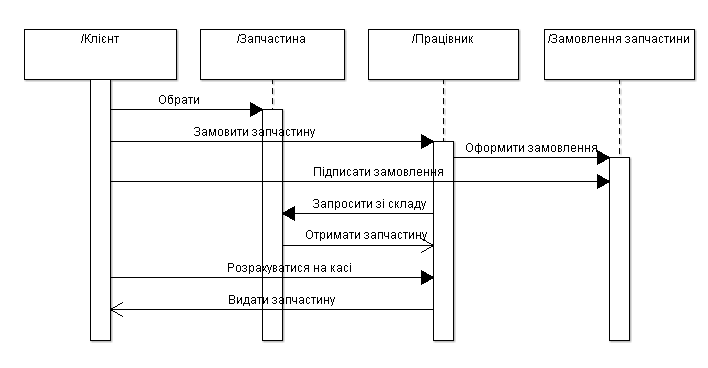
**Діаграма прецедентів**

Діаграма прецедентів — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.



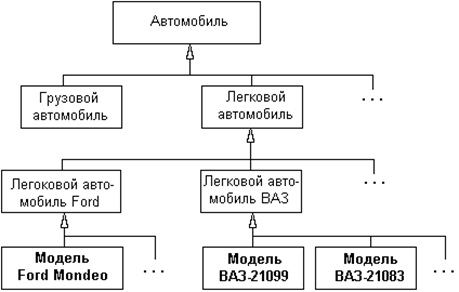
**Діагама послідовності**

**Діаграма послідовності** (англ. *sequence diagram*) — різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.



**Діаграма класів**

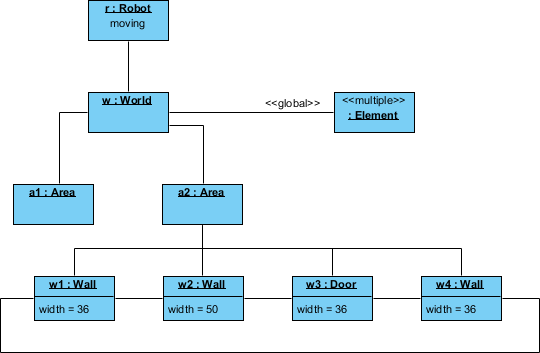
**Діагра́ма кла́сів** (англ. *class diagram*) — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.[[1]](https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%96%D0%B2#cite_note-rumbaught-1) Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.



**Діаграми об’єктів**

Діаграма об'єктів — в UML, діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (англ. collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.



**2.Проектування**

**2.1.Аналіз об’єктно-орієнтованої моделі об’єкту та алгоритмів його функціонування**

**2.1.1.Аналіз контексту проекту**

**2.1.1.1 Структурний аналіз предметної області**

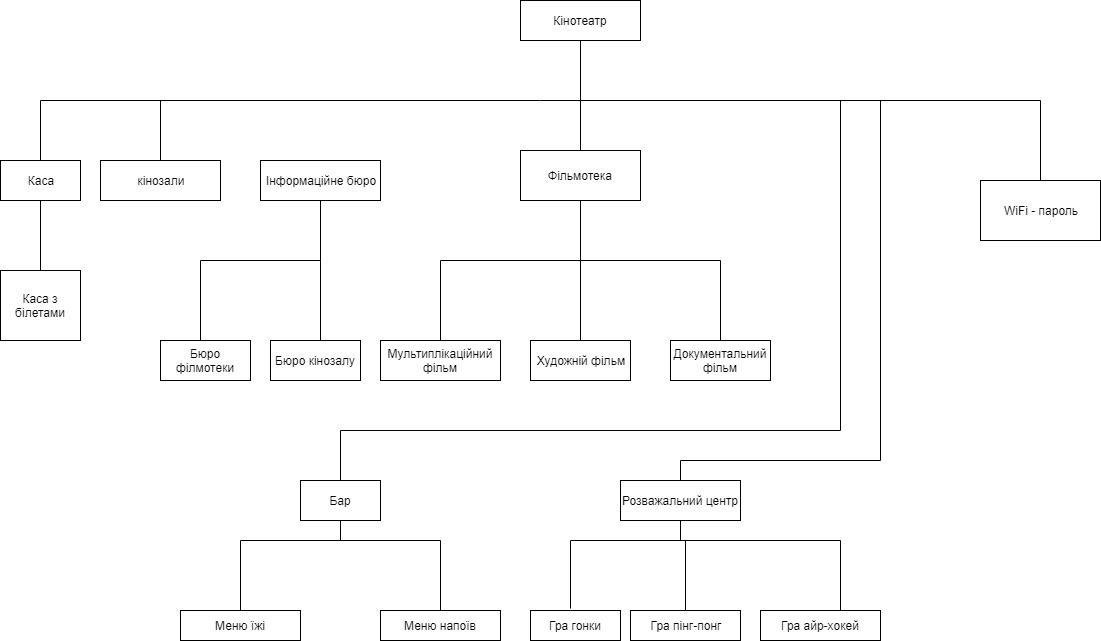


Рис.1 Опис предметної області

**2.1.1.2 Аналіз функціонального наповнення проекту**

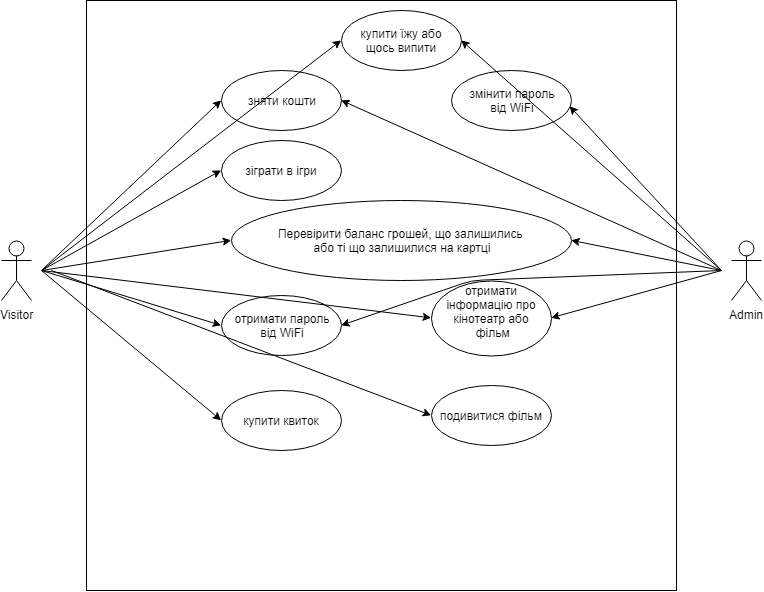


Рис.2 Діаграма прецендентів

**2.1.1.3 Вибір інтерфейсу проекту та пунктів меню, які він реалізовує**

Оскільки консольний тип меню є універсальний, я обрав його для реалізації курсової роботи.

Відкривши программу, нам пропонують зробити вибір Адміна або Відвідувача, а потім вибір який складається із 6 пунктів:

1 - Для перегляду розкладу;

2 - Для отримання інформації про кінотеатр;

3 - Для отримання паролю WiFi;

4 - Для перегляду меню розваг;

5 - Для перегляду меню бару;

6 - Для виходу із програми;

При виборі пункту “1”  у нас з’являється меню з 3 виборів:

1. Повернення до минулого меню;
2. Вибір та купівля квитку;
3. Інформація про фільм;

При виборі пункту “2” ми отримаємо інфорацію про кінотеатр або повернемося у попереднє меню, в залежності від вибору.

При виборі пункту “3” ми отримаємо пароль від WiFi або повернемося у попереднє меню, в залежності від вибору.

При виборі пункту “4”  у нас з’являється меню з 3 виборів:

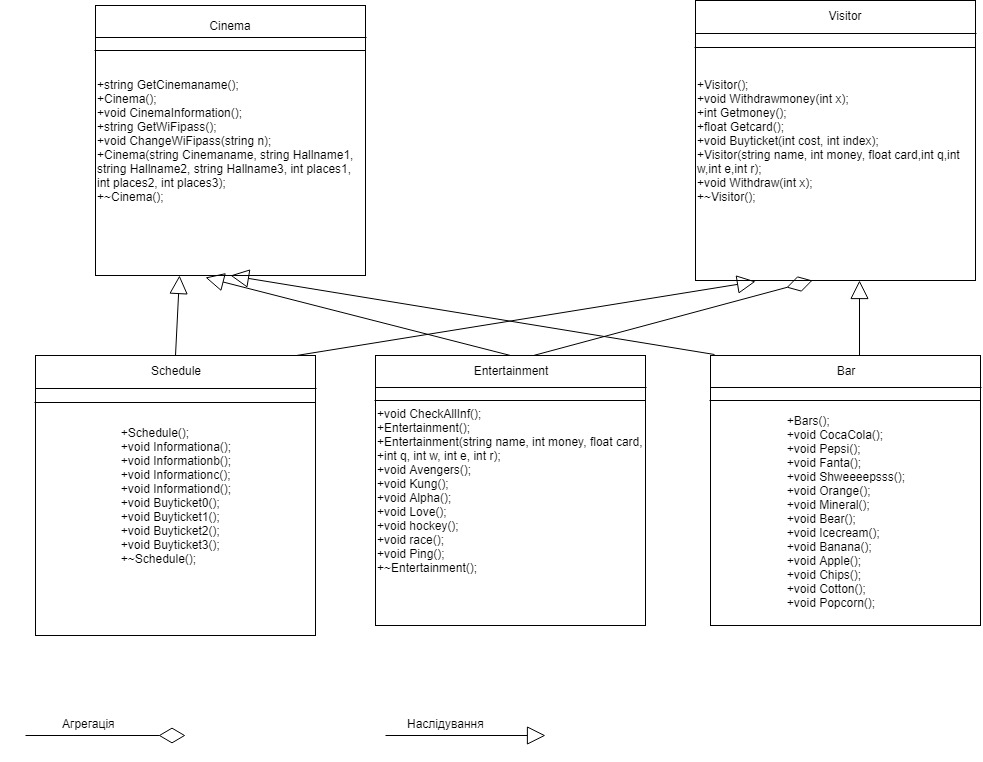
1. Повернення до минулого меню;
2. Грати в айр-хокей;
3. Грати в Need for speed;
4. Грати в пінг-понг;

При виборі пункту “5”  у нас з’являється меню з 3 виборів:

1. Повернення до минулого меню;
2. Зняти гроші з картки;
3. Купити попоїсти;
4. Купити випити;

Шостий пункт надає нам можливість вийти із програми.

