**Документация по проект Хотел(Информационни системи)**

[Линк към GitHub repository](https://github.com/annie-prog/HotelManagement)

**Глава 1: Увод**

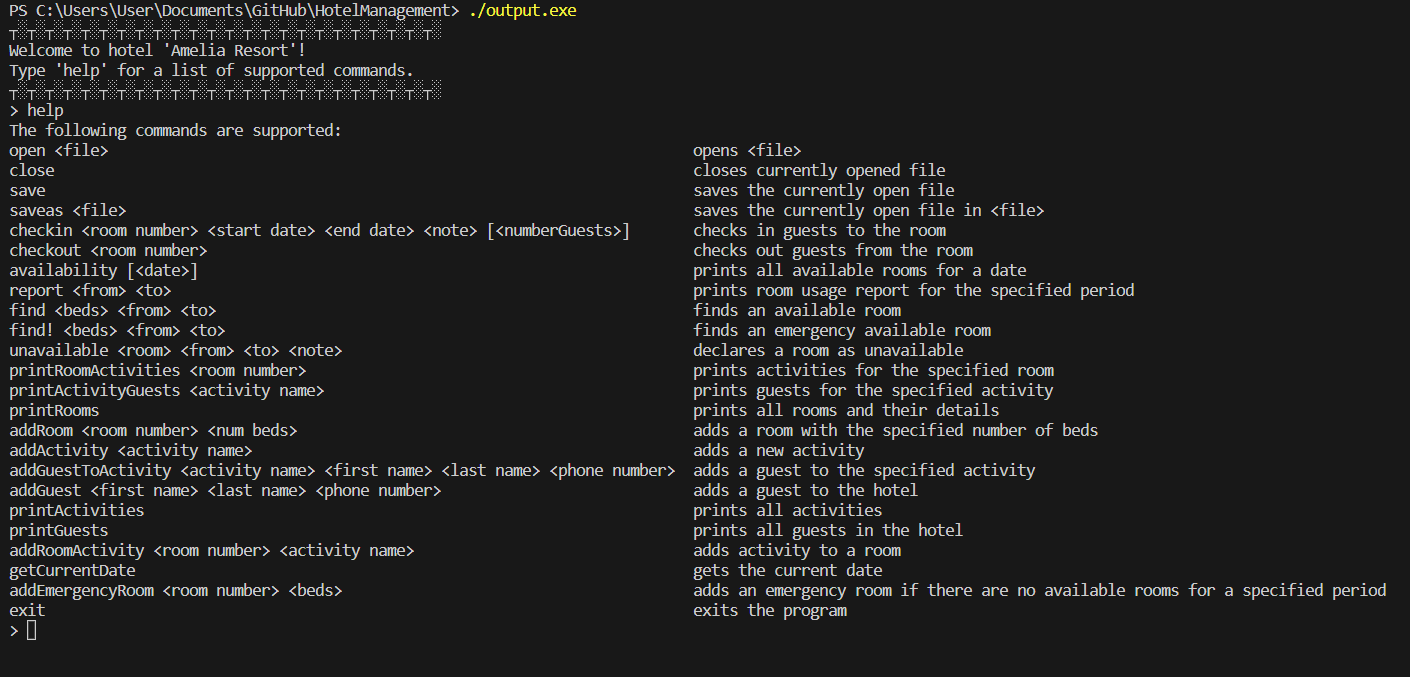
***1.1*** ***Описание и идея на проекта***:

Проектът „Хотел“ е комплексна система, разработена за управление на различни аспекти на хотелския бизнес. Предназначен е да предоставя възможност на потребителя да влезе в приложението и да извършва най-различни операции над хотелската система чрез конзолата, като данните за хотела се съхраняват във файл. Потребителят може да добавя стаи и гости в хотела, както и разни активности, с които на гостите да им е комфортно.

***1.2*** ***Цел и задачи на разработката***:

Целта на разработването на хотела е да се напише приложение с лесен за работа потребителски интерфейс. В приложението има основно меню, където потребителят може да въвежда командите, които иска да извърши върху системата на хотела. Проектът също така има за цел да оптимизира операциите, които се извършват на хотела, да улесни процесите на резервация, напускане на хотела, както и да предостави средства за анализ и отчет на използването на стаите и най-различни активности в хотела.

