Dokumentacja do projektu

Cinema booking system

Autor:
Nikodem Szafran

10.01.2024

1. Opis projektu

Projekt pozwala zostać klientem kina oraz menedżerem. Klient może przeglądać harmonogramy, sprawdzić jakie filmy są grane w kinie oraz kiedy. Może także kupować bilety oraz je anulować, a także wyświetlać. Klientów może być kilka, a każdy ma swój profil.

Menedżer kina ma ogromny wpływ na to jaki harmonogram zobaczy klient. Rola ta pozwala dodawać, wyświetlać i usuwać filmy oraz sale kinowe. Dodatkowo decyduje on o godzinach otwarcia kina w danych dniach. Może on także dodawać i usuwać filmy ręcznie, bądź też wygenerować nowy losowy harmonogram wybranego pokoju, dnia lub całego tygodnia. Ostatnimi funkcjonalnościami są: resetowanie całego harmonogramu danego pokoju, dnia lub całego tygodnia, wyświetlanie harmonogramów, a także wyszukiwanie kiedy dany film jest grany.

2. Project description

The project allows users to take on the roles of both a cinema client and a manager. As a client, one can browse schedules, check which movies are being shown in the cinema, and when. They can also purchase and cancel tickets, as well as view their history. There can be multiple customers, each with their own profile.

The cinema manager has significant control over the schedules visible to customers. This role enables the addition, display, and removal of movies and cinema rooms. Additionally, the manager decides the opening hours of the cinema on specific days. They can manually add or remove movies and generate a new random schedule for a selected room, day, or the entire week. The last functionalities include resetting the entire schedule for a specific room, day, or the entire week, displaying schedules, and searching for when a particular movie is being shown.

3. Instrukcja użytkownika

Po uruchomieniu wyświetla nam się menu wyboru roli: klienta kina lub menedżera. Program ma interfejs tekstowy, a operujemy w nim wpisując poprawne dane. W każdym miejscu w programie poza tym i wyjątkowymi przypadkami (np. wpisywaniem stringa np. tytułu filmu) – **jesteśmy w stanie wrócić do poprzedniego punktu programu wpisując '-1'.** W niektórych miejscach inne cyfry np. **'0'**, **'-2'** również mają specjalne zastosowanie, o czym program zawsze powiadomi.

a) Rola klienta

Najpierw zostaniemy poproszeni o wpisanie swojego imienia i nazwiska, które będzie widniało później na bilecie, oraz wieku (przedział [6,99]) – filmy mają restrykcje wiekowe. Gdy to uczynimy po chwili wyświetli nam się menu:

```
*************************MAIN MENU**********************************
Hey uName uSurname! This is main menu. Remember that you can ALWAYS go back by inputing '-1'.

AVAIABLE COMMANDS:
1. Buy ticket for a movie.
2. Show your tickets.
3. Cancel your ticket.
4. Show movies in play.
5. Show when and where movie is playing.
6. Show week/day schedule.
7. Clear console window.

Choose what you want to do by writing number:
```

Menu klienta

Dostępne komendy:

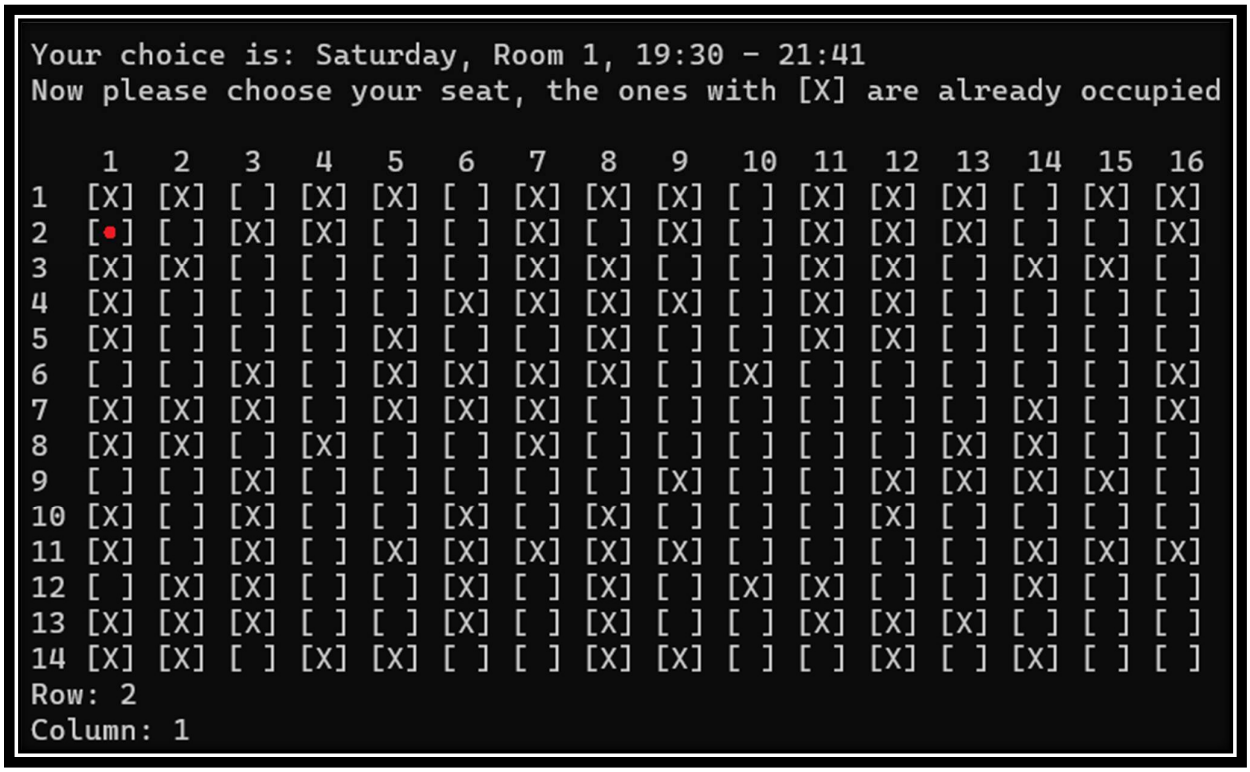
1. **Buy ticket for a movie** – ta opcja pozwala kupić bilet. Po jej wybraniu musimy wybrać czy interesuje nas konkretny dzień, czy film. Po wpisaniu, przechodzimy przez kolejne wybory, aż program wyświetli zawężoną naszymi wyborami listę filmu do wyboru.

```
Showing when and where movie "Top Gun: Maverick" is playing:

                Monday:

                Tuesday:

                Wendesday:
1.      Room 3, 13:05 - 15:16, polish subtitles, 3D

                Thursday:
2.      Room 1, 13:05 - 15:16, polish subtitles, 3D
3.      Room 1, 17:10 - 19:21, polish subtitles, 3D
4.      Room 1, 19:30 - 21:41, polish subtitles, 3D

                Friday:
5.      Room 1, 13:05 - 15:16, polish subtitles, 3D
6.      Room 1, 17:10 - 19:21, polish subtitles, 3D
7.      Room 1, 19:30 - 21:41, polish subtitles, 3D

                Saturday:
8.      Room 1, 13:05 - 15:16, polish subtitles, 3D
9.      Room 1, 17:10 - 19:21, polish subtitles, 3D
10.     Room 1, 19:30 - 21:41, polish subtitles, 3D

                Sunday:
11.     Room 1, 13:05 - 15:16, polish subtitles, 3D
12.     Room 2, 17:35 - 19:46, polish subtitles, 2D
Now please choose when and where you want to watch the movie by entering number of movie iteration:
```

Przykładowe wyszukanie filmu w okresie całego tygodnia

Mamy w niej napisane jakiego dnia i o jakiej godzinie jest grany film, a także inne przydatne informacje jak lokalizacja (audiowizualna) czy typ obrazu. Jak widzimy w tym przypadku wylosowany harmonogram nie wygenerował filmu *„Top Gun Maverick”* w poniedziałek i wtorek. Interesującą nas datę wybieramy wpisując numerek przy niej – np. wpisanie 10 oznacza że wybieramy: *pokój 1, 19:30 – 21:41, z polskimi napisami w 3D.*

Po tym przechodzimy do wyboru miejsca – wyświetla nam macierz miejsc na sali – te z [X] są zajęte (miejsca zajęte są losowo generowane w ramach realizmu).

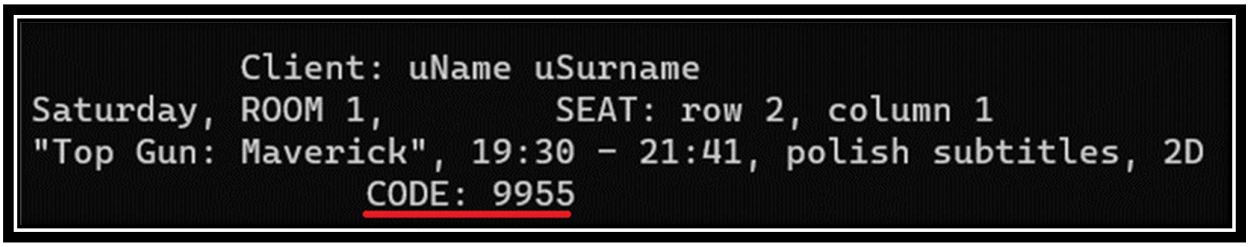


Macierz miejsc w sali kinowej, miejsca z [X] oznaczają zajęte miejsce

Miejsce wybieramy najpierw wpisując wiersz, a potem kolumnę która nas interesuję, w tym wypadku wybieramy miejsce zaznaczone czerwoną kropką. Jeśli wybierzemy wolne miejsca i spełniamy warunki wiekowe, bilet zostanie kupiony i zostanie wyświetlona odpowiednia informacja.