**2.1.2 Аналіз об’єктно-орієнтованої моделі проекту**

****Рис.3 Діаграма класів

**2.1.3 Аналіз алгоритмів функціонування проекту**

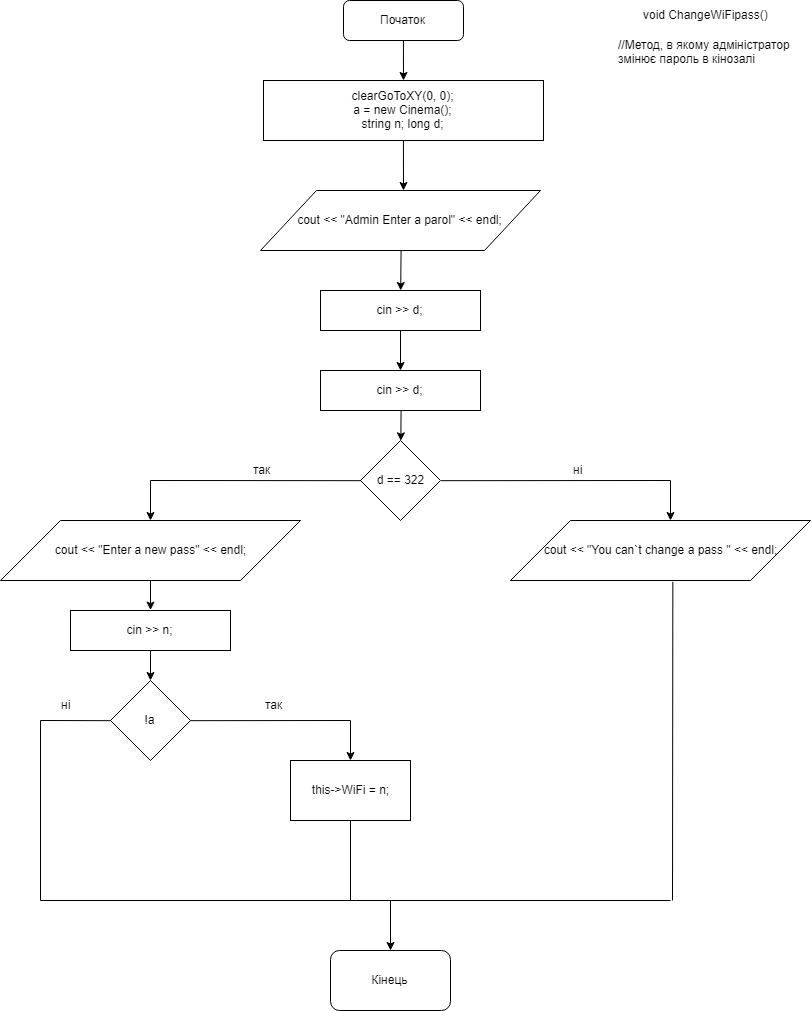
****

Рис.4 Блок-схема алгоритму метода ChangeWiFipass

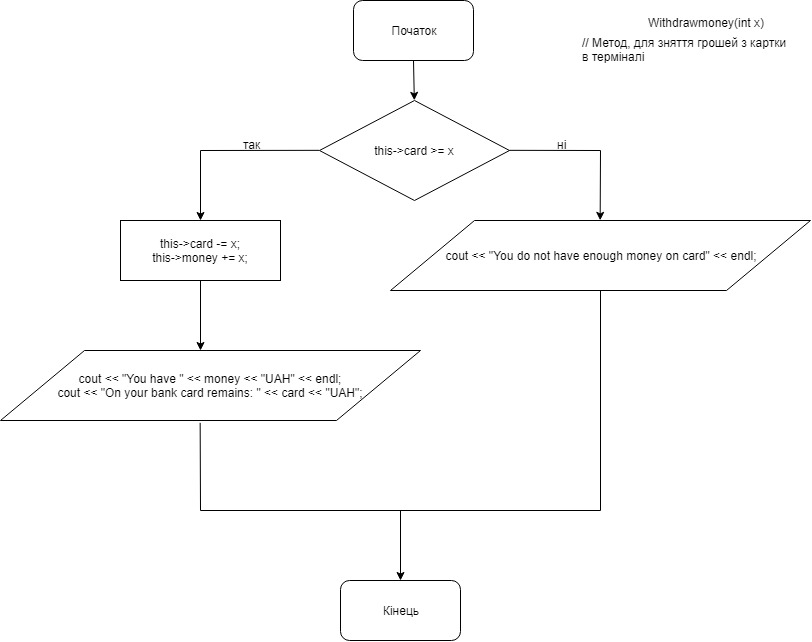


Рис.5 Блок-схема алгоритму метода Withdrawmoney

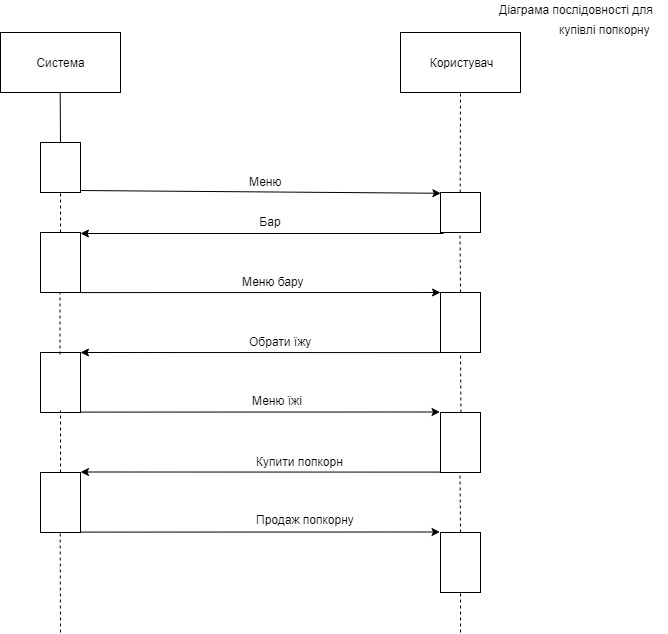


Рис.6 Діаграма послідовності

При запуску програми викликається меню, тоді користувач обирає бар і відправляє повідомлення для виклику бару. Далі система викликає меню бару,а користувач обирає меню їжі. Система виводить на екран меню їжі, а користувач обрирає 1 позицію з меню, в даному випадку попкорн. Далі Система відтворює продаж попкорну користувачеві і далі повертається в попереднє меню.

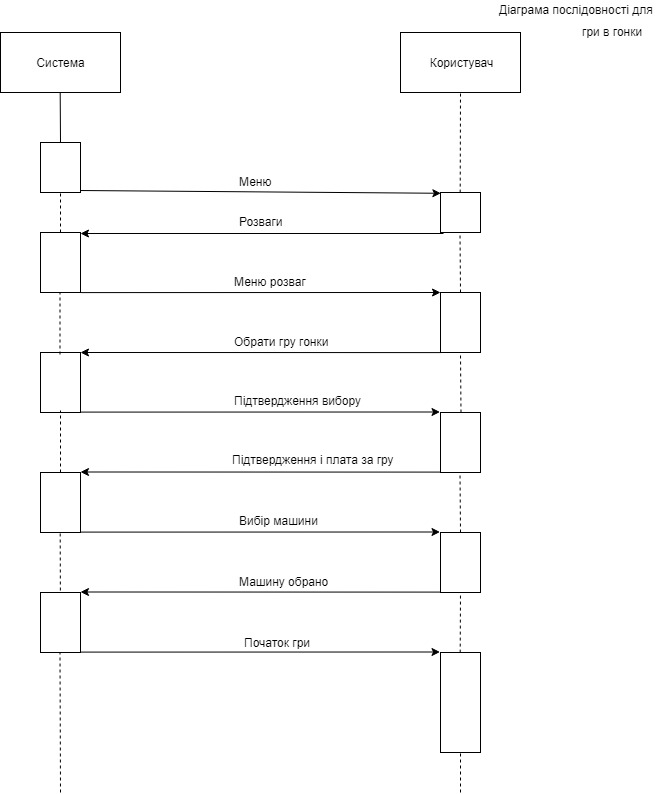


Рис.7 Діаграма послідовності

Спочатку система показує меню. Користувач обрав Розваги. Меню розваг, в ній користувач обирає гру гонки. Система перевіряє правильність вибору, і якщо все правильно починає гру. В грі користувач обирає машину та система вдтворює гру.

**2.2 Програмна реалізація проекту**

**2.2.1 Програмна реалізація об’єктно-орієнтованої моделі проекту**

Клас Entertainment, Bar, Schedule наслідують класи Cinema, Visitor

Приклад наслідування : Entertainment наслідує Visitor

|  |
| --- |
| **//Visitor.h**  #include<iostream>  #include <string>  using namespace std;  class Visitor  {  protected:  string name;  int money;  float card;  public:  int tickets[4];  Visitor();  void Withdrawmoney(int x);  int Getmoney();  float Getcard();  void Buyticket(int cost, int index);  Visitor(string name, int money, float card,int q,int w,int e,int r);  void Withdraw(int x);  ~Visitor();  };  **//Entertainment.h**  #pragma once  #include "Visitor.h"  #include<windows.h>  #include "Cinema.h"  class Entertainment :  public Cinema,public Visitor  {  public:  int time;    Entertainment();  Entertainment(string name, int money, float card, int q, int w, int e, int r);  void Avengers();  void Kung();  void Alpha();  void Love();  void hockey();  void race();  void Ping();  ~Entertainment();  };  **//Entertainment.cpp**  Entertainment::Entertainment(string name, int money, float card, int q, int w, int e, int r) : Visitor(name,money,card,q,w,e,r)  {  cout << "Hello " << this->name << ". You have :" << this->money << "UA. Money on card :" << this->card << endl;  }  **//Bar.h**  #pragma once  #include "Cinema.h"  #include "Visitor.h"  using namespace std;  class Bars :  public Cinema, public Visitor  {  public:  Bars();  void CocaCola();  void Pepsi();  void Fanta();  void Shweeeepsss();  void Orange();  void Mineral();  void Bear();  void Icecream();  void Banana();  void Apple();  void Chips();  void Cotton();  void Popcorn();    ~Bars();  };  **//Cinema.h**  #pragma once  #include<iostream>  #include <string>  #include <vector>  #include<iomanip>  #include<fstream>  using namespace std;  class Cinema  {  protected:  string Hallname1;  string Hallname2;  string Hallname3;  string Cinemaname;  string WiFi;  int places1;  int places2;  int places3;  public:  string GetCinemaname();  Cinema();  void CinemaInformation();  string GetWiFipass();  void ChangeWiFipass(string n);  Cinema(string Cinemaname, string Hallname1, string Hallname2, string Hallname3, int places1, int places2, int places3);  ~Cinema();  };  **//Schedule.h**  #pragma once  #include "Cinema.h"  #include "Visitor.h"  #include<vector>  class Schedule :  public Cinema,public Visitor  {  protected:    public:  Schedule();  void Informationa();  void Informationb();  void Informationc();  void Informationd();  void Buyticket0();  void Buyticket1();  void Buyticket2();  void Buyticket3();  ~Schedule();  }; |

