

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Отчет по заданию практикума

**«Система поддержки
бронирования и заселения гостиницы»**

Форманчук Антон

424 группа

Москва, 2022

Содержание

1	Уточнение постановки задачи	3
2	Диаграмма основных классов	4
3	Спецификации интерфейса	4
4	Диаграмма объектов	7
5	Инструментальные средства	7
6	Файловая структура	7
7	Пользовательский интерфейс	8

1 Уточнение постановки задачи

Небольшая гостиница содержит K номеров ($20 \leq K \leq 30$), различающихся по степени комфорта и стоимости: «люкс» — 500 у.е./день, «полулюкс» — 350 у.е./день, одноместные 100 у.е./день, простые двухместные — 160 у.е./день, двухместные с раскладным диваном — 200 у.е./день).

Требуется создать компьютерную систему, автоматизирующую управление занятостью номеров гостиницы. Система обрабатывает входной поток заявок двух видов:

1. заявки, бронирующие определенные типы номеров на определенный срок;
2. заявки на заселение в текущий момент на определенное время.

Система хранит информацию о фактической занятости всех номеров и о их занятости в ближайшие дни (учитываются уже оплаченные вперед дни), а также сведения о произведенной брони номеров, и использует все эти данные при обработке заявок. При бронировании номеров система автоматически формирует сообщение-подтверждение брони, а при выезде постояльцев она оформляет им счета.

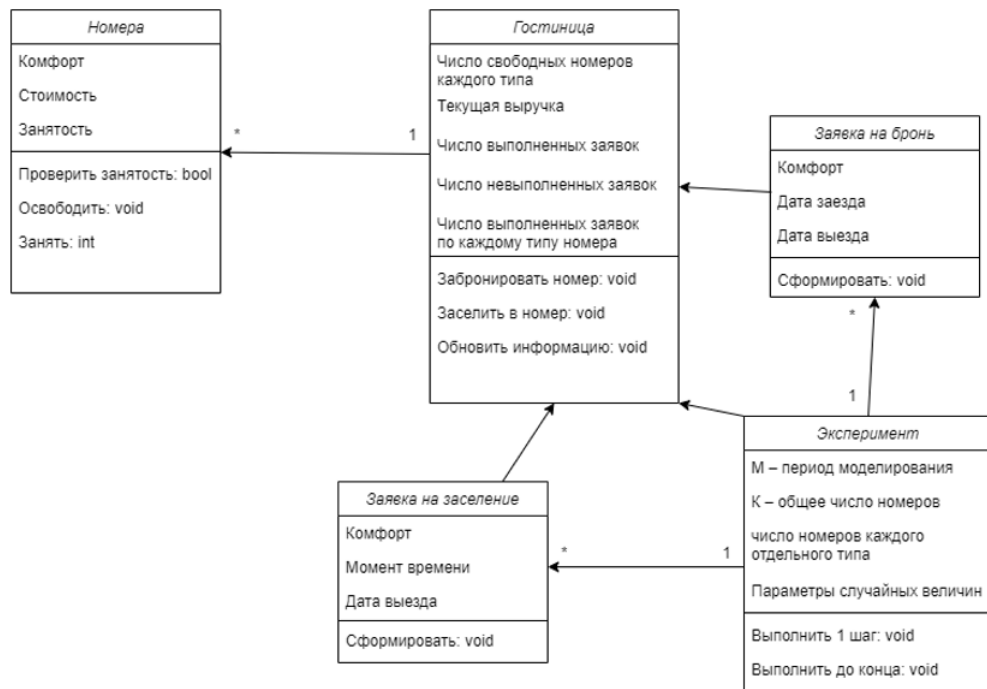
Стратегия обработки заявок строится так, чтобы добиться максимальной занятости гостиницы с целью увеличения ее прибыли. Для этого система гибко распоряжается номерным фондом: в частности, при нехватке нужных номеров можно использовать пустующие номера большей комфортности (по меньшей цене), например, при нехватке одноместных номеров можно поселить одного человека в двухместный номер (за 70% его стоимости).

Для тестирования построенной системы необходимо смоделировать входной поток заявок на бронирование и поселение. Вид и параметры каждой заявки определяются случайным образом. Количество поступающих заявок в каждый интервал времени также определяется случайным образом.

Период моделирования — M дней ($12 \leq K \leq 30$), шаг — 6 часов. Цель моделирования — изучение стратегий обработки заявок на заселение. В параметры моделирования следует включить: числа K и M , количество номеров каждой категории, характеристики используемых случайных величин.

В ходе моделирования и по его окончании система должна предоставлять всю необходимую информацию о занятости номеров гостиницы: выводится статистика заселения номеров, выполненных и невыполненных заявок, загруженность отдельных категорий номеров и гостиницы в целом, а также прибыль и текущее время.

