Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: «Объектно – ориентированное программирование»

**ОТЧЕТ**

По лабораторной работе №11.

                                                                       Выполнила: студент группы

ПИ(б)-2-21 Марат к. Жылдыз

Проверил: Мусабаев Э.Б.

**Бишкек – 2024**

**Лабораторный работа №11**

**Задания:**

17.14.Администратор гостиницы. Список номеров: класс, число мест. Список гостей: паспортные данные, даты приезда и отъезда, номер. Поселение гостей: выбор подходящего номера (при наличии свободных мест), регистрация, оформление квитанции. Отъезд: выбор всех постояльцев, отъезжающих сегодня, освобождение места или оформление задержки с выпиской дополнительной квитанции. Возможность досрочного отъезда с перерасчетом. Поиск гостя по произвольному признаку.

**Код программы:**

#include <iostream>

#include <Windows.h>

#include <vector>

#include <string>

#include <algorithm>

// Класс, представляющий комнату в отеле

class Room {

public:

std::string roomClass; // Класс комнаты (например, "Стандарт", "Люкс", "Сьют")

int capacity; // Вместимость комнаты

double price; // Цена за проживание в комнате

bool occupied; // Занята ли комната

// Конструктор класса Room

Room(std::string rc, int cap, double pr) : roomClass(rc), capacity(cap), price(pr), occupied(false) {}

};

// Класс, представляющий гостя отеля

class Guest {

public:

std::string passportData; // Данные паспорта гостя

std::string arrivalDate; // Дата прибытия гостя

std::string departureDate; // Дата отъезда гостя

int roomNumber; // Номер комнаты гостя

// Конструктор класса Guest

Guest(std::string pd, std::string ad, std::string dd, int rn)

: passportData(pd), arrivalDate(ad), departureDate(dd), roomNumber(rn) {}

};

// Класс администратора отеля

class HotelAdministrator {

private:

std::vector<Room> rooms; // Вектор комнат в отеле

std::vector<Guest> guests; // Вектор гостей отеля

public:

// Конструктор класса HotelAdministrator

HotelAdministrator(std::vector<Room> rs) : rooms(rs) {}

// Метод для заселения гостя в отель

void checkInGuest() {

std::cout << "\n=== Заселение гостя ===" << std::endl;

// Проверка наличия свободных номеров

bool anyAvailableRoom = false;

for (const auto& room : rooms) {

if (!room.occupied) {

anyAvailableRoom = true;

break;

}

}

if (!anyAvailableRoom) {

std::cout << "Извините, весь отель забронирован. Пожалуйста, вернитесь позже." << std::endl;

return;

}

// Вывод списка свободных номеров

std::cout << "Свободные номера:" << std::endl;

for (size\_t i = 0; i < rooms.size(); ++i) {

if (!rooms[i].occupied) {

std::cout << i + 1 << ". Класс номера: " << rooms[i].roomClass << ", Вместимость: " << rooms[i].capacity << ", Цена за проживание: " << rooms[i].price << std::endl;

}

}

// Выбор подходящего номера

int selectedRoom;

do {

std::cout << "Введите номер выбранного номера: ";

std::cin >> selectedRoom;

if (selectedRoom < 1 || selectedRoom > rooms.size() || rooms[selectedRoom - 1].occupied) {

std::cout << "Некорректный номер или номер уже занят. Пожалуйста, попробуйте еще раз." << std::endl;

}

} while (selectedRoom < 1 || selectedRoom > rooms.size() || rooms[selectedRoom - 1].occupied);

// Регистрация гостя

std::string passportData, arrivalDate, departureDate;

std::cout << "Введите данные паспорта гостя: ";

std::cin >> passportData;

std::cout << "Введите дату прибытия: ";

std::cin >> arrivalDate;

std::cout << "Введите дату отъезда: ";

std::cin >> departureDate;

rooms[selectedRoom - 1].occupied = true;

guests.push\_back(Guest(passportData, arrivalDate, departureDate, selectedRoom));

std::cout << "Гость успешно заселен." << std::endl;

}

// Метод для выселения гостя из отеля

void checkOutGuest() {

std::string currentDate;

std::cout << "\n=== Выселение гостя ===" << std::endl;

std::cout << "Введите текущую дату: ";

std::cin >> currentDate;

for (auto it = guests.begin(); it != guests.end();) {

if (it->departureDate == currentDate) {

rooms[it->roomNumber - 1].occupied = false;

it = guests.erase(it);

std::cout << "Гость успешно выселен." << std::endl;

return; // Возвращаемся после успешного выселения

}

else {

++it; // Инкрементируем только в случае, если не выселяем гостя на данной итерации

}

}

}

// Метод для досрочного выселения гостя из отеля с перерасчетом

void earlyCheckOutGuest() {

std::string passportData;

std::cout << "\n=== Досрочное выселение гостя ===" << std::endl;

std::cout << "Введите данные паспорта гостя: ";

std::cin >> passportData;

auto it = std::find\_if(guests.begin(), guests.end(), [&](const Guest& g) {

return g.passportData == passportData;

});

if (it != guests.end()) {

// Расчет суммы за проживание

double totalAmount = calculateTotalAmount(\*it);

std::cout << "Сумма за проживание: " << totalAmount << std::endl;

rooms[it->roomNumber - 1].occupied = false;

guests.erase(it);

std::cout << "Гость успешно выселен досрочно." << std::endl;

}

else {

std::cout << "Гость с данными паспорта " << passportData << " не найден." << std::endl;

}

}

// Метод для расчета суммы за проживание гостя

double calculateTotalAmount(const Guest& guest) {

int roomNumber = guest.roomNumber;

double pricePerNight = rooms[roomNumber - 1].price;