**2.2.2 Програмна реалізація алгоритмів функціонування проекту**

|  |
| --- |
| void Visitor::Buyticket(int cost, int index)  {  if (this->money > cost)  {  this->money -= cost;  tickets[index] += 1;  cout << "You have " << tickets[index] << " tickets on this film" << endl;  }  else  {  cout << "You do not have enough money." << endl;  cout << " You can withdraw your money in our Bar!" << endl;  }  } |

Метод Buyticket Класу Visitor, реалізовує покупку білета. Метод приймає значення ціни квитка, та індекс масиву квитків. Для початку він перевіряє чи ваших грошей вистачає для покупки. Якщо вистачає, то він переходить до наступного рядку, якщо ні , то вказує на це і виводить на екран, що гроші можна зняти у нашому терміналі, що знаходиться в барі. Далі він віднімає від ваших грошей ціну квитка і виводить на екран скільки білетів у вас є на фільм, на який ви щойно купили білет.

|  |
| --- |
| void Cinema::CinemaInformation()  {    string path = "myFile.txt";  fstream fout;  fout.open(path);  if (!fout.is\_open())  {  cout << "File is not open!" << endl;  }  else  {  fout << "It is information about Cinema. Visitor we are happy to welcome you to our cinema!" << endl;  fout << "3-room movie complex:" << Hallname1 << "," << Hallname2 << "," << Hallname3 << ",located on the first floor of the shopping center Podolyany. " << endl;  fout << "Currently, it is the only movie theater in the city, where you can view topical world cinema premieres." << endl;  fout << "We have a lot of guests, and all of them will suffice," << endl;  fout << "because the cinema halls can simultaneously accommodate 400 spectators." << endl;  fout << "All cinema halls are equipped with modern digital equipment. Two of them support 3D technology." << endl;  fout << "High-quality surround sound is provided by the Dolby Digital Surround EX system." << endl;  fout << "It's easy to get into a soft armchair with beverage and popcorn stands" << endl;  fout << "and be ready to plunge into a new, exciting reality. " << endl;  fout << "And on romantics in the last row of each room waiting for wide armchairs LoveSeats." << endl;  fout << "Bars" << endl;  fout << "We want you to have a rest in the cinema and get only pleasant emotions here. Therefore, we have 3 bars in total:" << endl;  fout << "a cocktail bar with armchairs and sofas, where you will find fragrant coffee," << endl;  fout << "ice cream, sweets, beer, alcoholic drinks and cocktails;" << endl;  fout << "Fresh bar, where juice-frees, milk cocktails and strips are prepared;" << endl;  fout << "Concierge Bar with popcorn and classic cinema snacks." << endl;  fout << "We do everything to become your favorite movie theater. Relax, relax and come back." << endl;  }  fout.close();  ifstream fin;  fin.open(path);  string str;  while(!fin.eof())  {  str = " ";  getline(fin, str);  cout << str << endl;  }  fin.close();    } |

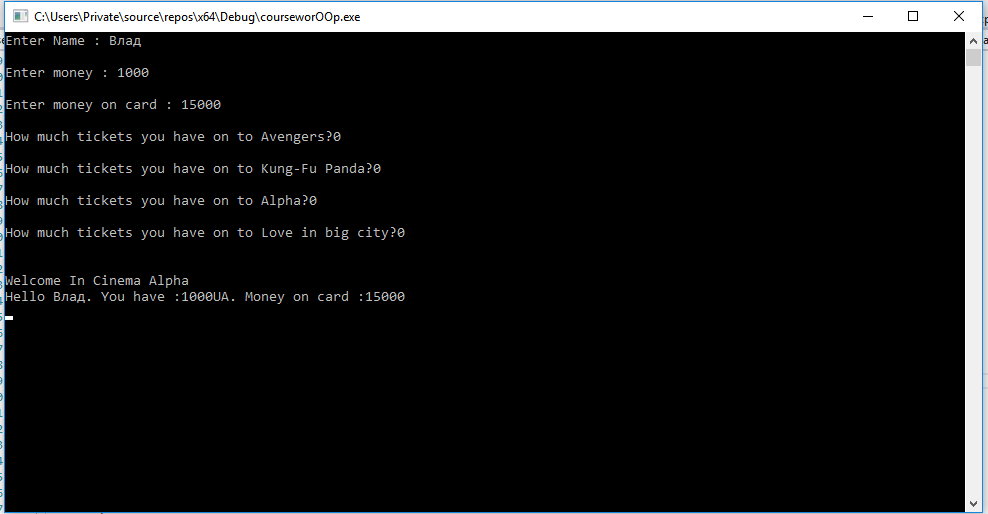
Метод CinemaInformation класу Cinema заносить дані в файл, та потім зчитує з файлу і виводить дані на екран. Спочатку я створюю змінну path типу string та присвоюю їй ім'я файлу. Далі я створив об'єкт fout з яким буду працювати для передання даних у файл. Далі я відкрив файл, перевірив чи він відкрився , і якщо відкрився, то передаю йому всю інформацію про кінотеатр. Потім закриваю fout. Тепер для зчитування з файлу я створюю fin та відкриваю файл. Створюю змінну str типу string. Поки файл не закінчиться, у мене працюватиме цикл з зчитування радка та присвоєння його у змінну str, та вивід рядка на екран. Після того як всі рядки виведуться, то файл закриється.

|  |
| --- |
| void Entertainment::race()  {  if (money >= 50)  {  srand(std::time(NULL));  int n;  cout << "This game cost 50 UA. If you want to play enter 1, else enter another number." << endl;  cin >> n;  if (n == 1)  {  cout << "Choose a car" << endl << "---------------------------------------------------" << endl;  cout << "Ferrari - enter 1" << endl << "Mercedes - enter 2" << endl << "BMW - enter 3" << endl << "Enter anyone number for exit" << endl;  int choose;  cin >> choose; cout << endl;  switch (choose)  {  case 1:  cout << "You choose a Ferrari. Go to race!" << endl;  Sleep(5000);  cout << "Race end. You are ride " << rand() % 10000 + 1000 << "km"<<endl;  break;  case 2:  cout << "You choose a Mercedes. Go to race!" << endl;  Sleep(5000);  cout << "Race end. You are ride " << rand() % 10000 + 1000 << "km" << endl;  break;  case 3:  cout << "You choose a BMW. Go to race!" << endl;  Sleep(5000);  cout << "Race end. You are ride " << rand() % 10000 + 1000 << "km" << endl;  break;  default:  break;  }  }  else  {  return;  }  }  else  {  cout << "You do not have enough money" << endl;  return;  }  getchar();  }  Метод race класу Entertainment реалізовує гру в гонки  void Visitor::Withdrawmoney(int x)  {  if (this->card >= x)  {  this->card -= x;  this->money += x;  cout << "You have " << money << "UAH" << endl;  cout << "On your bank card remains: " << card << "UAH" << endl;  }  else  {  cout << "You do not have enough money on card" << endl;  }  getchar();  }  Метод Withdrawmoney класу Visitor реалізовує зняття грошей в терміналі з картки |

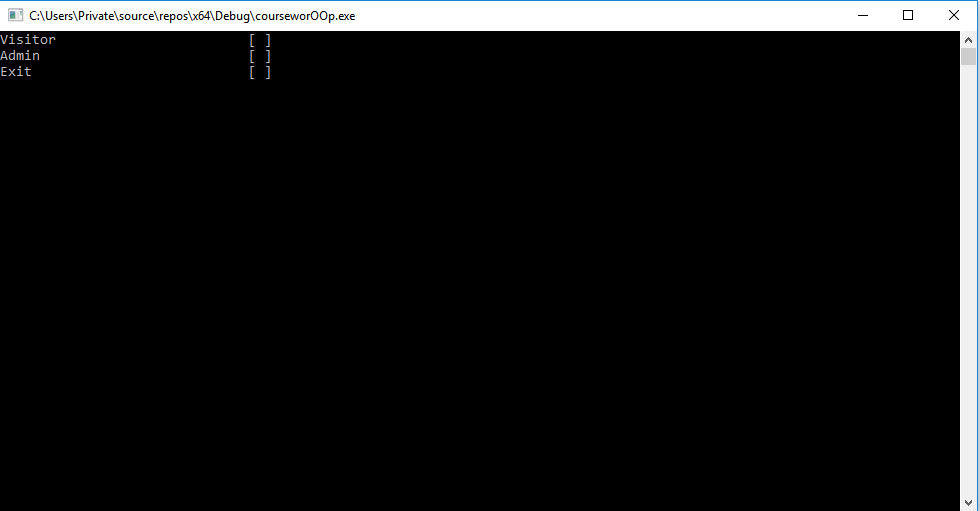
**2.2.3 Програмна реалізація інтерфейсу проекту та його пунктів меню**