2. **Show your tickets** – wyświetla nam nasze kupione bilety, po wyświetleniu konieczne jest wpisanie jakiejkolwiek cyfry by kontynuować.

3. **Cancel your tickets** – usuwa bilet, którego CODE podamy, jest on umieszczony na samym dole biletu – poniżej przykładowy bilet, jeśli chcielibyśmy go usunąć należałoby wpisać 9955. Wpisanie **'0'** pokazuje nam nasze bilety.



Przykładowy bilet, ticket CODE zaznaczono na czerwono

4. **Show movies in play** – wyświetla jakie filmy są aktualnie grane – wpisanie numerku filmu powoduje wyświetlenie dodatkowych informacji o tym filmie.

5. **Show when and where movie is playing** – po wybraniu interesującego nas okresu czasu wyświetla nam kiedy i gdzie dany film jest puszczany, wraz z dodatkowymi informacji – działa to identycznie jak w pierwszej opcji menu, gdzie musieliśmy wybrać interesujący nas termin z listy filmów.

6. **Show week/day schedule** – wyświetla harmonogram całego tygodnia lub danego dnia. Jeśli zdecydujemy się na harmonogram tygodniowy, to ma on dwie opcje wyświetlania:
   I. *At once* – wszystkie dni są wyświetlane od razu
   II. *One day at a time* – wyświetlamy jeden dzień, po czym użytkownik musi wpisać dowolną cyfrę, by przejść do kolejnego dnia – dzieje się tak do momentu, aż użytkownik wyświetli wszystkie dni.

7. **Clear console window** – czyści nam konsolę ze wszelkiego tekstu.

Jeśli chcemy zmienić rolę, możemy po prostu wyjść z menu – tak jak wszędzie używając **'-1'.** Jeśli jednak byliśmy chociaż raz wcześniej klientem i zechcemy wrócić do tej roli – to zobaczymy następujące okienko:

```
You chose to be a client! You can create new user profile by inputting ,0,. Client's profiles below:
1. uName uSurname, age: 21
Create new profile or enter number:
```

Okienko wyboru profilu klienta (pokazuje się, jeśli mamy co najmniej jeden profil!)

Pozwala ona nam wrócić do wcześniej utworzonych profili albo stworzyć nowy wpisując **'0'.**

b) Rola menedżera kina

Po wybraniu tej roli pokazuje nam się menu menedżera.

```
**************************MAIN MENU**********************************
Welcome to cinema manager menu, you can input '0' to get instructions how this menu works
AVAIABLE COMMANDS:
1. Create/delete movie or show movies list.
2. Create/delete room or show all rooms.
3. Initialize/delete cinema opening days or show them.
4. Generate new schedule, manually add/delete movies, show schedules.
5. Clear console window.

Choose what you want to do by writing number:
```

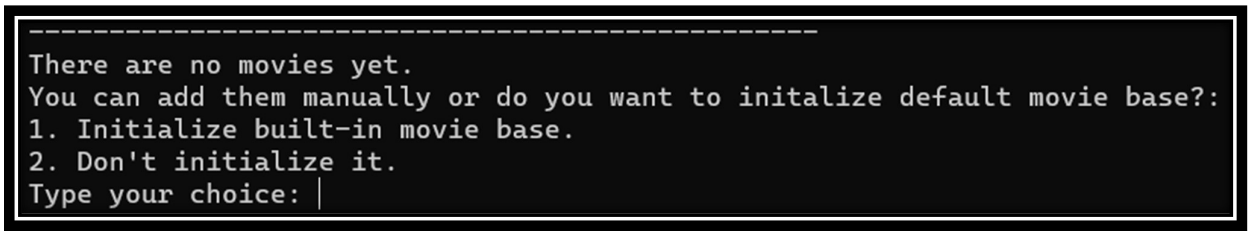
Menu menedżera kina

Po wpisaniu **'0'** użytkownikowi zostanie wyświetlona instrukcja. Jednakże tutaj nie będziemy jej przybliżać.

Tutaj na początku chciałbym ustalić pewne nazewnictwo ustaleń, niech **[MB]** oznacza **bazę filmów**, natomiast **[RB] bazę sal kinowych**.

