Zadanie drugie – po raz drugi

Żeby wyrównać oddawanie zadań z rozpoczynaniem nowych tematów na ćwiczeniach jeszcze raz zajmiemy się klasą `simpleDate`. Wprowadzimy do niej drobne innowacje z ostatniego wykładu tak, żeby wygodniej i rozsądniej zaimplementować metody klasy oraz uprościć testy klasy.
Usprawnienia będą następujące:
Użycie metod static,
Użycie składowej danych wspólnej dla wszystkich obiektów klasy (znów static)
Jawny (explicit) konstruktor
Zastąpienie metody print() operatorem <<
Zdefiniowanie operatora == (a może i !=)

Interfejs klasy (i nie tylko)

```
class simpleDate
{
public:
    explicit simpleDate(int day = 1, int month = 1, int year = 2020);
    int year() const;
    int month() const;
    int day() const;
    void next_day();
    void prev_day();
    static bool is_valid(int day, int month, int year);
    static int days_in_month(int month, int year);
    static bool is_leap_year(int year);
private:
    int dday, dmonth, dyear;
    const static std::vector<int> days_in_months;
};

bool operator==(const gi_date& left, const gi_date& right);
bool operator!=(const gi_date& left, const gi_date& right);
std::ostream& operator<<(std::ostream& os, const gi_date& right);
```

Operator <<

Operator << wyprzedza teorię. Jest jednak tak użyteczny w testach, że warto go mieć po ręką, nawet jeśli szczegóły jego realizacji nie są jasne.

Z ciekawostek mamy tu:

- wartość, która jest referencją na obiekt strumienia,
- argumenty przekazywane przez referencje,
- użycie manipulatorów strumieniowych.

```
std::ostream& operator<<(std::ostream& os, const gi_date& right)
{
    os << right.year() << '-' << setfill('0') << setw(2)
       << right.month() << '-' << right.day();
    return os;
}
```

Co zostało do zrobienia?

W trakcie ćwiczeń posunęliśmy prace nad klasą nieco do przodu: mamy zaimplementowane w pełni `is_leap_year()` i `days_in_month()` a częściowo `is_valid()`. Dwie ostatnie funkcje są przetestowane tylko symbolicznie.
Do zrobienia zostaje:

- Testy `days_in_month()`
- Dokończenie `is_valid()` (z testami),
- Zaimplementowanie operatorów `==` i `!=` poza klasą (podobnie jak zrobiliśmy to z operatorem `<<`)
- Zaimplementowanie przesuwania daty na następny/poprzedni dzień (`next_day()` i `prev_day()`) – z operatorem `==` testowanie będzie całkiem przyjemne

Dostarczanie rozwiązania

Rozwiązaniem zadania są pliki **`simpleDate.h`**, (definicja klasy) i **`simpleDate.cpp`** (definicje metod klasy).

Proszę załadować w `Moodle` tylko te dwa pliki (oczywiście na początku każdego pliku trzeba umieścić w komentarzu imię, nazwisko i nr albumu autora).

Rozwiązanie należy przekazać do:

23 listopada 2022 23:59