// Разбор дат прибытия и отъезда

// Предполагается, что даты записаны в формате "ГГГГ-ММ-ДД"

std::string arrivalDate = guest.arrivalDate;

std::string departureDate = guest.departureDate;

int nights = 1; // По умолчанию проживание как минимум на одну ночь

if (arrivalDate < departureDate) {

nights = std::stoi(departureDate.substr(8)) - std::stoi(arrivalDate.substr(8));

}

return pricePerNight \* nights;

}

// Метод для поиска гостя по данным паспорта, дате прибытия или дате отъезда

void searchGuest() {

std::string query;

std::cout << "\n=== Поиск гостя ===" << std::endl;

std::cout << "Введите данные паспорта гостя, дату прибытия или дату отъезда: ";

std::cin >> query;

for (const auto& guest : guests) {

if (guest.passportData == query || guest.arrivalDate == query || guest.departureDate == query) {

std::cout << "Гость найден. Данные паспорта: " << guest.passportData << ", Номер комнаты: " << guest.roomNumber << std::endl;

return;

}

}

std::cout << "Гость не найден." << std::endl;

}

// Метод для отображения меню администратора отеля

void showMenu() {

int choice;

do {

std::cout << "\n=== Меню администратора гостиницы ===" << std::endl;

std::cout << "1. Заселение гостя" << std::endl;

std::cout << "2. Выселение гостя" << std::endl;

std::cout << "3. Досрочное выселение гостя" << std::endl;

std::cout << "4. Поиск гостя" << std::endl;

std::cout << "5. Выход" << std::endl;

std::cout << "Введите ваш выбор: ";

std::cin >> choice;

switch (choice) {

case 1:

checkInGuest();

break;

case 2:

checkOutGuest();

break;

case 3:

earlyCheckOutGuest();

break;

case 4:

searchGuest();

break;

case 5:

std::cout << "Завершение работы..." << std::endl;

break;

default:

std::cout << "Некорректный выбор. Пожалуйста, попробуйте еще раз." << std::endl;

}

} while (choice != 5);

}

};

int main() {

// Создание вектора комнат и передача его в конструктор администратора отеля

std::vector<Room> rooms = { Room("Стандарт", 2, 50.0), Room("Люкс", 4, 100.0), Room("Сьют", 6, 150.0) };

HotelAdministrator admin(rooms);

// Установка кодировки для консоли Windows

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

// Отображение меню администратора отеля

admin.showMenu();

return 0;

}

**Результат программы:**

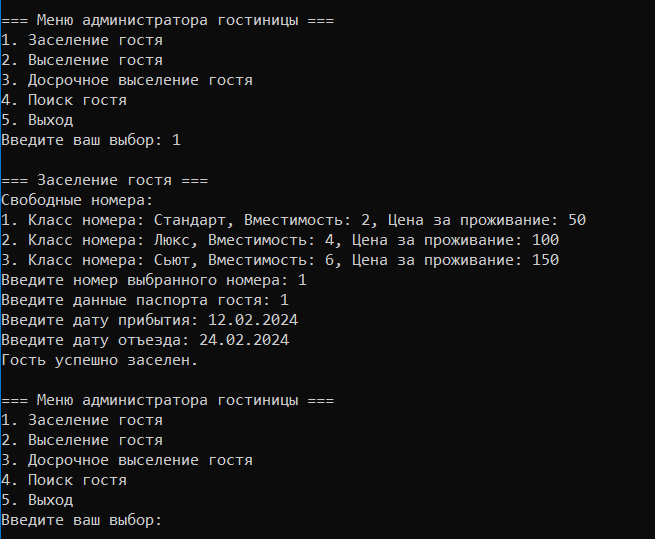


Рис.1 При выборе Заселение гостя

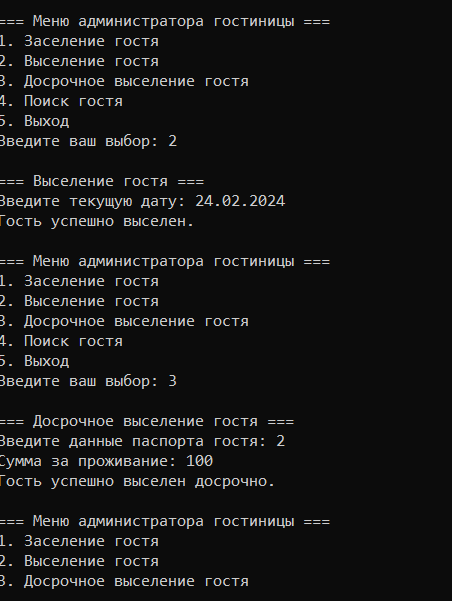


Рис.2 При выборе Выселение гостя

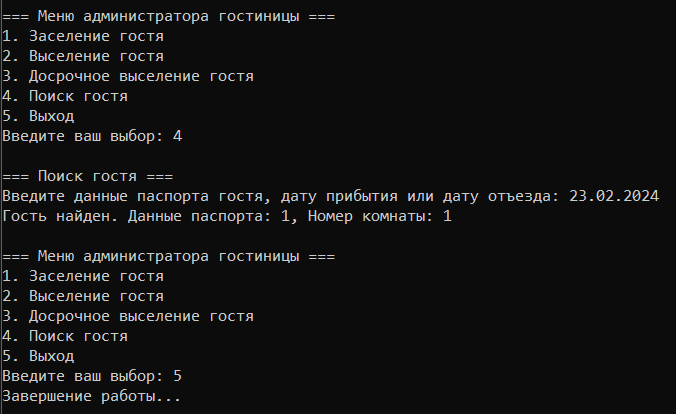


Рис.3 При Досрочного выселение гостя