Dostępne komendy:

1. **Create/delete movie or show movies list** – jeśli odpalamy to podmenu a nie mamy żadnych zainicjalizowanych filmów, czyli żadnych filmów w [MB], to wyświetla się następujący komunikat:



Okienko inicjalizacji domyślnie wbudowanej MB

Pyta się on czy chcemy zainicjalizować wbudowaną bazę filmów, która wygląda następująco:

- *Megamind*, Animated, duration: 1h36m, minAge: 9+, PL DUB, 3D
- *Shrek*, Animated, 1h30m, 7+, PL DUB, 3D
- *Avengers: Endgame*, Action, 3h01m, 16+, PL SUB, 3D
- *Top Gun: Maverick*, Action, 2h11m, 16+, PL SUB, 3D
- *The Nun 2*, Horror, 1h40m, 16+, PL SUB, 2D
- *IT: Chapter Two*, Horror, 2h40m, 16+, PL SUB, 2D
- *Forrest Gump*, Comedy, 2h21m, 13+, PL SUB, 2D
- *The In-Laws 2*, Comedy, 1h35m, 13+, PL DUB, 2D

Jeśli zdecydujemy się by ją zainicjalizować to pojawią się one w [MB]

Niezależnie od wyboru następnie przenosimy się już do podmenu, które ma następujące funkcje:

I. ***Add new movie*** – pozwala dodać nowy film do [MB]. Należy pamiętać, że wpisanie **'-1'** tutaj powoduje powrót do głównego podmenu. Użytkownik wpisuje w kolejności: nazwę, gatunek, ile godzin trwa, ile minut trwa, minimalny wiek, czy ma polski dubbing oraz czy może być wyświetlany w 3D.

II. Delete all/a movie

i. Delete all movies – usuwa wszystkie filmy z [MB]

ii. Delete a movie – usuwa wybrany przez użytkownika film

III. ***Reshow movies list*** – wyświetla listę filmów, czyli nasze [MB]. Jeśli wpiszemy numerek filmu to wypiszą się o nim dodatkowe informacje – czyli gatunek, czas trwania, minimalny wiek, typ obrazu i lokalizacje

2. **Create/delete room or show all rooms** – jeśli odpalamy to menu bez żadnych zainicjalizowanych sal, czyli [RB] jest pusta to wyświetli się podobny komunikat jak zostało to już opisane w poprzednim podmenu (1.).

   Będzie on oczywiście pytał czy użytkownik chce zainicjalizować wbudowaną bazę sal, która wygląda następująco:

   - Number 1, row: 14, column: 16, can display 3D
   - Number 2, 10x12, can't display 3D
   - Number 3, 10x10, can display 3D
   - Number 4, 7x8, can't display 3D

   Niezależnie od naszej decyzji zostaniemy przeniesieni do podmenu, gdzie mamy następujące opcje:

   I. ***Add new room*** – pozwala dodać nową salę do [RB]. Należy pamiętać, że wpisanie **'-1'** tutaj powoduje powrót do głównego podmenu. Użytkownik wpisuje w kolejności: ilość rzędów, kolumn oraz czy pokój obsługuje filmy 3D.

   II. Delete all/a room

      i. Delete all rooms – usuwa wszystkie sale z [RB]

      ii. Delete a room – należy wpisać numer sali, którą chcemy usunąć, należy pamiętać że jeśli pokoje wtedy ponumerują się od początku – tzn. jeśli usuniemy salę o numerze 1, to jakiś inny wskoczy na jego miejsce – jeśli były co najmniej dwie sale.

   III. ***Reshow rooms list*** – wyświetla listę sal, czyli nasze [RB].

3. **Initialize/delete cinema opening days or show them** – podmenu mające na celu ustalenie godzin otwarcia kina w dane dni.

   Bardzo ważne: po ustaleniu godzin należy koniecznie załadować pokoje do danych dni (opcja 5. w tym podmenu). Bez tego nie będziemy w stanie generować czy modyfikować harmonogramów!

   I. Automatically initialization - in all days cinema is open 13:00 - 23:00 – automatyczna, powoduje że wszystkie dni są otwarte w godzinach 13:00 – 23:00.

   II. Manually initialization - you can pick cinema open hours in all/some days – użytkownik najpierw wpisuje domyślne godziny otwarcia, które będą zastosowane do wszystkich dni, których nie wybierze. Następnie pokazuje się lista dni – użytkownik wybiera, **w kolejności od najwcześniejszego** (nie ma możliwości po wybraniu środy, zmienić godzin poniedziałku), dni których godziny otwarcia chce pozmieniać. Użytkownik wraca do podmenu, kiedy skończy edytować niedzielę lub wcześniej, jeśli wpisze **'0'**.