|  |
| --- |
| extern struct MenuElement menuA[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuB[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuC[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuC1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuD[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuE[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuF[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuG[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuH[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuK[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuV[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuQ[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuW[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuD1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuE1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuF1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuG1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuH1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuK1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuV1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuA1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuB1[MAX\_MENU\_ITEMS\_COUNT];  struct MenuElement  baseMenu[MAX\_MENU\_ITEMS\_COUNT] = {  { "Visitor", menuQ, (void(\*)())NULL },  { "Admin", menuW, (void(\*)())NULL },  { "Exit", (struct MenuElement \*)NULL, Exit },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuW[MAX\_MENU\_ITEMS\_COUNT] = {  { "Schedule", menuA, (void(\*)())NULL },  { "information about cinema", menuB, (void(\*)())NULL },  { "WiFi pass", menuC, (void(\*)())NULL },  { "FoodDrink", menuE, (void(\*)())NULL },  { "Exit", (struct MenuElement \*)NULL, Exit },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  ,menuQ[MAX\_MENU\_ITEMS\_COUNT] = {  { "Schedule", menuA, (void(\*)())NULL },  { "information about cinema", menuB, (void(\*)())NULL },  { "WiFi pass", menuC1, (void(\*)())NULL },  { "Ententainment", menuD, (void(\*)())NULL },  { "FoodDrink", menuE, (void(\*)())NULL },  { "Exit", (struct MenuElement \*)NULL, Exit },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuA[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Choose and buy ticket", menuH,(void(\*)())NULL },  { "Information about films ", menuK,(void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuB[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Get information about cinema", (struct MenuElement \*)NULL, InfaboutCinema },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuA1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Choose and buy ticket", menuH1,(void(\*)())NULL },  { "Information about films ", menuK1,(void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuB1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Get information about cinema", (struct MenuElement \*)NULL, InfaboutCinema },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuC[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Get WiFi password", (struct MenuElement \*)NULL, WiFipass },  { "Change WiFi password", (struct MenuElement \*)NULL, ChangeWiFipass },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuC1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Get WiFi password", (struct MenuElement \*)NULL, WiFipass },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuD[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Watch film", menuV, (void(\*)())NULL},  { "Play Air-hockey", (struct MenuElement \*)NULL, Hockey },  { "Play Need for speed", (struct MenuElement \*)NULL, Race },  { "Play Ping-pong", (struct MenuElement \*)NULL, Ping },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuE[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Wirhdraw money",(struct MenuElement \*)NULL,WITHDRAW },  { "Check money on card",(struct MenuElement \*)NULL,Checkcard },  { "Check money",(struct MenuElement \*)NULL,Checkmoney },  { "Buy Food", menuG, (void(\*)())NULL },  { "Buy Drink", menuF, (void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuF[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE, (void(\*)())NULL },  { "Coca-Cola (15 UA)", (struct MenuElement \*)NULL, CocaCola },  { "Pepsi(15 UA)", (struct MenuElement \*)NULL, Pepsi },  { "Shweeeepsss(25 UA)", (struct MenuElement \*)NULL, Shweeeepsss },  { "Orange juice(20 UA)", (struct MenuElement \*)NULL, Orange },  { "Mineral water(10 UA)", (struct MenuElement \*)NULL, Mineral },  { "Fanta(15 UA)", (struct MenuElement \*)NULL, Fanta },  { "Bear(25 UA)", (struct MenuElement \*)NULL, Bear },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  },  menuG[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE, (void(\*)())NULL },  { "Potato Chips Lays(40 UA) ", (struct MenuElement \*)NULL, Chips},  { "Popcorn(40 UA)",(struct MenuElement \*)NULL, Popcorn},  { "Cotton candy(50 UA)", (struct MenuElement \*)NULL, Cotton },  { "Apples(5 UA)", (struct MenuElement \*)NULL, Apple },  { "Bananas(10 UA)", (struct MenuElement \*)NULL, Banana },  { "Ice cream(15 UA)", (struct MenuElement \*)NULL, Icecream },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuH[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA, (void(\*)())NULL },  { "Avengers : The End(120 UA)", (struct MenuElement \*)NULL, Buy1 },  { "Kung-fu Panda 4(110 UA)", (struct MenuElement \*)NULL, Buy2 },  { "Alpha(100 UA)", (struct MenuElement \*)NULL, Buy3 },  { "Love in big city(90 UA)", (struct MenuElement \*)NULL, Buy4 },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuV[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuD, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, WAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, WKung },  { "Alpha", (struct MenuElement \*)NULL, WAlpha },  { "Love in big city", (struct MenuElement \*)NULL, WLove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuK[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, IAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, IKung },  { "Alpha", (struct MenuElement \*)NULL, IAlpha },  { "Love in big city", (struct MenuElement \*)NULL, ILove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  },menuD1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Watch film", menuV1, (void(\*)())NULL},  { "Play Air-hockey", (struct MenuElement \*)NULL, Hockey },  { "Play Need for speed", (struct MenuElement \*)NULL, Race },  { "Play Ping-pong", (struct MenuElement \*)NULL, Ping },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuE1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Wirhdraw money",(struct MenuElement \*)NULL,WITHDRAW },  { "Check money on card",(struct MenuElement \*)NULL,Checkcard },  { "Check money",(struct MenuElement \*)NULL,Checkmoney },  { "Buy Food", menuG1, (void(\*)())NULL },  { "Buy Drink", menuF1, (void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuF1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE1, (void(\*)())NULL },  { "Coca-Cola (15 UA)", (struct MenuElement \*)NULL, CocaCola },  { "Pepsi(15 UA)", (struct MenuElement \*)NULL, Pepsi },  { "Shweeeepsss(25 UA)", (struct MenuElement \*)NULL, Shweeeepsss },  { "Orange juice(20 UA)", (struct MenuElement \*)NULL, Orange },  { "Mineral water(10 UA)", (struct MenuElement \*)NULL, Mineral },  { "Fanta(15 UA)", (struct MenuElement \*)NULL, Fanta },  { "Bear(25 UA)", (struct MenuElement \*)NULL, Bear },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  },  menuG1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE1, (void(\*)())NULL },  { "Potato Chips Lays(40 UA) ", (struct MenuElement \*)NULL, Chips},  { "Popcorn(40 UA)",(struct MenuElement \*)NULL, Popcorn},  { "Cotton candy(50 UA)", (struct MenuElement \*)NULL, Cotton },  { "Apples(5 UA)", (struct MenuElement \*)NULL, Apple },  { "Bananas(10 UA)", (struct MenuElement \*)NULL, Banana },  { "Ice cream(15 UA)", (struct MenuElement \*)NULL, Icecream },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuH1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA1, (void(\*)())NULL },  { "Avengers : The End(120 UA)", (struct MenuElement \*)NULL, Buy1 },  { "Kung-fu Panda 4(110 UA)", (struct MenuElement \*)NULL, Buy2 },  { "Alpha(100 UA)", (struct MenuElement \*)NULL, Buy3 },  { "Love in big city(90 UA)", (struct MenuElement \*)NULL, Buy4 },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuV1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuD1, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, WAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, WKung },  { "Alpha", (struct MenuElement \*)NULL, WAlpha },  { "Love in big city", (struct MenuElement \*)NULL, WLove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuK1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA1, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, IAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, IKung },  { "Alpha", (struct MenuElement \*)NULL, IAlpha },  { "Love in big city", (struct MenuElement \*)NULL, ILove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }; |