***Функционалност:***

* checkin <номер на стая> <от> <до> <бележка> [<брой гости>] - Регистриране в стая с номер от дата до дата и се добавя бележка. Незадължителният параметър задава броя на гостите, настанени в стаята. Ако той не е указан, се счита, че броят на настанените гости е равен на броя на леглата в стаята.
* availability [<дата>] - Извежда списък на свободните стаи на дата, ако не е зададена, се използва текущата дата.
* checkout <номер на стая> - Освобождаване на заета стая с номер <номер на стая>
* report <от> <до> - Извежда справка за използването на стаи в периода от дата <от> до дата <до>. Извежда се списък, в който за всяка стая, използвана в дадения период, се извежда и броя на дните, в които е била използвана.
* find <брой легла> <от> <до> - Намиране на подходяща свободна стая с поне <брой легла> на брой легла в периода от дата <от> до дата <до>. При наличие на повече свободни стаи се предпочитат такива с по-малко на брой легла.
* find! <брой легла> <от> <до> - Да се реализира алгоритъм, който предлага спешно намиране на стая за важен гост в случай на липса на свободни стаи за даден период. Алгоритъмът да предлага разместване на настанените от най-много две стаи.
* unavailable <номер на стая> <от> <до> <бележка> - Обявява стаята с номер <номер на стая> от дата <от> до дата <до> за временно недостъпна и се добавя бележка <бележка>. В стаята няма регистриран гост, но никой не може да бъде настанен в нея.
* open <име на файл> - отваряне на съдържанието на даден файл. Всички останали команди могат да се изпълняват само ако има успешно зареден файл.
* close – затваряне на текущия отворен документ. Изчиства се текущо заредената информация и след това програмата не може да изпълнява други команди, освен отваряне на файл.
* save - записва направените промени обратно в същия файл, от който са били прочетени данните.
* saveas <име на файл> - записва направените промени във файл, като позволява потребителя да укаже името му
* help – извежда кратка информация за поддържаните от програмата команди:
* 
* exit – излиза от програмата

***Задачи:***

* Да се създадат основни класове
* Да се реализират необходимите структури от данни и алгоритми
* Да се направят валидации на данните
* Да се направи структура за данните
* Да се записва цялата информация от програмата във файлове
* Да се зарежда вече запазената информация от файловете
* Да се направи потребителски интерфейс

***1.3 Структура на документацията***:

*Документацията е структурирана по следния начин:*

* Увод
* Преглед на предметната област
* Проектиране
* Реализация, тестване
* Заключение

**Глава 2: Преглед на предметната област**

***2.1.* *Основни дефиниции, концепции и алгоритми, които ще бъдат използвани***

В проектa ще бъдат използвани Comma-separated values (csv) файлове. Ще се обработват данни и ще се записват във файлове и след това ще се четат от буфер. Ще се използват класовете HotelApp и CSVUtils за по-лесно съхранение на последователност от команди. Ще има separation compilation за по-добра структура и по-добро четене на кода, както и множество от функции. Ще се използва и Singleton шаблона за създаване на единствен хотел. Също така ще се използва и алгоритъм за намиране на спешна стая за настаняване на важен гост при липса на места в хотела.

***2.2. Дефиниране на проблеми и сложност на поставената задача***

Проблемите, за които програмата е създадена да решава са:

* Преглед на стаи, гости, активности, използваемост на стаите
* Добавяне на стаи, гости, активности
* Преместване на гости в други стаи при необходимост
* Маркиране на стаи за заети, ако са в ремонт
* Обработване на файлове чрез команди от конзолата

Сложността на поставената задача беше подходяща за дадения срок за нейното изпълнение.

***2.3. Подходи, методи за решаване на поставените проблемите***

За решаване на гореспоменатите проблеми, програмата използва множество функции и класове.

* Статичния клас CSVUtils със статични полета за четене на файл и писане във файл чрез конзолата.
  + Функции за записване във файл
  + Функции за четене от файл
* Композиция на класовете Guest, Accommodation, Room, Activity, Reservation в класа HotelSystem, тъй като в хотела се съдържа всичко останало.
  + Функции за добавяне на резервации, активности, стаи, гости
  + Обработване на данните за резервации, активности, стаи и гости
  + Функции за изчистване на стаи и гости (checkout)
* Класът HotelApp, който се явява като Engine на проекта
  + Функции за обработване на командите.
  + Функция за обработване на командата в зависимост коя от всички е зададена.

**Глава 3: Проектиране**

***3.1. Обща архитектура***

Основни класове, които се използват в последствие за структурата на други класове:

* Guest
* Room
* Reservation
* Activity

Следващите класове са подредени по начина в който се използват един в друг:

* Accommodation – Има полета от тип Guest
* Activity – Има полета от тип Accommodation
* Reservation – Има полета от тип Accommodation
* Room – Има полета от тип Reservation, Activity, Accommodation
* HotelSystem – Има полета от тип Reservation, Activity, Accommodation, Room
* CSVUtils– Има статични полета от тип void
* HotelApp – Има поле от тип HotelSystem

И накрая остава основния файл на програмата, а именно HotelApp.cpp, която контролира входа и изхода(интерфейса на програмата) и също така останалите функционалности на платформата.