   III. Delete all days – resetuje wszystkie godziny otwarcia (tracony jest również harmonogram).

   IV. Show days' cinema opening hours – pokazuje godziny otwarcia kina w danych dniach.

   V. Load all initialiazed rooms to days' tracks – używać dopiero po ustaleniu godzin otwarcia kina, inicjalizuje wszystkie pokoje obecne w tej chwili w [RB] do naszych dni.

4. **Generate new schedule, manually add/delete movies, show schedules** – podmenu odpowiedzialne za wszelakie zmiany jak I wyświetlanie harmonogramu.

    I. ***Regenerate schedule*** – pozwala nam zregenerować (wylosować ponownie) harmonogram sali, dnia lub całego tygodnia.

        Na początku dokonujemy wyboru czy chcemy zregenerować tydzień czy cały dzień, jeśli chcemy tydzień to wybieramy stosowną opcje, po czym następuje jego ponowne losowanie.

        Wybranie dnia natomiast najpierw każe nam wybrać konkretny dzień, gdzie używając **'0'** jesteśmy w stanie wyświetlić sobie harmonogram dowolnego dnia. Gdy już zdecydowaliśmy się na konkretny dzień, to wybieramy czy chcemy zregenerować cały dzień czy salę. Jeśli to pierwsze, to zregenerowany zostanie cały dzień, jeśli jednak wybraliśmy salę to wyświetla się nam ich lista – wybieramy wpisując numerek sali. Możemy również wyświetlić tutaj harmonogram danej sali wpisując **'0'**.

    II. ***Manually add new movie to schedule*** – pozwala nam ręcznie dodać film w danym okresie czasu.

        Tutaj najpierw wybieramy konkretny dzień, w którym chcemy dodać nasz film, używając **'0'** jesteśmy w stanie wyświetlić sobie harmonogram dowolnego dnia. Kolejnym etapem jest wybór sali, których harmonogramy również możemy podglądać tym samym przyciskiem. Następnym etapem jest wybór filmu – tutaj **'0'** służy do tego, by móc dowiedzieć się więcej informacji o filmie. Na sam koniec wpisujemy godzinę i minutę rozpoczęcia. Jeśli film nie wadzi innym filmom, a także nie zaczyna się wcześniej niż 5 minut przed zakończeniem ostatniego filmu – jest ustalany.

        Użytkownik na sam koniec może wybrać do którego etapu chcę wrócić: wyboru godzin, filmu, sali, dnia czy do głównego podmenu – ma to na celu ułatwienie dodawania seryjnie filmów ręcznie, oraz przyśpieszyć czas powrotu do podmenu.

    III. ***Delete movie from schedule*** – pozwala ręcznie usuwać filmy z harmonogramu

        Pierwszym wyborem jest wybór dnia, klasycznie mamy harmonogram dnia pod '0'. Później dokonujemy wyboru filmu, natomiast pod '0' możemy wyświetlić harmonogram dowolnej salki. Gdy wybierzemy film wyświetla nam się lista iteracji filmów w tym okresie, wybieramy jeden z nich, a ten jest usuwany.

        Użytkownik na sam koniec może wybrać do którego etapu chcę wrócić: iteracji filmu, samego filmu, dnia czy do głównego podmenu – ma to na celu ułatwienie seryjnego usuwania filmów ręcznie, oraz przyśpieszyć czas powrotu do podmenu.

    IV. ***Reset schedule*** – pozwala nam zresetować harmonogram całego tygodnia, dnia lub salki.

    V. ***Show week/day schedule*** – wyświetla nam harmonogramy, działa dokładnie tak samo jak 6. z menu klienta

    VI. ***Show when and where movie is playing*** – wyświetla kiedy i gdzie dany film jest grany – działa dokładnie tak samo jak 6. z menu klienta

5. **Clear console window** – czyści nam konsolę ze wszelkiego tekstu.

4. Kompilacja

Projekt odpalany przez standardową kompilację. Testowany był tylko w systemie Windows.

5. Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

- *ValidityCheckFunctions.h, ValidityCheckFunctions.cpp* – deklaracja oraz implementacja funkcji sprawdzających typy wpisanych zmiennych oraz ich poprawność,
- *Time.h, Time.cpp,* - deklaracja oraz implementacja funkcji odpowiedzialnych za sprawdzanie poprawności, operacje matematyczne na czasie oraz jego wyświetlanie,
- *Movie.h, Movie.cpp* – deklaracja oraz implementacja klasy *Movie,*
- *Minute.h, Minute.cpp* – deklaracja oraz implementacja klasy *Minute,*
- *Day.h, Day.cpp* – deklaracja oraz implementacja klasy *Day,*
- *Seat.h, Seat.cpp* – deklaracja oraz implementacja klasy *Seat,*
- *Room.h, Room.cpp* – deklaracja oraz implementacja klasy *Room,*
- *Track.h, Track.cpp* – deklaracja oraz implementacja klasy *Track,*
- *Order.h, Order.cpp* – deklaracja oraz implementacja klasy *Order,*
- *User.h, User.cpp* – deklaracja oraz implementacja klasy *User,*
- *CinemaInterface.h, CinemaInterface.cpp* – deklaracja oraz implementacja klasy *CinemaInterface,*
- *Cinema booking system.cpp* – plik źródłowy, w którym znajduję się główny main.

6. Opis klas

W projekcie utworzono następujące klasy:

- **`ValidityCheckFunctions`** – klasa funkcji sprawdzających.
  - `bool isInRange(int startRange, int stopRange, int variable, bool inform)` – sprawdza czy `variable` znajduje się w przedziale [`startRange, stopRange`], jeśli nie – zwraca `false`, `inform` pozwala wyłączyć komunikaty tekstowe,
  - `bool isCommand(vector<string> commands, string variable, bool inform)` – jeśli `variable` nie występuje w `commands`, zwraca `false`, `inform` pozwala wyłączyć komunikaty tekstowe,
  - `bool isInputInt(int& inputVariable, bool inform)` – po wywołaniu użytkownik wpisuje co chce z klawiatury, jeśli nie jest to liczbą – zwracany jest `false`, `inform` pozwala wyłączyć komunikaty tekstowe.

- **`Time`** – klasa zawierające przydatne funkcje związane z czasem i operacjami na nim.
  - `bool checkIsTimeCorrect(int hour, int mins, bool inform)` – sprawdza czy podane wartości są poprawne, jeśli są zwraca `true`, `inform` pozwala wyłączyć komunikaty tekstowe,

- `void addTimes(int hour1, int min1, int hour2, int min2, int& sumHour, int& sumMin)` – dodaje do siebie godziny i minuty i zwraca wynik w `sumHour` i `sumMin`,
- `void preetyHours(int startHour, int startMin, int endHour, int endMin, bool endl)` – wypisuje godzinę podaną w argumentach, `endl` powoduje przejście do kolejnej linijki (jak `endl` w `cout`),
- `void minToPeriod(int eventStart, int eventEnd, int& eventStartHour, int& eventStartMin, int& eventEndHour, int& eventEndMin)` – po podaniu minuty początku i końca wydarzenia, zamienia nam je na godziny.

- **`Movie`** – klasa reprezentująca film.
  - `string get_name(void) const` – getter pola `fname`, zwracający nazwę filmu,
  - `string get_genre(void) const` – getter pola `fgenre`, zwracający gatunek filmu,
  - `int get_hours(void) const` – getter pola `fhours`, zwracający ile godzin trwa film,
  - `int get_minutes(void) const` – getter pola `fminutes`, zwracający ile minut trwa film,
  - `int get_minAge(void) const` – getter pola `fminAge`, zwracający minimalny wiek wymagany do obejrzenia filmu,
  - `bool get_PLdub(void) const` – getter pola `fPLdub`, zwracający informację o lokalizacji filmu,
  - `bool get_is3D(void) const` – getter pola `fis3D`, zwracający informację o tym czy film jest 3D,
  - `void set_minAge(int minAge)` – setter pola `fminAge`,
  - `void set_PLdub(bool PLdub)` – setter pola `fPLdub`,
  - `void set_is3D(bool is3D)` – setter pola `fis3D`,
  - `void info()` – wypisuje parametry filmu.

- **`Minute`** – klasa reprezentująca minutę.
  - `bool get_isUsed() const` – getter pola `fisUsed`, zwracacający informację czy ta minuta jest zajęta (np. przez jakieś wydarzenie),
  - `string get_byWhat() const` – getter pola `fbyWhat`, zwracający informację przez co minuta jest zajęta,
  - `void set_isUsed(bool isUsed)` – setter pola `fisUsed`,
  - `void set_byWhat(string byWhat)` – setter pola `fbyWhat`.