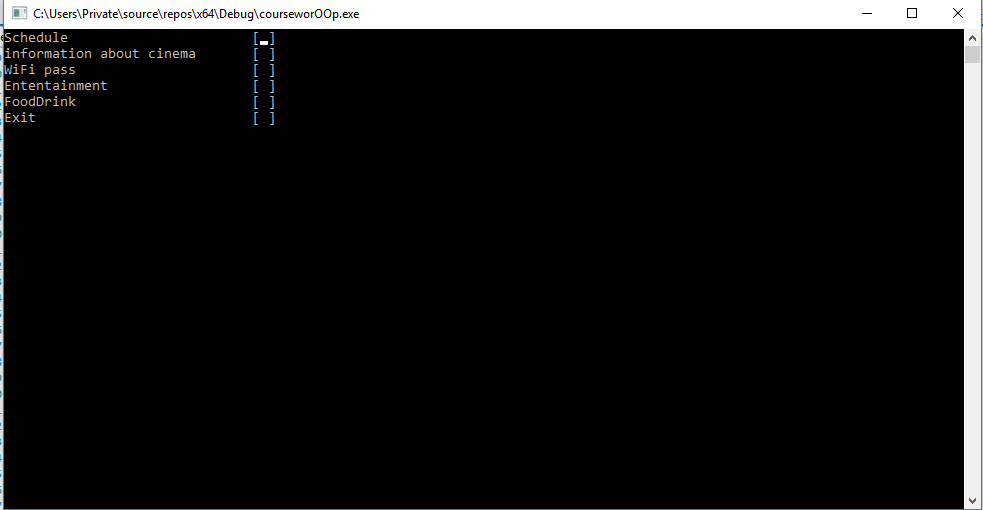
**3.Відлагодження**



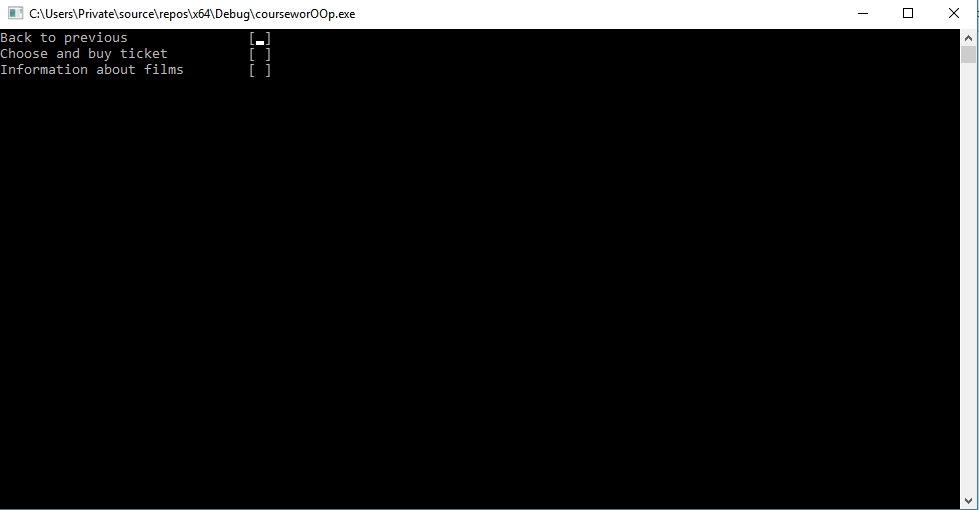
**Рис 3.1 Введення інформації**



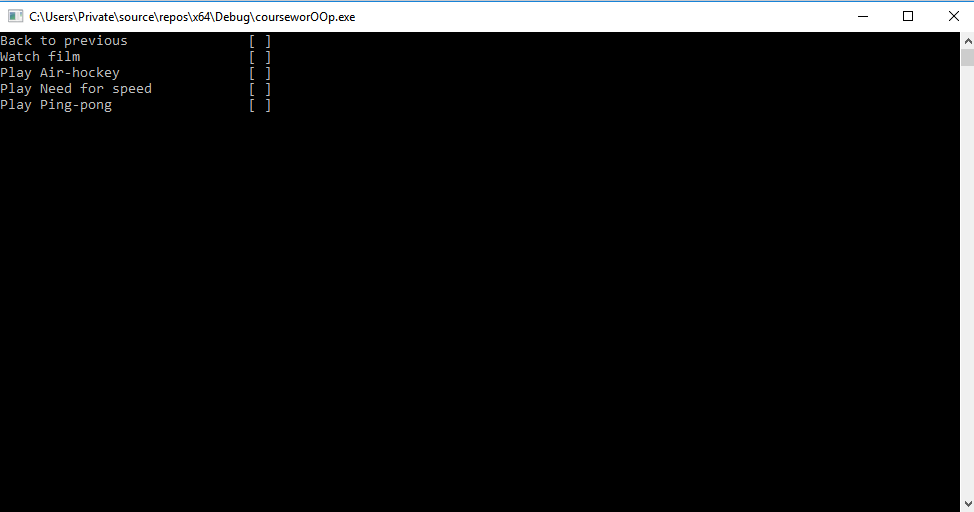
**Рис 3.2 Початкове Меню**



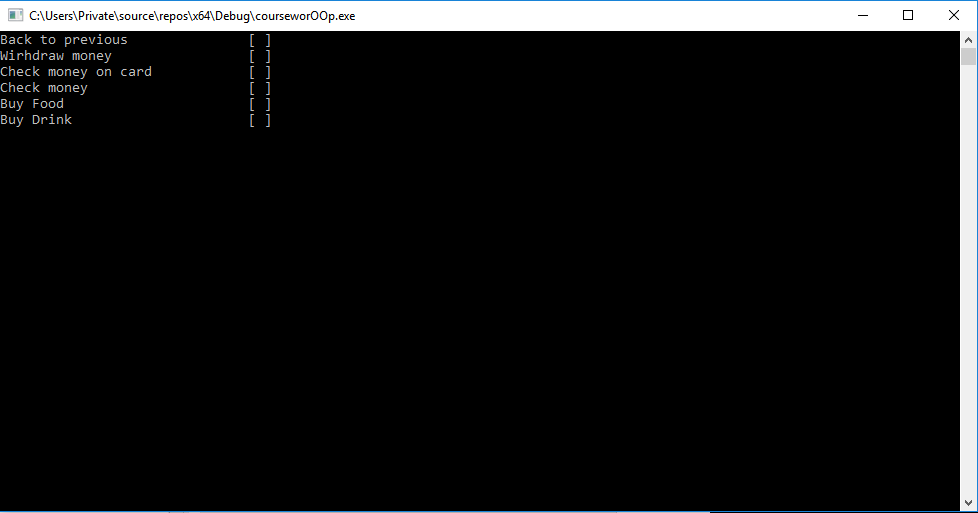
**Рис 3.3 Головне меню**



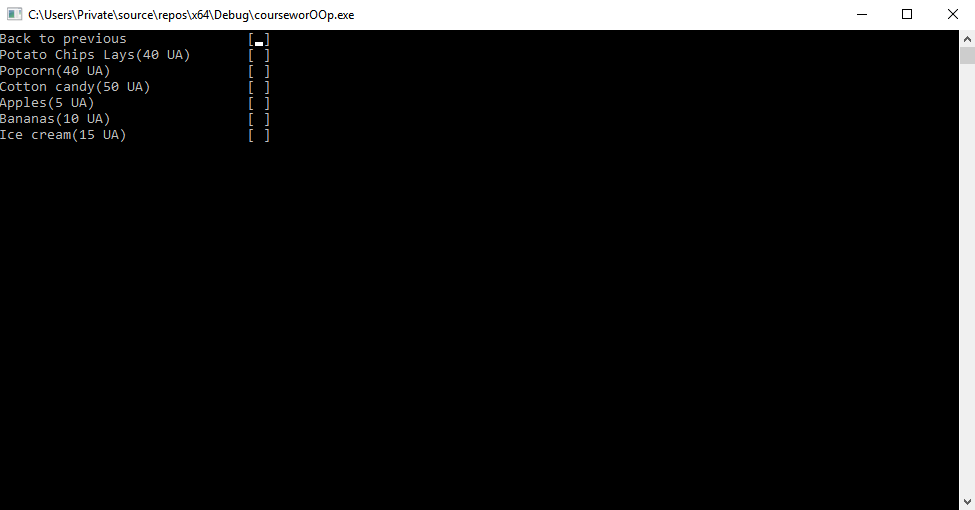
**Рис 3.4 Меню розкладу**



**Рис 3.5 Меню розваг**



**Рис 3.6 Меню бару**



**Рис 3.7 Меню Їжі бару**

**3.1.Висновки**

Використання ООП для вирішення задач такого типу значно полегшує програмісту роботу, робить код більш структурованим та дозволяє легше його читати. Для себе можу зробити важливий висновок, що ООП — гнучкий інструмент який ідеально підходить для написання складних програм які потребують описання деяких об’єктів та їх поведінки.

В ході виконання даної курсової роботи я ознайомився з основними принципами об’єктно-орієнтованого програмування та навчився застосовувати їх на практиці, використовуючи функціональну мову програмування С++.

Засвоїв всі правила використання класів. Провів детальний аналіз предметної області, вивчив проблеми і актуальність створення програмного забезпечення. Під час написання програми зіткнувся з багатьма проблемами, тому отримав знання і навички для їх вирішення. Покращив свої вміння у програмуванні.

Навчився знаходити потрібну інформацію з різних джерел, вміти читати і швидко розбиратись у чужих кодах. Застосовувати вже готові алгоритми у власних цілях і вносити в них певні корективи для коректної роботи програмного забезпечення.

**3.2.Список літератури**

**Використана література :** [1] Кнут Д. «Мистецтво програмування». І.Д. Вільямс, 2007.

[2] Є.В. Пишкін. «Основні концепції та механізми ООП». БХВ-Петербург, 2005.

[3] Греді Буч, Роберт А. Максимчук, Майкл У. Енгл, Боббі Дж. Янг, Джим Коналл, Келлі А. Х'юстон. «Об'єктно-орієнтований аналіз та проектування», І.Д. Вільямс, 2010.

[4] Бьярн Страуструп «Мова програмування С++» (четверте видання), Addison–Wesley, 2013