2 Диаграмма основных классов



3 Спецификации интерфейса

```

//интерфейс класса, представляющего номер
class room{
public:
    //конструктор
    room(comfort comf_, int cost_);
    //занять номер
    int take_a_room();
    //проверить занятость номера
    bool check_free() const;
    //освободить номер
    void set_free();
};

//интерфейс класса, представляющего гостиницу
class hotel{
public:
    //конструктор по умолчанию
  
```

```

    hotel();
    //конструктор
    hotel(map <comfort, int> a);
    //забронировать номер определенного типа на конкретный срок
    void book(comfort, my_time, my_time);
    //обновить информацию по занятости номеров на текущее время
    void update_info(my_time time_);
    //получить число выполненных запросов на бронь
    int get_num_of_completed_requests() const;
    //получить число невыполненных запросов на бронь
    int get_num_of_unfulfilled_requests() const;
    //получить число выполненных запросов на бронь по типу номеров
    map <comfort, int> get_num_of_completed_requests_by_rooms() const;
    //получить текущую выручку
    int get_cur_revenue() const;
    //получить занятость номеров в данный момент
    map <comfort, pair<int, int> get_stats(my_time) const;
};
//интерфейс класса, представляющую заявку
class book_request{
public:
    //конструктор
    book_request(type_of_req_type, comfort comf_, my_time time1_, my_time
time2_);
    //получить тип номера
    comfort get_comfort() const;
    //получить время заселения
    my_time get_time1() const;
    //получить время выезда
    my_time get_time2() const;
    //получить тип заявки
    type_of_req get_type() const;
    //сформировать заявку в гостиницу
    void form(hotel& my_hotel);

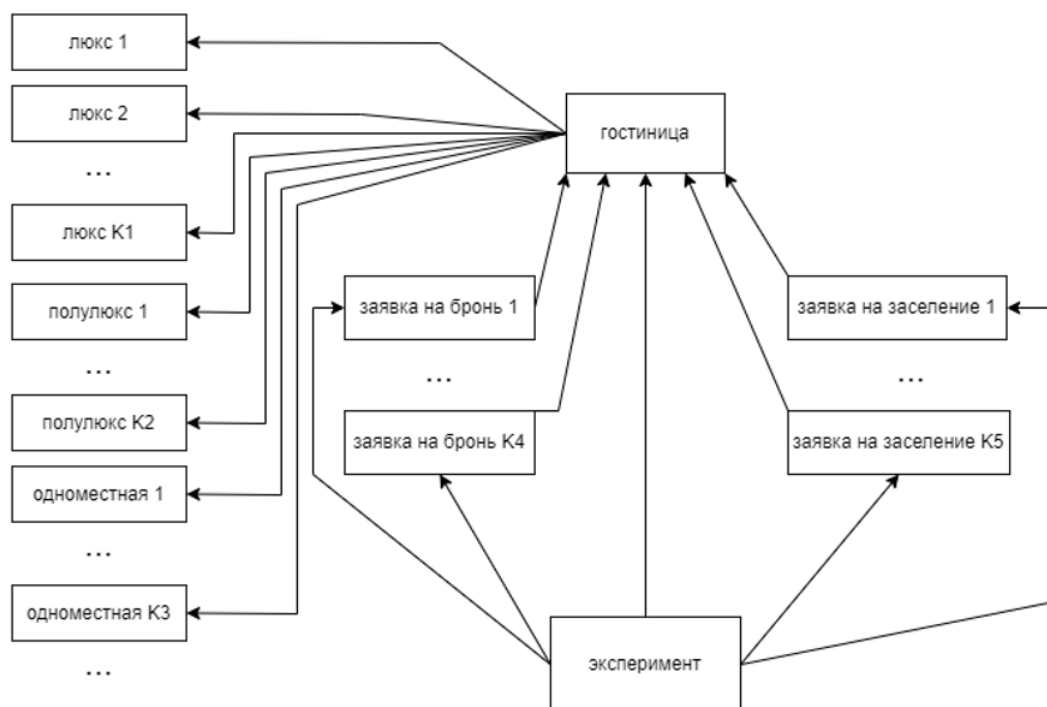
```

```

};
class experiment{
public:
    //конструктор по умолчанию
    experiment();
    //конструктор
    experiment(int M_, int K_, map <comfort, int> a);
    //получить текущее время
    my_time get_cur_time() const;
    //выполнить 1 шаг моделирования
    void complete_one_step();
    //выполнить все шаги моделирования
    void complete_all_steps();
    //получить число выполненных запросов на бронь
    int get_num_of_completed_requests() const;
    //получить все заявки в данный момент времени
    vector <book_request> get_vec_of_book_requests() const;
    //получить число невыполненных запросов на бронь
    int get_num_of_unfulfilled_requests() const;
    //получить число выполненных запросов на бронь по типу номеров
    map <comfort, int> get_num_of_completed_requests_by_rooms();
    //получить текущую выручку
    int get_cur_revenue() const;
    //получить занятость номеров в данный момент
    map <comfort, pair<int, int> get_stats() const;
};

```

4 Диаграмма объектов



5 Инструментальные средства

Язык разработки — C++

Среда разработки – Visual Studio Code

Используемые библиотеки – SFML

6 Файловая структура

Заголовочные файлы:

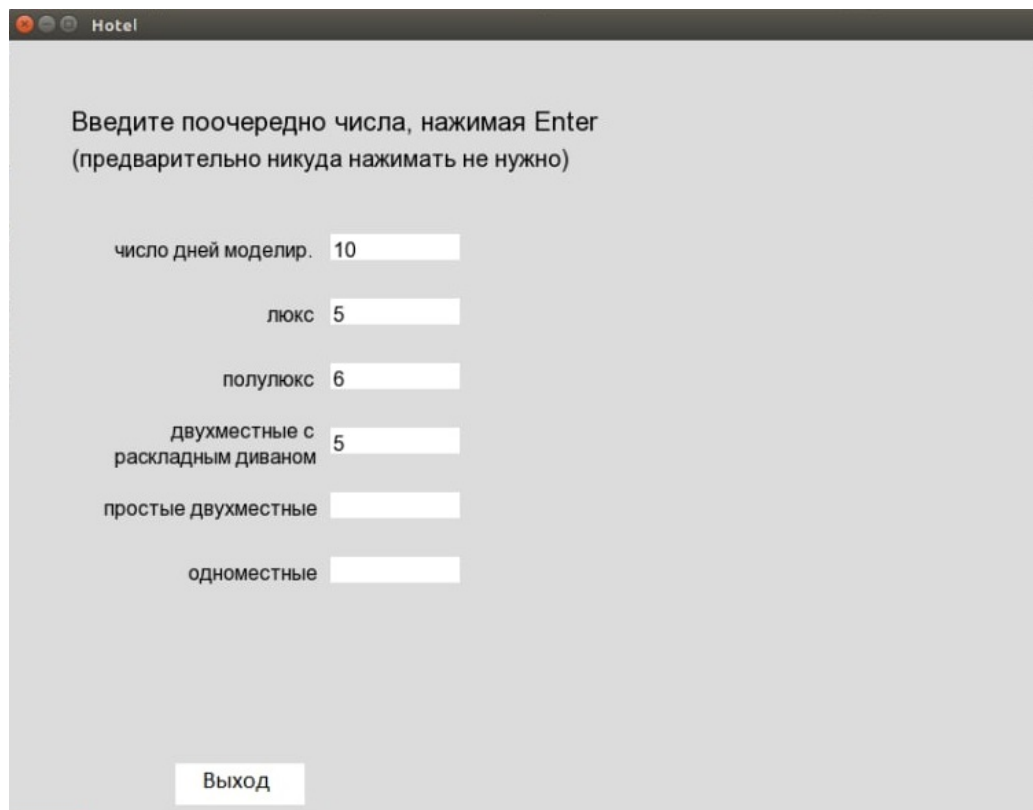
- globals.h — описание вспомогательных структур и функций
- book_request.h — описание соответствующего класса
- experiment.h — описание соответствующего класса
- hotel.h — описание соответствующего класса
- room.h — описание соответствующего класса

Исходники:

- `globals.cpp` — реализация вспомогательных функций
- `book_request.cpp` — реализация методов соответствующего класса
- `experiment.cpp` — реализация методов соответствующего класса
- `hotel.cpp` — реализация методов соответствующего класса
- `room.cpp` — реализация методов соответствующего класса
- `main.cpp` — реализация графического интерфейса и точка входа для запуска проекта

7 Пользовательский интерфейс

При запуске программы появляется окно:



Hotel

Введите поочередно числа, нажимая Enter
(предварительно никуда нажимать не нужно)

число дней моделир. 10

люкс 5

полулюкс 6

двухместные с раскладным диваном 5

простые двухместные

одноместные

Выход

Пользователю необходимо ввести параметры моделирования, нажимая Enter после ввода каждого числа. Затем появляется следующее окно:

Hotel

комфорт	общее число номеров	занято в данный момент
люкс	5	1
полулюкс	6	0
двухместные с раскладным диваном	5	1
простые двухместные	4	0
одноместные	8	1

Текущие дата и время

1 д., 12 ч.

Всего поступило заявок

10

Всего выполнено заявок

10

Невыполнено заявок

0

Выполнено заявок по номерам

люкс: 1

полулюкс: 2

2-мест. с див.: 4

прост. 2-мест.: 2

1-мест.: 1

Текущая выручка

3460

тип заявки	комфорт	время заселен.	время выселения
заселение	прост. 2-мест.	1 д., 12 ч.	4 д., 12 ч.
заселение	прост. 2-мест.	1 д., 12 ч.	4 д., 12 ч.
заселение	2-мест. с див.	1 д., 12 ч.	2 д., 12 ч.
бронирование	полулюкс	2 д., 12 ч.	3 д., 12 ч.
бронирование	полулюкс	3 д., 12 ч.	4 д., 12 ч.

Выход

Шаг

Выполнить до конца

Чтобы закончить моделирование и выйти из приложения, пользователю необходимо нажать "Выход". Нажатие кнопки "Шаг" выполняет 1 шаг моделирования, а кнопки "Выполнить до конца" — выполняет все оставшиеся шаги моделирования.