***3.2. Диаграми и слоеве с най-важните извадки от кода***

* Четене на гостите от файла:

void CSVUtils::SaveGuestsToCSV(const std::vector<std::vector<std::string>>& guests, const std::string& filename) {

    std::ofstream file(filename);

    if (!file.is\_open()) {

        std::cout << "Error with opening the file" << std::endl;

        return;

    }

    for (const auto& guest : guests) {

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

            file << guest[i];

            if (i < guest.size() - 1) {

                file << ",";

          }

        }

        file << std::endl;

    }

    file.close();

}

std::vector<std::vector<std::string>> CSVUtils::ReadGuestsFromCSV(const std::string& filename) {

    std::vector<std::vector<std::string>> guests;

    std::ifstream file(filename);

    if (!file.is\_open()) {

        std::cout << "Error with opening the file!" << filename <<std::endl;

return guests;

    }

    std::string line;

    while (getline(file, line)) {

        std::vector<std::string> guest;

        size\_t start = 0;

        size\_t end = line.find(',');

        while (end != std::string::npos) {

            guest.push\_back(line.substr(start, end - start));

            start = end + 1;

            end = line.find(',', start);

        }

        guest.push\_back(line.substr(start));

        guests.push\_back(guest);

    }

    file.close();

    return guests;}

void CSVUtils::PrintGuests(const std::vector<std::vector<std::string>>& guests) {

    for (const auto& guest : guests) {

        for (const auto& data : guest) {

            std::cout << data << " ";

        }

        std::cout << std::endl;

    }

}

* За стаите, резервациите и активностите е аналогично. Тук се използва класът CSVUtils в главния ни клас:

void HotelApp::ProcessAddGuestCommand(const std::vector<std::string>& tokens) {

    try {

        if (!isOpen) {

            std::cout << "No file is currently open." << std::endl;

            return;

        }

        if (tokens.size() < 4) {

            std::cout << "Invalid command. Please provide guest's first name, last name, and phone number." << std::endl;

            return;

        }

        std::string firstName = tokens[1];

        std::string lastName = tokens[2];

        std::string phoneNumber = tokens[3];

        Guest\* guest = new Guest(firstName, lastName, phoneNumber);

        hotel->AddGuest(guest);

        std::string filename = currentFile;

        std::ofstream file(filename, std::ios::app);

        if (!file.is\_open()) {

            std::cout << "Error opening the file " << filename << std::endl;

            return;

        }

        std::streambuf\* coutBuffer = std::cout.rdbuf();

        std::cout.rdbuf(file.rdbuf());

        std::vector<std::vector<std::string>> previousGuests = CSVUtils::ReadGuestsFromCSV(filename);

        previousGuests.push\_back({ firstName, lastName, phoneNumber });

        CSVUtils::SaveGuestsToCSV(previousGuests, filename);

        std::cout.rdbuf(coutBuffer);

        file.close();

        std::cout << "Guest " << firstName << " " << lastName << " has been added!" << std::endl;

    }

    catch (const std::invalid\_argument& e) {

        std::cerr << "Error: " << e.what() << std::endl;

        CSVUtils::ClearCSVFiles(currentFile);

        return;

    }

}

* Запазване на csv файла

void HotelApp::ProcessCloseCommand() {

    if (!isOpen) {

        std::cout << "No file is currently open." << std::endl;

        return;

    }

    std::cout << "Successfully closed " << currentFile << std::endl;

    isOpen = false;

    currentFile = "";

}

void HotelApp::ProcessSaveCommand() {

    if (!isOpen) {

        std::cout << "No file is currently open." << std::endl;

        return;

    }

    std::cout << "Successfully saved " << currentFile << std::endl;

}

void HotelApp::ProcessSaveAsCommand(const std::vector<std::string>& tokens) {

    if (!isOpen) {

        std::cout << "No file is currently open." << std::endl;

        return;

    }

    if (tokens.size() < 2) {

        std::cout << "Please provide a file path to save as." << std::endl;

        return;

    }

    std::string newFilePath = tokens[1];

    std::ifstream current(currentFile);

    std::string fileContent((std::istreambuf\_iterator<char>(current)), std::istreambuf\_iterator<char>());

    current.close();

    std::ofstream newFile(newFilePath);

    if (!newFile) {

        std::cout << "Error creating the new file." << std::endl;

        return;

    }

    newFile << fileContent;

    newFile.close();

    std::cout << "Successfully saved as " << newFilePath << std::endl;

}

**Глава 4: Реализация, тестване**

***Управление на паметта и алгоритми. Оптимизации.***

Програмата е написана по опростен начин. Всеки клас е дефиниран със своите полета и функционалности. Като за всеки реализиран метод са изпълнени стандарти, семантики и алгоритмите им са с оптимални сложности. Всеки клас отговаря собственоръчно за управлението на паметта си. Всеки един клас който използва динамична памет в последствие я освобождава при приключването на програмата. Използвани са материалите, изучавани по ООП през семестъра, също така са използвани минимален брой други библиотеки, освен <iostream>, <string> и <fstream>. Класовете са тествани с doctest.h.

**Глава 5. Заключение**

***5.1. Обобщение на изпълнението на началните цели***

Програмата се справя с работа с файлове. Успешно обработване на данните в хотела. Модификации на компонентите в хотела. Принтиране на данните в хотела.

***5.2. Насоки за бъдещо развитие и усъвършенстване***

В следващи издания на софтуера ще се извършат оптимизации на използваните структури и алгоритми, с цел програмата да придобие прототипен характер, подобен на някои от известните програми, разработени от софтуерни компании.