- **`Day`** – klasa reprezentująca dzień, w której można ustawiać wydarzenia.
  - `vector<Minute> get_time(void) const` – getter wektora `czas`, który reprezentuje nam 1440 minut (czyli jeden cały dzień),

- `void findEventLength(int searchStart, int searchStop)` – wyszukuje i wypisuje kiedy kończy się wydarzenie rozpoczynające się w `searchStart`, przedział szukania jest ograniczony do [`searchStart`, `searchStop`],
- `void getEventLength(string eventName, int& eventStart, int& eventEnd, int searchStartHour, int searchStartMin, int searchStopHour, int searchStopMin)` – wyszukuje i zwraca nam minutę rozpoczęcia i zakończenia wydarzenia w zadanym przedziale czasowym [`searchStart`, `searchStop`],
- `bool setEvent(string eventName, int startHour, int startMinute, int endHour, int endMinute, bool inform)` – ustawia nam wydarzenie o nazwie `eventName` w podanym przez nas czasie, `inform` pozwala wyłączyć komunikaty tekstowe, zwraca `true`, jeśli udało się je ustawić
- `void deleteEvent(string eventName, int startHour, int startMinute, int endHour, int endMinute, bool inform)` – wyszukuje i usuwa nam wydarzenie w przedziale czasowym [`start`, `end`], `inform` pozwala wyłączyć komunikaty tekstowe,
- `void changeEventHours(string eventName, int newStartHour, int newStartMinute, int newEndHour, int newEndMinute, int oldStartHour, int oldStartMinute, int oldEndHour, int oldEndMinute, bool inform)` – zmienia czas w jakim odbywa się istniejące wydarzenie z `oldHours` na `newHours`,
- `void printOut(int startHour, int startMin, int endHour, int endMin)` – wypisuje nam harmonogram dnia w przedziale czasowym [`start`, `end`],
- `void reset(bool inform)` – resetuje harmonogram całego dnia.

- **`Seat`** – klasa reprezentująca pojedyncze miejse na sali kinowej.
  - `bool get_isTaken(void) const` – getter pola `fisTaken` – zwraca informację czy miejsce jest zajęte,
  - `string get_byWho(void) const` – getter pola `fbyWho`, zwraca informację, na kogo wypisane jest miejsce,
  - `void set_isTaken(bool change)` – setter pola `fisTaken`,
  - `void set_byWho(string someone)` – setter pola `fbyWho`.

- **`Room`** – klasa reprezentująca salę kinową.
  - `int get_number(void) const` – getter pola `fnumber`, który zwraca numer pokoju,
  - `int get_rows(void) const` – getter pola `sRows`, czyli ilości rzędów/wierszy w macierzy siedzeń sali,
  - `int get_columns(void) const` – getter pola `sColumns`, czyli ilości kolumn w macierzy siedzeń sali,
  - `bool get_display3D(void) const` – getter pola `fdisplay3D`, zwracający informację, czy sala może wyświetlać 3D,
  - `map<vector<int>, vector<vector<Seat>>> get_eventSeats(void) const` – getter mapy `eventSeats`, w której są macierze siedzeń dla wydarzeń odbywających się w sali,

- `void set_eventSeats(map<vector<int>, vector<vector<Seat>>> feventSeats)` – setter mapy `eventSeats`,
- `void set_number(int number)` – setter pola `fnumber`,
- `void set_rows(int rows)` – setter pola `sRows`,
- `void set_columns(int columns)` – setter pola `sColumns`,
- `void set_display3D(bool canDisplay)` – setter pola `fdisplay3D`,
- `void randomlyTakeSeats(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin)` – losowo zajmuje miejsca w macierzy siedzeń w wydarzeniu odbywającym się dokładnie w tym czasie,
- `void displaySeats(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin)` – wyświetla macierz siedzeń wydarzenia odbywającego się w podanym czasie,
- `void onlyRoomInfo(bool displaySeats)` – wyświetla informację o pokoju, `displaySeats` pozwala wyświetlić pustą macierz siedzeń sali,
- `void info(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin)` – wyświetla informację o sali, razem z macierzą siedzeń wydarzenia odbywającego się w podanym czasie,
- `void newSeatMatrix(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin)` – tworzy nową unikalną macierz siedzeń dla wydarzenia odbywającego się w podanym czasie,
- `void deleteSeatMatrix(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin)` – usuwa macierz siedzeń wydarzenia odbywającego się w z zadanym czasie,
- `void changeSeatMatrixHours(int oldEventStartHour, int oldEventStartMin, int oldEventEndHour, int oldEventEndMin, int newEventStartHour, int newEventStartMin, int newEventEndHour, int newEventEndMin)` – macierz siedzeń wcześniej przypisana do wydarzenia odbywającego się w czasie `oldHours` zostaje przypisana do wydarzenia odbywającego się w `newHours`,
- `bool isSeatFree(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin, int row, int column)` – sprawdza czy dane miejsce jest zajęte w wydarzeniu odbywającym w czasie `eventHours`, jeśli jest wolne to zwraca `true`,
- `void takeSeat(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin, int row, int column, string surname)` – zajmuje zadane miejsce w wydarzeniu odbywającym się w czasie `eventHours`,
- `void freeSeat(int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin, int row, int column)` – zwalnia zadane miejsce jeśli jest zajęte dla wydarzenia odbywającego się w czasie `eventHours`,
- `void printOutEventSeats()` – wypisuje wszystkie czasy, które posiadają unikalne macierze siedzeń.

