Започнат на	събота, 17 април 2021, 10:34
Състояние	Завършен
Приключен на	събота, 17 април 2021, 12:32
Изминало време	1 час 58 мин.

Въпрос **1**

Отговорен

От максимално 12,00

В задачата използвайте наготово:

- класа `std::string` (от заглавен файл `<string>`).
- заглавен файл `<cstdint>` (за работа със `uint8_t` и `uint16_t`).
- заглавен файл `<stdexcept>` (за работа с класовете за изключения).
- заглавен файл `<cstddef>` (за работа със `std::size_t`).

В решението НЕ МОЖЕ да се използват контейнерите от STL (`std::vector`, `std::list` и т.н.)

За всеки от класовете НЕ МОЖЕ да се променя описаният интерфейс, но можете да добавите каквито прецените private функции или да разпишете допълнителни класове, които да ви помогнат в реализацията.

За всеки от класовете преценете дали трябва да реализирате функциите от rule of 3 (копиращ конструктор, копиращо присвояване, деструктор).

А) *(3 точки)* Дата ще представяме чрез три цели, положителни числа – ден, месец и година. За целите на задачата ще считаме, че една дата е валидна, ако:

- Месецът е стойност между 1 и 12.
- Денят не може да бъде 0.
- Ако месецът е януари(1), март(3), май(5), юли(7), август(8), октомври(10) или декември(12), денят е ≤ 31
- Ако месецът е април(4), юни(6), септември(9) или ноември(11), денят е ≤ 30
- Ако месецът е февруари(2) и годината е високосна, денят е ≤ 29
- Ако месецът е февруари(2) и годината НЕ Е високосна, денят е ≤ 28

Дали една година е високосна определяме по следния начин:

- Ако годината не е кратна на 4, значи НЕ Е високосна.
- В противен случай, ако НЕ Е кратна на 100, значи е високосна.
- Ако е кратна и на 4, и на 100, но НЕ Е кратна на 400, значи НЕ Е високосна.
- В противен случай е високосна година

Реализирайте клас `Date` представящ дата. Вътрешно класът да пази информацията като три променливи от следните типове:

- Дата – `std::uint8_t`
- Месец – `std::uint8_t`
- Година – `std::uint16_t`

Класът да има следния интерфейс:

- Класът НЕ ТРЯБВА да има конструктор по подразбиране.
- `Date(unsigned int day, unsigned int month, unsigned int year)`
  конструктор, който създава нов обект за дата с ден `day`, месец `month` и година `year`. Ако датата не е валидна, да се хвърля изключение от тип `std::invalid_argument`.
- `unsigned int day() const`
  връща деня като `unsigned int`.
- `unsigned int month() const`
  връща месеца като `unsigned int`.
- `unsigned int year() const`
  връща годината като `unsigned int`.
- `bool operator==(const Date& rhs) const`
  Връща истина, ако датата съхранена в текущия обект е същата като тази в `rhs`.
- `bool operator<(const Date& rhs) const`
  връща истина, ако датата съхранена в текущия обект е по-ранна от тази съхранена в обекта `rhs`.

Б) *(1 точка)* Реализирайте клас `Registration` (регистрация на превозно средство). Той трябва да има следните публични член-променливи:

- `id` – регистрационен номер, константа от тип `std::string`.
- `date` – дата на регистрация, константа от тип `Date`.

Класът да има следния интерфейс:

- Класът ДА НЯМА конструктор по подразбиране
- `Registration(const std::string& id, const Date& date)`
- `bool operator==(const Registration& rhs) const`
  Връща истина, ако номерът и датата на текущия обект съвпадат с тези на `rhs`.
- `bool operator<(const Registration& rhs) const`

Проверява дали текущата регистрация предхожда тази в `rhs`. Считаме, че една регистрация A предхожда друга регистрация B или (1) ако датата на A е преди тази на B, или (2) ако двете дати съвпадат, но регистрационният номер на A предхожда лексикографски този на B.

Упътване: класът `std::string` има оператори `<` и `==`, които можете да използвате наготово.

В) *(5 точки)* Реализирайте клас `RegistrationList` представящ списък от регистрациите на една кола.

Списъкът има капацитет, който се указва при неговото създаване. Капацитетът е от тип `std::size_t` и може да бъде произволно голям (помислете какво значи това за решението; как трябва да осигурите паметта). Той се указва само веднъж при създаването на списъка и после не може да се променя.

Регистрациите в списъка трябва да се пазят сортирани в нарастващ ред.

Обърнете внимание, че `Registration` обектите нямат default constructor. Преценете как да адресирате този проблем (например използвайте масив от указатели).

Класът да има следния интерфейс:

- `RegistrationList(std::size_t capacity)` създава списък, който може да съдържа най-много capacity на брой регистрации.
- Всички функции от rule of 3 (по желание: всички от rule of 5). Забележете, че `RegistrationList` не е като класа за гараж от домашното. `RegistrationList` притежава обектите съхранени в него и трябва да ги почиства в деструктора си. При копиране на списък, новото копие трябва да създаде за себе си нови обекти от тип `Registration`; то не трябва да сочи към тези на оригинала. Копието трябва да бъде със същия капацитет като оригинала. Операторът за присвояване да дава strong exception safety guarantee.
- `void insert(const std::string& id, const Date& date)`
  добавя регистрацията с номер `id` и дата `date`. Тъй като класът трябва да поддържа регистрациите сортирани в нарастващ ред, тази операция трябва да вмъкне новия запис на подходящо място в списъка. Ако операцията не успее (например няма повече място), да се хвърля изключение от тип `std::exception`. Операцията да дава strong exception guarantee.
- `const Registration& at(std::size_t index) const`
  достъп до елемента намиращ се на позиция `index`. Ако такъв няма, да се хвърля изключение `std::out_of_range`.
- `const Registration& operator[](std::size_t index) const`
  достъп до елемента намиращ се на позиция `index`. Функцията да не прави проверка за коректност дали `index` е валидна позиция. (В debug режим `assert`-вайте дали `index` е валидна позиция).
- `bool empty() const`
  Проверява дали списъка е празен (т.е. в него не е била добавена нито една регистрация).
- `std::size_t capacity() const`
  капацитет на списъка.
- `std::size_t size() const`
  брой регистрации добавени в списъка.

Г) *(3 точки)* Напишете програма, която:

1. Въвежда от потребителя число N и след това създава списък от регистрации с дължина N.
2. Въвежда от потребителя точно N регистрации и ги запазва в списъка. Ако потребителят въведе некоректна дата или пита да създаде два записа с един и същ номер, програмата ви трябва да може да улови хвърленото изключение. В такъв случай се извежда съобщение за грешка и потребителят може отново да опита да въведе данните за регистрацията.
3. Извежда на екрана всички съхранени регистрации в реда, в който те се съдържат в списъка.

k1.cpp

◄ Инструкция за контролното

Отиди на ...

Първо домашно ►