**ДОДАТОК А. Лістинг проекту**

|  |
| --- |
| **//Visitor.h**  #pragma once  #include<iostream>  #include <string>  using namespace std;  class Visitor  {  protected:  string name;  int money;  float card;  public:  int tickets[4];  Visitor();  void Withdrawmoney(int x);  int Getmoney();  float Getcard();  void Buyticket(int cost, int index);  Visitor(string name, int money, float card,int q,int w,int e,int r);  void Withdraw(int x);  ~Visitor();  };  **//Visitor.cpp**  #include "Visitor.h"  Visitor::Visitor()  {  name = "You";  money = 0;  card = 0.0;  cout << "Welcome In Cinema Alpha" << endl;  }  void Visitor::Withdrawmoney(int x)  {  if (this->card >= x)  {  this->card -= x;  this->money += x;  cout << "You have " << money << "UAH" << endl;  cout << "On your bank card remains: " << card << "UAH" << endl;  }  else  {  cout << "You do not have enough money on card" << endl;  }  getchar();  }  int Visitor::Getmoney()  {  return money;  }  float Visitor::Getcard()  {  return card;  }    void Visitor::Buyticket(int cost, int index)  {  if (this->money > cost)  {  this->money -= cost;  tickets[index] += 1;  cout << "You have " << tickets[index] << " tickets on this film" << endl;  }  else  {  cout << "You do not have enough money." << endl;  cout << " You can withdraw your money in our Bar!" << endl;  }  }  Visitor::Visitor(string name, int money, float card,int q,int w,int e,int r)  {  this->tickets[0] = q;  this->tickets[1] = w;  this->tickets[2] = e;  this->tickets[3] = r;  this->name = name;  this->card = card;  this->money = money;    getchar();  }  void Visitor::Withdraw(int x)  {  if (this->money > x)  { this->money -= x;  cout << "You buy it! Thank you for buy" << endl;  }  else  {  cout << "You do not have enough money." << endl;  cout << " You can withdraw your money in our Bar!" << endl;  }  }    Visitor::~Visitor()  {  }  **//Cinema.h**  #pragma once  #include<iostream>  #include <string>  #include <vector>  #include<iomanip>  #include<fstream>  using namespace std;  class Cinema  {  protected:  string Hallname1;  string Hallname2;  string Hallname3;  string Cinemaname;  string WiFi;  int places1;  int places2;  int places3;  public:  string GetCinemaname();  Cinema();  void CinemaInformation();  string GetWiFipass();  void ChangeWiFipass(string n);  Cinema(string Cinemaname, string Hallname1, string Hallname2, string Hallname3, int places1, int places2, int places3);  ~Cinema();  };  **//Cinema.cpp**  #include "Cinema.h"  using namespace std;  Cinema::Cinema()  {  Cinemaname = "Alpha";  WiFi = "YOULOVEMYCINEMA";  Hallname1 = "Paradise";  Hallname2 = "Space";  Hallname3 = "Philip";  places1=100;  places2=30;  places3=270;  }  void Cinema::CinemaInformation()  {    string path = "myFile.txt";  fstream fout;  fout.open(path);  if (!fout.is\_open())  {  cout << "File is not open!" << endl;  }  else  {  fout << "It is information about Cinema. Visitor we are happy to welcome you to our cinema!" << endl;  fout << "3-room movie complex:" << Hallname1 << "," << Hallname2 << "," << Hallname3 << ",located on the first floor of the shopping center Podolyany. " << endl;  fout << "Currently, it is the only movie theater in the city, where you can view topical world cinema premieres." << endl;  fout << "We have a lot of guests, and all of them will suffice," << endl;  fout << "because the cinema halls can simultaneously accommodate 400 spectators." << endl;  fout << "All cinema halls are equipped with modern digital equipment. Two of them support 3D technology." << endl;  fout << "High-quality surround sound is provided by the Dolby Digital Surround EX system." << endl;  fout << "It's easy to get into a soft armchair with beverage and popcorn stands" << endl;  fout << "and be ready to plunge into a new, exciting reality. " << endl;  fout << "And on romantics in the last row of each room waiting for wide armchairs LoveSeats." << endl;  fout << "Bars" << endl;  fout << "We want you to have a rest in the cinema and get only pleasant emotions here. Therefore, we have 3 bars in total:" << endl;  fout << "a cocktail bar with armchairs and sofas, where you will find fragrant coffee," << endl;  fout << "ice cream, sweets, beer, alcoholic drinks and cocktails;" << endl;  fout << "Fresh bar, where juice-frees, milk cocktails and strips are prepared;" << endl;  fout << "Concierge Bar with popcorn and classic cinema snacks." << endl;  fout << "We do everything to become your favorite movie theater. Relax, relax and come back." << endl;  }  fout.close();  ifstream fin;  fin.open(path);  string str;  while(!fin.eof())  {  str = " ";  getline(fin, str);  cout << str << endl;  }    fin.close();  }  string Cinema::GetWiFipass()  {  return WiFi;  }  void Cinema::ChangeWiFipass(string n)  {    this->WiFi = n;    }  Cinema::Cinema(string Cinemaname, string Hallname1, string Hallname2, string Hallname3, int places1, int places2, int places3)  {  WiFi = "YOULOVEMYCINEMA";  this->Cinemaname = Cinemaname;  this->Hallname1 = Hallname1;  this->Hallname2 = Hallname2;  this->Hallname3 = Hallname3;  this->places1 = places1;  this->places2 = places2;  this->places3 = places3;  }  string Cinema::GetCinemaname()  {  return Cinemaname;  }  Cinema::~Cinema()  {  }  **//Entertainment.cpp**  #include "Entertainment.h"  using namespace std;  Entertainment::Entertainment()  {  }  Entertainment::Entertainment(string name, int money, float card, int q, int w, int e, int r) : Visitor(name,money,card,q,w,e,r)  {  cout << "Hello " << this->name << ". You have :" << this->money << "UA. Money on card :" << this->card << endl;  }  Для скорочення наступних 4 методів, я напишу тільки назву методів тому, що вони мають схожий функціонал, а для цього наведу приклад 1 з них  void Entertainment::Avengers()  {  cout << "BASH!" << endl;    cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << endl;  cout << "BOOOOOOOOM!" << endl;    cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << endl;  cout << "Tanos STOP!!!!!DONT TOUCH GLOVE INFINITY!!!" << endl;    cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << endl;  cout <<"BADABOOOOOM!"<< endl;    cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << endl;  cout << "Avengers, You lose this war" << endl;  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << endl;  cout << "We will be back tanos. Avengers DO NOT GIVE UP!";  cout << endl;  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << ".";  Sleep(100);  cout << endl;  cout << "The end." << endl;  }  void Entertainment::Kung()  void Entertainment::Alpha()  {    void Entertainment::Love()  {    }  void Entertainment::hockey()  {  if (money >= 50)  {  int n;  cout << "This game cost 50 UA. If you want to play enter 1, else enter another number." << endl;  cin >> n;  if (n == 1)  {  money -= 50;  time = 3600;  int coins[2] = { 0,0 };  int amountPlayers = 2;  srand(std::time(NULL));  long long t = 0;  while (t < time) {  for (int i = 0; i < amountPlayers; ++i) {  int result = rand() % 2;  coins[i] += result;  if (result == 1) {  cout << i + 1 << " player scores in the hole(" << coins[i] << ")" << endl;  }  }  t += rand() % (time / 2);  if (t > time) {  t = time;  }  Sleep(500);  cout << "Playing time: " << t << endl;  }  cout << "Scoreboard:" << endl;  for (int i = 0; i < amountPlayers; ++i) {  cout << i + 1 << " player has " << coins[i] << " score(s)" << endl;  }  }  else  {  return;  }  }  else  {  cout << "You do not have enough money" << endl;  return;  }  getchar();  }  void Entertainment::race()  {  if (money >= 50)  {  srand(std::time(NULL));  int n;  cout << "This game cost 50 UA. If you want to play enter 1, else enter another number." << endl;  cin >> n;  if (n == 1)  {  cout << "Choose a car" << endl << "---------------------------------------------------" << endl;  cout << "Ferrari - enter 1" << endl << "Mercedes - enter 2" << endl << "BMW - enter 3" << endl << "Enter anyone number for exit" << endl;  int choose;  cin >> choose; cout << endl;  switch (choose)  {  case 1:  cout << "You choose a Ferrari. Go to race!" << endl;  Sleep(5000);  cout << "Race end. You are ride " << rand() % 10000 + 1000 << "km"<<endl;  break;  case 2:  cout << "You choose a Mercedes. Go to race!" << endl;  Sleep(5000);  cout << "Race end. You are ride " << rand() % 10000 + 1000 << "km" << endl;  break;  case 3:  cout << "You choose a BMW. Go to race!" << endl;  Sleep(5000);  cout << "Race end. You are ride " << rand() % 10000 + 1000 << "km" << endl;  break;  default:  break;  }  }  else  {  return;  }  }  else  {  cout << "You do not have enough money" << endl;  return;  }  getchar();  }  void Entertainment::Ping()  {  if (money >= 50)  {  int n;  cout << "This game cost 50 UA. If you want to play enter 1, else enter another number." << endl;  cin >> n;  if (n == 1)  {  money -= 50;  time = 3600;  int coins[2] = { 0,0 };  int amountPlayers = 2;  srand(std::time(NULL));  long long t = 0;  while (t < time) {  for (int i = 0; i < amountPlayers; ++i) {  int result = rand() % 2;  coins[i] += result;  if (result == 1) {  cout << i + 1 << " player scores (" << coins[i] << ")" << endl;  }  }  t += rand() % (time / 2);  if (t > time) {  t = time;  }  Sleep(500);  cout << "Playing time: " << t << endl;  }  cout << "Scoreboard:" << endl;  for (int i = 0; i < amountPlayers; ++i) {  cout << i + 1 << " player has " << coins[i] << " score(s)" << endl;  }  }  else  {  return;  }  }  else  {  cout << "You do not have enough money" << endl;  return;  }  getchar();  }  Entertainment::~Entertainment()  {  }  **//Entertainment.h**  #pragma once  #include "Visitor.h"  #include<windows.h>  #include "Cinema.h"  class Entertainment :  public Cinema,public Visitor  {  public:  int time;  Visitor vlad;  Entertainment();  Entertainment(string name, int money, float card, int q, int w, int e, int r);  void Avengers();  void Kung();  void Alpha();  void Love();  void hockey();  void race();  void Ping();  ~Entertainment();  };  **//Schedule.h**  #pragma once  #include "Cinema.h"  #include "Visitor.h"  #include<vector>  class Schedule :  public Cinema,public Visitor  {  protected:    public:  Schedule();  void Informationa();  void Informationb();  void Informationc();  void Informationd();  void Buyticket0();  void Buyticket1();  void Buyticket2();  void Buyticket3();  ~Schedule();  };  **//Schedule.cpp**  #include "Schedule.h"  Schedule::Schedule()  {    }  void Schedule::Informationa()  {  ifstream fin;  fin.open("Avengers.txt");  string str;  while (!fin.eof())  {  str = " ";  getline(fin, str);  cout << str << endl;  }  fin.close();  }  void Schedule::Informationb()  {  ifstream fin;  fin.open("Kung.txt");  string str;  while (!fin.eof())  {  str = " ";  getline(fin, str);  cout << str << endl;  }  fin.close();  }  void Schedule::Informationc()  {  ifstream fin;  fin.open("Alpha.txt");  string str;  while (!fin.eof())  {  str = " ";  getline(fin, str);  cout << str << endl;  }  fin.close();  }  void Schedule::Informationd()  {  ifstream fin;  fin.open("Love.txt");  string str;  while (!fin.eof())  {  str = " ";  getline(fin, str);  cout << str << endl;  }  fin.close();  }  Schedule::~Schedule()  {  }  **//Bars.h**  #pragma once  #include "Cinema.h"  #include "Visitor.h"  using namespace std;  class Bars :  public Cinema, public Visitor  {  public:  Bars();  void CocaCola();  void Pepsi();  void Fanta();  void Shweeeepsss();  void Orange();  void Mineral();  void Bear();  void Icecream();  void Banana();  void Apple();  void Chips();  void Cotton();  void Popcorn();    ~Bars();  };  **//Bars.cpp**  #include "Bars.h"  Bars::Bars():Visitor()  {  }  void Bars::CocaCola()  {  cout << "You bought CocaCola" << endl;  }  void Bars::Pepsi()  {  cout << "You bought Pepsi" << endl;  }  void Bars::Fanta()  {  cout << "You bought Fanta" << endl;  }  void Bars::Shweeeepsss()  {  cout << "You bought Shweeeepsss" << endl;  }  void Bars::Orange()  {  cout << "You bought Orange juice" << endl;  }  void Bars::Mineral()  {  cout << "You bought Mineral water" << endl;  }  void Bars::Bear()  {  cout << "You bought Bear" << endl;  }  void Bars::Chips()  {    cout << "You bought Chips" << endl;    }  void Bars::Popcorn()  {    cout << "You bought Popcorn" << endl;    }  void Bars::Cotton()  {    cout << "You bought Cotton" << endl;  }  void Bars::Apple()  {    cout << "You bought Apple" << endl;    }  void Bars::Banana()  {    cout << "You bought Banana" << endl;    }  void Bars::Icecream()  {    cout << "You bought Ice-cream" << endl;    }  Bars::~Bars()  {  }  **//Source.cpp**  #include<iostream>  #include<conio.h>  #include <string>  #include "Visitor.h"  #include "Cinema.h"  #include "Entertainment.h"  #include"Schedule.h"  #include "Bars.h"  #include <Windows.h>  using namespace std;  string n; int m; float c;  Cinema \* a;  Visitor \* v;  Bars \* b;  Schedule \* s;  Entertainment \* e;  #define \_CRT\_SECURE\_NO\_WARNINGS  #define WIN32\_LEAN\_AND\_MEAN  //#include "stdafx.h"  // from HKEY\_USERS\.DEFAULT\Keyboard Layout\Preload  #define ENG\_KEYBOARD\_LAYOUT "00000409"  #define UKR\_KEYBOARD\_LAYOUT "00000422"  #define RUS\_KEYBOARD\_LAYOUT "00000419"  #include "stdio.h"  #include "conio.h"  #include <stdlib.h>  #include <wchar.h>  #include <windows.h>  //#define DISABLE\_DEFAULT\_ACTION  #define MAX\_MENU\_SPACE 1000  #define MAX\_LENGTH\_MENU\_NAME 100  #define MAX\_MENU\_ITEMS\_COUNT 100  void goToXY(int x, int y)  {  COORD p = { x, y };  SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), p);  }  void clearGoToXY(int x, int y) {  COORD topLeft = { x, y };  HANDLE console = GetStdHandle(STD\_OUTPUT\_HANDLE);  CONSOLE\_SCREEN\_BUFFER\_INFO screen;  DWORD written;  GetConsoleScreenBufferInfo(console, &screen);  FillConsoleOutputCharacterA(  console, ' ', screen.dwSize.X \* screen.dwSize.Y, topLeft, &written  );  FillConsoleOutputAttribute(  console, FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_BLUE,  screen.dwSize.X \* screen.dwSize.Y, topLeft, &written  );  SetConsoleCursorPosition(console, topLeft);  }  #define KEY\_UP 72  #define KEY\_DOWN 80  #define KEY\_LEFT 75  #define KEY\_RIGHT 77  #define KEY\_ENTER 13  #define KEY\_BACKSPACE 8  #define KEY\_ESCAPE 27  unsigned int runSpaceOp(int x, int y, int kolommen, char spaceOp[MAX\_MENU\_SPACE][MAX\_MENU\_SPACE], int rijen = MAX\_MENU\_SPACE)  {  int index = 0, indey = 0, keuze, lol = MAX\_MENU\_SPACE;  do {  goToXY(index + x, indey + y);  keuze = 0;  keuze = \_getch();  if (keuze == 0 || keuze == 0xE0 || keuze == 224)  {  keuze = \_getch();  goToXY(index + x, indey + y);  switch (keuze)  {  case KEY\_UP:indey--;  break;  case KEY\_LEFT: index--;  break;  case KEY\_DOWN:indey++;  break;  case KEY\_RIGHT: index++;  break;  }  indey = (indey <= 0) ? 0 : (indey > kolommen) ? kolommen : indey;  index = (index <= 0) ? 0 : (index > rijen) ? rijen : index;  }  if (keuze == 32 || (keuze >= 46 && keuze <= 57) || (keuze <= 64 && keuze >= 126))  {  spaceOp[index][indey] = (char)keuze;  printf("%c", spaceOp[index][indey]);  index++;  index = (index <= 0) ? 0 : (index > rijen) ? rijen : index;  }  if (keuze == KEY\_ENTER)  {  return indey;  }  } while (keuze != KEY\_ESCAPE);  return -1; //  }  void runTextOp(int x, int y, int kolommen, char spaceOp[MAX\_MENU\_SPACE][MAX\_MENU\_SPACE], int rijen = MAX\_MENU\_SPACE)  {  int index = 0, indey = 0, keuze, lol = MAX\_MENU\_SPACE;  do {  goToXY(index + x, indey + y);  keuze = 0;  keuze = \_getch();  if (keuze == 0 || keuze == 0xE0 || keuze == 224)  {  keuze = \_getch();  goToXY(index + x, indey + y);  switch (keuze)  {  case KEY\_UP:indey--;  break;  case KEY\_LEFT: index--;  break;  case KEY\_DOWN:indey++;  break;  case KEY\_RIGHT: index++;  break;  }  indey = (indey <= 0) ? 0 : (indey > kolommen) ? kolommen : indey;  index = (index <= 0) ? 0 : (index > rijen) ? rijen : index;  }  if (keuze == 32 || (keuze >= 46 && keuze <= 57) || (keuze <= 64 && keuze >= 126))  {  spaceOp[index][indey] = (char)keuze;  printf("%c", spaceOp[index][indey]);  index++;  index = (index <= 0) ? 0 : (index > rijen) ? rijen : index;  }  if (keuze == KEY\_BACKSPACE)  {  index = index--;  goToXY(index + x, indey + y);  printf(" ");  index = (index <= 0) ? 0 : (index > rijen) ? rijen : index;  }  if (keuze == KEY\_ENTER)  {  index = 0;  indey++;  indey = (indey >= kolommen) ? kolommen : indey++;  }  } while (keuze != KEY\_ESCAPE);  }  void defaultAction() {  #ifndef DISABLE\_DEFAULT\_ACTION  clearGoToXY(0, 0);  printf("default action");  getchar();  #endif  }  void ChangeWiFipass() {  clearGoToXY(0, 0);  //Cinema a;  a = new Cinema();  string n; long d;  cout << "Admin Enter a parol" << endl;  cin >> d;  if (d == 322)  {  cout << "Enter a new pass" << endl;  cin >> n;  if (!a) {  return;  }  a->ChangeWiFipass(n);  getchar();  }  else  {  cout << "You can`t change a pass " << endl;  }  }  void WiFipass() {  clearGoToXY(0, 0);  //a = new Cinema();  if (!a) {  return;  }  cout << a->GetWiFipass() << endl;  getchar();  //delete a;  }  void Checkcard() {  //Visitor b;  clearGoToXY(0, 0);  cout << v->Getcard() << endl;  getchar();  }  void Checkmoney() {  //Visitor v;  clearGoToXY(0, 0);  cout << v->Getmoney() << endl;  getchar();  }  void Buy1() {  //Visitor b;  clearGoToXY(0, 0);  ////v = new Visitor(n, m, c);  v->Buyticket(120, 0);  cout << "You buy ticket on the Avengers : The End" << endl;  getchar();  }  void Buy2() {  //Visitor b;  clearGoToXY(0, 0);  //v = new Visitor(n, m, c);  cout << "You buy ticket on the Kung-fu Panda 4" << endl;  v->Buyticket(110, 1);      getchar();  }  void Buy3() {  //Visitor b;  clearGoToXY(0, 0);  //v = new Visitor(n, m, c);  cout << "You buy ticket on the Alpha" << endl;  v->Buyticket(100, 2);    getchar();  }  void Buy4() {  //Visitor b;  clearGoToXY(0, 0);  cout << "You buy ticket on the Love in big city" << endl;  //v = new Visitor(n, m, c);    v->Buyticket(90, 3);  getchar();  }  void InfaboutCinema() {  clearGoToXY(0, 0);  Bars a;  a.CinemaInformation();  getchar();  }  void WITHDRAW() {  clearGoToXY(0, 0);  //b = new Bars;  double x;  cout << "How much money you want to withdraw?:" ;  cin >> x;  cout << endl;  v->Withdrawmoney(x);  getchar();  }  void CocaCola()  {  clearGoToXY(0, 0);  b->CocaCola();  v->Withdraw(15);  getchar();  }  void Pepsi()  {  clearGoToXY(0, 0);  b->Pepsi();  v->Withdraw(15);  getchar();  }  void Fanta()  {  clearGoToXY(0, 0);  b->Fanta();  v->Withdraw(15);  getchar();  }  void Shweeeepsss()  {  clearGoToXY(0, 0);  b->Shweeeepsss(); v->Withdraw(25);  getchar();  }  void Orange()  {  clearGoToXY(0, 0);  b->Orange();  v->Withdraw(20);  getchar();  }  void Mineral()  {  clearGoToXY(0, 0);  b->Mineral();  v->Withdraw(10);  getchar();  }  void Bear()  {  clearGoToXY(0, 0);  b->Bear();  v->Withdraw(25);  getchar();  }  void Chips()  {  clearGoToXY(0, 0);  b->Chips();  v->Withdraw(40);  getchar();  }  void Popcorn()  {  clearGoToXY(0, 0);  b->Popcorn();  v->Withdraw(40);  getchar();  }  void Cotton()  {  clearGoToXY(0, 0);  b->Cotton();  v->Withdraw(50);  getchar();  }  void Apple()  {  clearGoToXY(0, 0);  b->Apple();  v->Withdraw(5);  getchar();  }  void Banana()  {  clearGoToXY(0, 0);  b->Banana();  v->Withdraw(10);  getchar();  }  void Icecream()  {  clearGoToXY(0, 0);  b->Icecream();  v->Withdraw(15);  getchar();  }  void IAvengers()  {  clearGoToXY(0, 0);  cout << "Information about Avengers" << endl;  s->Informationa();  getchar();  }  void IAlpha()  {  clearGoToXY(0, 0);  cout << "Information about Alpha" << endl;  s->Informationc();  getchar();  }  void IKung()  {  clearGoToXY(0, 0);  cout << "Information about Kung-Fu Panda" << endl;  s->Informationb();  getchar();  }  void ILove()  {  clearGoToXY(0, 0);  cout << "Information about Love in big city" << endl;  s->Informationd();  getchar();  }  void WLove()  {  clearGoToXY(0, 0);  if (v->tickets[3] > 0)  {  e->Love();  }  else  {  cout << "Buy ticket" << endl;  }  getchar();  }  void WAlpha()  {  clearGoToXY(0, 0);  if (v->tickets[2] > 0)  {  e->Alpha();  }  else  {  cout << "Buy ticket" << endl;  }  getchar();  }  void WKung()  {  clearGoToXY(0, 0);  if (v->tickets[1] > 0)  {  e->Kung();  }  else  {  cout << "Buy ticket" << endl;  }    getchar();  }  void WAvengers()  {  clearGoToXY(0, 0);  if (v->tickets[0] > 0)  {  e->Avengers();  }  else  {  cout << "Buy ticket" << endl;  }    getchar();  }  void Hockey()  {  clearGoToXY(0, 0);  e->hockey();  getchar();  }  void Race()  {  clearGoToXY(0, 0);  e->race();  getchar();  }  void Ping()  {  clearGoToXY(0, 0);  e->Ping();  getchar();  }  void Exit()  {  exit(0);  }  struct MenuElement {  char menuName[1000];  struct MenuElement \* SubMenuSet;  void(\*action)();  };  extern struct MenuElement