- **`Track`** – klasa reprezentująca dzień, w którym kino jest otwarte.

- `map<int, Room> get_rooms(void) const` – getter mapy `rooms`, czyli pokoi jakie są dostępne danego dnia kinowego,
- `map<int, Day> get_roomTracks(void) const` – getter mapy `roomTracks`, w którym zawarte są harmonogramy sal kinowych,
- `int get_openTime(void) const` – getter zwracający minutę w której otwierane jest kino,
- `int get_closeTime(void) const` – getter zwracający minutę, w której kino jest zamykane,
- `void set_rooms(map<int, Room> m_rooms)` – setter pozwalający ustawić mapę `rooms`,
- `void addRoom(Room room)` – dodaje `room` do naszej mapy `rooms`,
- `bool setMovie(Room room, Movie movie, int eventStartHour, int eventStartMin, bool inform)` – ustawia film w zadanych parametrach, zwraca `true`, gdy udało się go ustawić, `inform` pozwala wyłączyć komunikaty tekstowe,
- `void deleteMovie(Room room, Movie movie, int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin, bool inform)` – usuwa film w danej sali w czasie zadanym przez użytkownika,
- `void changeMovieHours(Room room, Movie movie, int newEventStartHour, int newEventStartMin, int oldEventStartHour, int oldEventStartMin)` – zmienia czas trwania filmu z `oldEventStart` na `newEventStart`,
- `void showRoomSchedule(Room room, int eventStartHour, int eventStartMin, int eventEndHour, int eventEndMin)` – wyświetla harmonogram sali `room` w zadanym przez użytkownika czasie,
- `void resetRoomSchedule(Room room, bool inform)` – resetuje harmonogram sali `room`, `inform` pozwala wyłączyć komunikaty tekstowe,
- `void showTodaysSchedule(bool startText)` – pokazuje harmonogram całego dnia kinowego, czyli wszystkich sal po kolei, `startText` steruje wyświetlaniem komunikatu tekstowego,
- `void resetTodaysSchedule(bool inform)` – resetuje harmonogram całego dnia kinowego,
- `map<vector<int>, vector<int>> searchForMovie(Movie movie, bool wantStartText, bool numberSearched, int numberSearchedStart)` – szukamy danego filmu w danym dniu kinowym, `wantStartText` odpowiada za wyświetlanie komunikatu startowego, `numberSearched` przełącza nam iterowanie numerkami kolejnych znalezionych terminów, kiedy film jest grany, a `numberSearchedStart` to liczba od której zaczniemy nasze wypisywanie tych iteracji.

- **`Order`** – klasa reprezentująca bilet kinowy.
  - `int get_Code(void) const` – getter pola `fCode`, czyli unikalnego losowanego kodu biletu,
  - `vector<int> get_hours(void) const` – getter zwracający godziny, w których trwa film na który wypisany jest bilet,

- `vector<int> get_misc(void) const` – getter zwracający informację o miejscu na które wypisany jest bilet, numer sali, oraz o lokalizacji i typie obrazu filmu,
- `vector<string> get_strings(void) const` – getter zwracający nazwę filmu, dzień oraz na kogo wypisany jest bilet,
- `void ticketInfo()` – wypisuje informacje zawarte w bilecie,
- `bool buyTicket(Track& track, Room room, string day, int row, int column, Movie movie, int startHour, int startMin, int endHour, int endMin, bool wantPLdub, bool want3D, string userSurname, int userAge)` – zmienia parametry biletu na podane powyżej.

- **`User`** – klasa reprezentująca klienta kina.
  - `string get_surname(void) const` – getter pola `fsurname`,
  - `int get_age(void) const` – getter pola `fage`,
  - `void buyTicket(Track& track, Room room, string day, int row, int column, Movie movie, int startHour, int startMin, int endHour, int endMin, bool wantPLdub, bool want3D, string userSurname, int userAge)` – kupowanie biletu o podanych parametrach,
  - `void showTickets()` – wypisuje bilety jakie posiada klient,
  - `bool deleteTicket(vector<Track>& week, int ticketCode)` – usuwa bilet o podanym `ticketCode`, `week` jest podany, by móc zwolnić miejsce z macierzy siedzeń.

- **`CinemaInterface`** – klasa zajmująca się interfejsem użytkownika.
  - `void start()` – odpala cały funkcjonalny program wraz z interfejsem użytkownika.

7. Inne

brak