Димитър Тодоров Добренов

9MI0800108

ФМИ КН-1 , 1 курс, 1 група

Връзка към хранилището в GitHub: <https://github.com/dobrenov4/OOP_Project--Hotel.git>

(хранилището най-вероятно не е настроено напълно правилно – много от скоро използвам GitHub)

(по принцип на всяка функция съм написал кратки коментари на английски в кода, но според условията записвам и отделна документация тук)

Задачата ми беше да направя проект (Хотел) от предварително дадените условия. Реализирал съм решението по следния начин:

Създадени са 3 класа - [Date], [Room] и [Hotel]. Те биват използвани в главната програма която комбинира работата с тях и създава (много прост) конзолен интерфейс.

**[Date]**

Класът [Date] е създаден защото проектът изисква много работа с дати. Това създава необходимост от много проверки за валидност и усложнява работата. Лично сметнах, че ще бъде най-удобно да си създам собствен клас [Date], представляващ календарна дата и според условията на задачата е реализирано във формат ISO-8601 (yyyy-mm-dd). Това позволява всички проверки да бъдат изпълнявани вътрешно и значително улеснява работата с датите.

**Член данни на [Date]:**

short year – съхранява годината

short month – съхранява месеца

short day – съхранява деня

**Функции в [Date]:**

boolSet(const std::string &d) - Проверява дали стринга представлява валидна дата във формат ISO-8601 и ако е валидна, обектът приема стойност отговаряща на датата. Ако приеме датата връща стойност true.

void Clear() – Връща обекта до начално състояние. Това е датата първи януари нулевата година.

friend void Swap(Date &D1, Date &D2) – Разменя стойностите на обектите.

bool CheckLeapYear(const int y) – Проверява дали числото представлява високосна година. Ако да връща стойност true.

bool CheckDateValidity(const std::string d) - Проверява дали стринга представлява валидна дата във формат ISO-8601 и ако е валидна връща стойност true.

int NumberOfDaysBetwee(const Date D1, const Date D2) – Връща колко дни са изминали между двете дати.

Date TodaY() – Връща настоящата дата.

**[Room]**

Класът [Room], е създаден като основен носител на информация. Използван е в класа [Hotel]. Всяка стая съхранява своите данни, а съвкупността от стаи представлява хотел.

**Член данни на [Room]:**

bool status – Представлява статуса на стаята. (true – Това е обичайният статус на стаята и означава че е в експлоатация. false – Когато статусът на стаята е във състояние false, това означава, че стаята не е в експлоатация и не се предлага за клиенти.)

short roomNumber – номера на стаята

short numberOfBeds – колко легла има в стаята

short numberOfGuests – колко хора има в стаята в момента

Date from – началната дата на използване то на стаята

Date to – крайната дата на използване то на стаята

std::string note – предвидено за слагане на коментари, бележки или имена на гости

**Функции в [Room] :**

bool CheckAvailable() – Повярва дали обекта представлява свободна стая и ако връща стойност true.

void Clear() – Връща обекта до началните му стойности.

void Vacate() – Връща обекта до началните му стойности, като запазва само roomNumber и numberOfBeds.

friend void SwapStats(Room &R1, Room &R2) – Разменя стойностите на стаите като запазва само roomNumber и numberOfBeds.

friend void Swap(Room &R1, Room &R2) – Разменя всички стойности на обектите.

**[Hotel]**

Класът [Hotel] технически представлява динамичен масив от стаи (и има размер).

[Hotel] използва в себе си [Room], който от своя страна използва [Date]. Това позволява в main да бъдат реализирани функционалностите от условието.

(използването на този клас в main, малко или много, представлява цялото решение на задачата)

**Член данни на [Hotel]:**

int size – размера на масива от стаи

Room \*rooms - масив от стаи

**Функции в [Hotel]:**

void StatusRefresh();

void Sort() – Сортира масива от стаи (по номер на стаята).

void Clear() - Унищожава масива от стаи.

int getSize() – Връща размера на масива.

bool AddRoom(const Room &R) – Добавя нова стая към масива и ако е успешно връща true.

bool RemoveRoom(const short rn) – Премахва стая от хотела, като използва номера на стаята за да я идентифицира. Ако премахването е успешно, връща true.

int FindRoom(const short rn) – Намира стая по номера и връща индекса на позицията и в масива.

bool CheckIfRoomExists(const short rn) - Поверява дали съществува такава стая в масива, като използва номера на стаята за да я идентифицира. Ако същества връща true.

bool TxtPrintAll(const std::string FilePath) – Записва цялата информация за всички стаи в масива в подреден (табличен) вид в текстов файл, според подадения път. Ако записването е успешно връща true.

void ConsolePrintAll() – Изкарва цялата информация за всички стаи в масива в подреден (табличен) вид на конзолата.

int IntLen(int X) – Връща дължината на подаденото цяло число. (колко цифрено е числото)

Лично по мое мнение най-затрудняващата беше работата с датите и всичките проверки свързани с това. В началото започнах без класа [Date], наложи се да го създам, защото бързо разбрах колко необходима ще ми бъде консистентността която класа позволява. Друга неприятна част беше работата с командите, тяхното разпознаване и проверката за валидност на данните. В началото имах съвсем различна система за приемане и извеждане на информация, но по средата на работата се наложи изцяло да преработя този аспект на апликацията. Все още не съм напълно доволен от изпълнението (може още много да се желае), но за сега това работи достатъчно добре.

Разбира се имам много идеи за подобрение или развитие на апликацията:

* Да се изчисти и подобри входа и изхода на данни.
* Може да се създаде интерфейс извън конзолата (windows application).
* Могат да се добавят тонове функционалности като:
  + История на нощувките
  + Списък с пълната информация и документи на всеки гост във всяка стая
  + Възможност за предварително резервиране на стаи (със специален идентификатор)
  + Разделяне на стаите на видове спрямо лист от предлагани екстри
  + Калкулиране на цена на престоя в хотела и предлагане на пакети (all-inclusive, семейни, групови, сезони…)
  + (много други по-дребни идеи)

По мое мнение ако наистина бъдат реализирани поне някой подобрения от предложените и евентуално се обмисли сигурността на апликацията, може да се създаде нещо което поне на теория може да се използва в реалността.