

CinemaFinder auxilia o utilizador na pesquisa de um local e horário para assistir a um filme. Esta classe inclui informação sobre os cinemas existentes (fila de prioridade *cinemas*) e filmes (tabela de dispersão *films*).

A classe **Cinema** representa um local de cinema. Um cinema possui um nome (*name*), a distância deste cinema ao local onde se encontra o utilizador (*distance*), o número de salas existentes (*numberOfRooms*) e uma lista de serviços disponíveis no local: “estacionamento”, “restauração”, ... (*services*).

A classe **Cinema** contém ainda informação sobre os filmes em exibição, horários e sala (classe **FilmTime**) na árvore binária de pesquisa *timetable*. Numa mesma sala, não pode existir mais que um filme em exibição no mesmo horário (por uma questão de simplificação, o tempo de duração de um filme não é considerado). Diferentes salas podem exibir o mesmo filme.

A informação presente na árvore *timetable* está ordenada crescentemente por horário do filme e, em caso de empate, crescentemente por ID da sala de cinema.

A fila de prioridade *cinemas* deve conter no seu início a sala de cinema mais próxima e, em caso de empate, a que possui maior número de serviços disponíveis.

As classes **CinemaFinder**, **Cinema**, **FilmTime** e **Film** estão parcialmente definidas a seguir.

```
typedef tr1::unordered_set<FilmPtr, hFilmPtr,  
hFilmPtr> tabHFilm
```

```
class CinemaFinder {  
    priority_queue<Cinema> cinemas;  
    tabHFilm films;  
public:  
    CinemaFinder();  
    virtual ~CinemaFinder();  
    tabHFilm getFilms() const;  
    priority_queue<Cinema> getCinemas() const;  
    void addFilm(Film *f1);  
    void addCinema(const Cinema &c1);  
  
    class CinemaNotFound {};  
};
```

```
class FilmTime {  
    unsigned hour;  
    Film *film;  
    unsigned roomID;  
public:  
    FilmTime(unsigned h, Film *f, unsigned id);  
    virtual ~FilmTime();  
    unsigned getHour() const;  
    unsigned getRoomID() const;  
    Film* getFilm() const;  
    void setFilm(Film* f);  
    bool operator == (const FilmTime &ft) const;  
    bool operator < (const FilmTime &ft1) const;  
//TODO:
```

```
class Cinema {  
    string name;  
    unsigned distance;  
    unsigned numberOfRooms;  
    list<string> services;  
    BST<FilmTime> timetable;  
public:  
    Cinema(string nm, unsigned d, unsigned nr);  
    Cinema(string nm, unsigned d, unsigned nr,  
            list<string> ls);  
  
    virtual ~Cinema();  
    string getName() const;  
    unsigned getDistance() const;  
    list<string> getServices() const;  
    BST<FilmTime> getTimetable() const;  
    void addService(string servicel);  
    void addFilmTime(const FilmTime &ft1);  
    bool operator < (const Cinema & c1) const;  
//TODO  
};
```

```
class Film {  
    string title;  
    list<string> actors;  
public:  
    Film(string tit, list<string> a);  
    Film(string tit);  
    virtual ~Film();  
    string getTitle() const;  
    list<string> getActors() const;  
    void addActor(string actorName);  
};  
struct FilmPtr {
```

};

Film *film;

};

- a) Considere a classe **Cinema**, que possui um membro-dado *timetable*, que é uma árvore binária de pesquisa (BST) onde se encontram os filmes em exibição, horários e salas. A inicialização desta BST já é realizada no construtor da classe:

```
Cinema::Cinema(string nm, unsigned id, unsigned nr, list<string> ls):  
    name(nm), distance(d), numberOfRooms(nr), services(ls),  
    timetable(FilmTime(0, NULL, 0))  
{}
```

Implemente os operadores necessários à correta utilização da BST.

- a1) [2.5 valores] Implemente na classe **Cinema** o membro-função:

Film filmAtHour(unsigned &h1) const*

Esta função procura (na BST *timetable*) o filme que se encontra em exibição no horário *h1* (em qualquer sala). Se não existir nenhum filme na hora *h1*, retorna o filme imediatamente mais cedo e atualiza o argumento *h1* com a hora desse filme. Se não existir nenhum filme antes ou na hora *h1*, a função retorna *NULL*.

- a2) [2.5 valores] Implemente na classe **Cinema** o membro-função:

bool modifyHour(unsigned h1, unsigned room1, unsigned h2)

Esta função altera, na sala *room1*, a hora do filme de *h1* para *h2*. Se não existir na sala *room1* qualquer filme com exibição marcada para a hora *h1*, a função não faz nada e retorna *false*. Se a sala *room1* estiver ocupada na hora *h2*, a função não faz nada e retorna *false*. Se a alteração da hora for efetuada com sucesso, a função retorna *true*.

- a3) [3 valores] Implemente na classe **Cinema** o membro-função:

*unsigned addFilm(Film *f1, unsigned h1)*

Esta função adiciona o filme *f1* a exibir no horário *h1* na BST *timetable*. Considere que as salas são numeradas a partir de 1 e são ocupadas sequencialmente de acordo com o seu ID, isto é, para um mesmo horário, primeiro é usada a sala de ID=1, depois a sala de ID=2,... A função retorna o ID da sala a usar para a exibição do filme. Se não existir sala disponível no horário pretendido (note que o cinema tem um número fixo de salas), a função retorna 0.

- b) Considere a classe **CinemaFinder**, que possui um membro-dado *films*, que é uma tabela de dispersão onde se encontra informação sobre filmes (apontadores para a classe **Film**). Considere que dois filmes com o mesmo título são iguais.

Implemente os operadores necessários à correta utilização da tabela de dispersão.

- b1) [3 valores] Implemente na classe **CinemaFinder** o membro-função:

list<string> filmsWithActor(string actorName) const

Esta função retorna uma lista com os títulos dos filmes em que o actor de nome *actorName* participa.

b2) [3 valores] Implemente o membro-função da classe **CinemaFinder**:

void addActor(string filmTitle, string actorName)

Esta função adiciona o ator de nome *actorName* ao filme de título *filmTitle*. Se não existe nenhum filme com o título *filmTitle*, este novo filme é adicionado à tabela de dispersão *films*. Note que a classe **Film** já possui o membro-função *void addActor(string)*.

c) Considere a classe **CinemaFinder**, que possui um membro-dado *cinemas*, que é uma fila de prioridade onde se encontra informação sobre os diferentes cinemas (objetos da classe **Cinema**).

Implemente os operadores necessários à correta utilização da fila de prioridade.

c1) [3 valores] Implemente na classe **CinemaFinder** o membro-função:

string nearestCinema(string service1) const

Esta função retorna o cinema mais próximo que possui o serviço *service1*. Se não existir nenhum cinema com esse serviço, a função retorna "".

c2) [3 valores] Um empresário local decide investir com um novo serviço no cinema mais próximo de si, devido à crescente popularidade que a localidade apresenta. Implemente na classe **CinemaFinder** o membro-função:

void addServiceToNearestCinema(string service1, unsigned maxDistance)

Esta função adiciona o serviço *service1* à lista dos serviços disponíveis no cinema mais próximo (o que se encontra no início da fila de prioridade *cinemas*), desde que este se encontre a uma distância inferior a *maxDistance*. Se não existir nenhum cinema que satisfaça este requisito, a função lança a exceção **CinemaNotFound** (a exceção já se encontra definida na classe **CinemaFinder**). Assuma que pode sempre ser adicionado um serviço a um cinema que eventualmente já o tenha.