menuA[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuB[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuC[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuC1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuD[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuE[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuF[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuG[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuH[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuK[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuV[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuQ[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuW[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuD1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuE1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuF1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuG1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuH1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuK1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuV1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuA1[MAX\_MENU\_ITEMS\_COUNT];  extern struct MenuElement menuB1[MAX\_MENU\_ITEMS\_COUNT];  struct MenuElement  baseMenu[MAX\_MENU\_ITEMS\_COUNT] = {  { "Visitor", menuQ, (void(\*)())NULL },  { "Admin", menuW, (void(\*)())NULL },  { "Exit", (struct MenuElement \*)NULL, Exit },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuW[MAX\_MENU\_ITEMS\_COUNT] = {  { "Schedule", menuA, (void(\*)())NULL },  { "information about cinema", menuB, (void(\*)())NULL },  { "WiFi pass", menuC, (void(\*)())NULL },  { "FoodDrink", menuE, (void(\*)())NULL },  { "Exit", (struct MenuElement \*)NULL, Exit },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  ,menuQ[MAX\_MENU\_ITEMS\_COUNT] = {  { "Schedule", menuA, (void(\*)())NULL },  { "information about cinema", menuB, (void(\*)())NULL },  { "WiFi pass", menuC1, (void(\*)())NULL },  { "Ententainment", menuD, (void(\*)())NULL },  { "FoodDrink", menuE, (void(\*)())NULL },  { "Exit", (struct MenuElement \*)NULL, Exit },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuA[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Choose and buy ticket", menuH,(void(\*)())NULL },  { "Information about films ", menuK,(void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuB[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Get information about cinema", (struct MenuElement \*)NULL, InfaboutCinema },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuA1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Choose and buy ticket", menuH1,(void(\*)())NULL },  { "Information about films ", menuK1,(void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuB1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Get information about cinema", (struct MenuElement \*)NULL, InfaboutCinema },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuC[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Get WiFi password", (struct MenuElement \*)NULL, WiFipass },  { "Change WiFi password", (struct MenuElement \*)NULL, ChangeWiFipass },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuC1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Get WiFi password", (struct MenuElement \*)NULL, WiFipass },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuD[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Watch film", menuV, (void(\*)())NULL},  { "Play Air-hockey", (struct MenuElement \*)NULL, Hockey },  { "Play Need for speed", (struct MenuElement \*)NULL, Race },  { "Play Ping-pong", (struct MenuElement \*)NULL, Ping },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuE[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuQ, (void(\*)())NULL },  { "Wirhdraw money",(struct MenuElement \*)NULL,WITHDRAW },  { "Check money on card",(struct MenuElement \*)NULL,Checkcard },  { "Check money",(struct MenuElement \*)NULL,Checkmoney },  { "Buy Food", menuG, (void(\*)())NULL },  { "Buy Drink", menuF, (void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuF[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE, (void(\*)())NULL },  { "Coca-Cola (15 UA)", (struct MenuElement \*)NULL, CocaCola },  { "Pepsi(15 UA)", (struct MenuElement \*)NULL, Pepsi },  { "Shweeeepsss(25 UA)", (struct MenuElement \*)NULL, Shweeeepsss },  { "Orange juice(20 UA)", (struct MenuElement \*)NULL, Orange },  { "Mineral water(10 UA)", (struct MenuElement \*)NULL, Mineral },  { "Fanta(15 UA)", (struct MenuElement \*)NULL, Fanta },  { "Bear(25 UA)", (struct MenuElement \*)NULL, Bear },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  },  menuG[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE, (void(\*)())NULL },  { "Potato Chips Lays(40 UA) ", (struct MenuElement \*)NULL, Chips},  { "Popcorn(40 UA)",(struct MenuElement \*)NULL, Popcorn},  { "Cotton candy(50 UA)", (struct MenuElement \*)NULL, Cotton },  { "Apples(5 UA)", (struct MenuElement \*)NULL, Apple },  { "Bananas(10 UA)", (struct MenuElement \*)NULL, Banana },  { "Ice cream(15 UA)", (struct MenuElement \*)NULL, Icecream },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuH[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA, (void(\*)())NULL },  { "Avengers : The End(120 UA)", (struct MenuElement \*)NULL, Buy1 },  { "Kung-fu Panda 4(110 UA)", (struct MenuElement \*)NULL, Buy2 },  { "Alpha(100 UA)", (struct MenuElement \*)NULL, Buy3 },  { "Love in big city(90 UA)", (struct MenuElement \*)NULL, Buy4 },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuV[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuD, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, WAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, WKung },  { "Alpha", (struct MenuElement \*)NULL, WAlpha },  { "Love in big city", (struct MenuElement \*)NULL, WLove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuK[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, IAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, IKung },  { "Alpha", (struct MenuElement \*)NULL, IAlpha },  { "Love in big city", (struct MenuElement \*)NULL, ILove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  },menuD1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Watch film", menuV1, (void(\*)())NULL},  { "Play Air-hockey", (struct MenuElement \*)NULL, Hockey },  { "Play Need for speed", (struct MenuElement \*)NULL, Race },  { "Play Ping-pong", (struct MenuElement \*)NULL, Ping },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuE1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuW, (void(\*)())NULL },  { "Wirhdraw money",(struct MenuElement \*)NULL,WITHDRAW },  { "Check money on card",(struct MenuElement \*)NULL,Checkcard },  { "Check money",(struct MenuElement \*)NULL,Checkmoney },  { "Buy Food", menuG1, (void(\*)())NULL },  { "Buy Drink", menuF1, (void(\*)())NULL},  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }, menuF1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE1, (void(\*)())NULL },  { "Coca-Cola (15 UA)", (struct MenuElement \*)NULL, CocaCola },  { "Pepsi(15 UA)", (struct MenuElement \*)NULL, Pepsi },  { "Shweeeepsss(25 UA)", (struct MenuElement \*)NULL, Shweeeepsss },  { "Orange juice(20 UA)", (struct MenuElement \*)NULL, Orange },  { "Mineral water(10 UA)", (struct MenuElement \*)NULL, Mineral },  { "Fanta(15 UA)", (struct MenuElement \*)NULL, Fanta },  { "Bear(25 UA)", (struct MenuElement \*)NULL, Bear },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  },  menuG1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuE1, (void(\*)())NULL },  { "Potato Chips Lays(40 UA) ", (struct MenuElement \*)NULL, Chips},  { "Popcorn(40 UA)",(struct MenuElement \*)NULL, Popcorn},  { "Cotton candy(50 UA)", (struct MenuElement \*)NULL, Cotton },  { "Apples(5 UA)", (struct MenuElement \*)NULL, Apple },  { "Bananas(10 UA)", (struct MenuElement \*)NULL, Banana },  { "Ice cream(15 UA)", (struct MenuElement \*)NULL, Icecream },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuH1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA1, (void(\*)())NULL },  { "Avengers : The End(120 UA)", (struct MenuElement \*)NULL, Buy1 },  { "Kung-fu Panda 4(110 UA)", (struct MenuElement \*)NULL, Buy2 },  { "Alpha(100 UA)", (struct MenuElement \*)NULL, Buy3 },  { "Love in big city(90 UA)", (struct MenuElement \*)NULL, Buy4 },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuV1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuD1, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, WAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, WKung },  { "Alpha", (struct MenuElement \*)NULL, WAlpha },  { "Love in big city", (struct MenuElement \*)NULL, WLove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  }  , menuK1[MAX\_MENU\_ITEMS\_COUNT] = {  { "Back to previous", menuA1, (void(\*)())NULL },  { "Avengers : The End", (struct MenuElement \*)NULL, IAvengers },  { "Kung-fu Panda 4", (struct MenuElement \*)NULL, IKung },  { "Alpha", (struct MenuElement \*)NULL, IAlpha },  { "Love in big city", (struct MenuElement \*)NULL, ILove },  { "", (struct MenuElement \*)NULL, (void(\*)())NULL }  };    unsigned int render(struct MenuElement \* menu) {  unsigned int index = 0;  clearGoToXY(0, 0);  if (menu) {  for (; menu[index].menuName[0]; ++index) {  printf(menu[index].menuName);  clearGoToXY(31, index);  printf("[ ]\r\n");  }  }  return --index;  }  int main() {  a = NULL;  string n; int m; float c; int q, w, e1, r;  cout << "Enter Name : ";  cin >> n; cout << endl;  cout << "Enter money : ";  cin >> m; cout << endl;  cout << "Enter money on card : ";  cin >> c; cout << endl;  cout << "How much tickets you have on to Avengers?";  cin >> q; cout << endl;  cout << "How much tickets you have on to Kung-Fu Panda?";  cin >> w; cout << endl;  cout << "How much tickets you have on to Alpha?";  cin >> e1; cout << endl;  cout << "How much tickets you have on to Love in big city?";  cin >> r; cout << endl;  v = new Visitor(n, m, c,q,w,e1,r);  a = new Cinema();  e = new Entertainment(n, m, c, q, w, e1, r);  getchar();  /\*Cinema CCinema("Miracle", "CCinemathebest", "Paradise", "Alpha", "Space", 100, 30, 270);  string a; int b; int c; int choose; int choose1; int choose2; int choose3;  cout << "Hello my friend! What is your name?:";  cin >> a;  cout << "How much money do you have?:";  cin >> b;  cout << "How much money do you have on your card?:";  cin >> c;  \*/  struct MenuElement \* menu = baseMenu;  char spaceOp[MAX\_MENU\_SPACE][MAX\_MENU\_SPACE];  PostMessage(  GetForegroundWindow(),  WM\_INPUTLANGCHANGEREQUEST,  2,  (UINT)LoadKeyboardLayoutA(ENG\_KEYBOARD\_LAYOUT, KLF\_ACTIVATE)  );  while (true) {  unsigned int lastIndex = render(menu);  if ((int)lastIndex == -1) {  \_getch();  return 0;  }  unsigned int selectedIndex = runSpaceOp(32, 0, lastIndex, spaceOp, 0);  if ((int)selectedIndex == -1) {  \_getch();  return 0;  }  if (menu[selectedIndex].action != NULL) {  menu[selectedIndex].action();  }  if (menu[selectedIndex].SubMenuSet != NULL) {  menu = menu[selectedIndex].SubMenuSet;  }  }  \_getch();  return 0;  } |