laboratorium: zadanie 5 termin: 26 marca – 5 kwietnia 2018 r.

KURS JĘZYKA C++

WYDARZENIA W CZASIE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

4 października 1582 papież Grzegorz XIII wprowadził kalendarz gregoriański. Zmiany wprowadzono na podstawie bulli papieskiej *Inter gravissimas*. Dokument nakazywał pominąć 10 dni z dotychczasowego kalendarza, dlatego też po 4 października 1582 roku (środa) nastąpił od razu 15 października (czwartek).

Poprzedni kalendarz, juliański, obowiązywał od 46 roku p.n.e. w imperium rzymskim, a później w krajach chrześcijańskich. Rok (zwykły) trwał według niego 365 dni, ale co 4 lata był dłuższy o dzień (rok przestępny). Nie nakładał się on jednak na rok słoneczny, co spowodowało, że po około 130 latach równonoc przesunęła się o jeden dzień.

Kalendarz juliański przewidywał dni przestępne, jednak tę regułę unowocześniono w kalendarzu gregoriańskim. Za lata przestępne uznano tylko te podzielne przez 4, z wyjątkiem podzielnych przez 100, ale lata podzielne przez 400 pozostawały przestępne. Kalendarz gregoriański niemal odpowiada kalendarzowi astronomicznemu, dzięki temu nie został zaburzony cykl pór roku w stosunku do dat kalendarzowych. Wprowadzenie kalendarza gregoriańskiego i nowej regulacji lat przestępnych i tak całkowicie nie rozwiązały problemu — opóźnienie kalendarza względem czasu astronomicznego wynosi jeden dzień na około 3000 lat.

Zadanie 1.

Zdefiniuj klasę `Data` do przechowywania daty w obowiązującym obecnie *kalendarzu gregoriańskim*. Przyjmij, że pierwszym dniem w tym kalendarzu jest *15 października 1582 roku*, czyli dzień, w którym papież Grzegorz XIII zarządził zmianę kalendarza (poprzednio obowiązywał *kalendarz juliański*). W kalendarzu tym ustalono nowy sposób rozstrzygania czy rok jest przestępny czy nie (czy luty ma 29 czy 28 dni):

> *Rok jest przestępny jeśli dzieli się przez 4 i nie dzieli się przez 100, za wyjątkiem lat podzielnych przez 400, które zawsze są przestępne.*

Zaprojektuj tą klasę tak, aby chronione pola z danymi (dzień, miesiąć, rok) były udostępniane przez odpowiednie gettery. Obiekt takiej klasy będzie więc można zainicjalizować podając konkretne wartości składowe daty. Dostarcz statycznej metody chronionej do sprawdzania czy rok jest przestępny i wykorzystaj ją w prywatnej metodzie badającej poprawność daty (wywoływanej w konstruktorze); w metodzie sprawdzającej poprawność daty skorzystaj ze statycznej tablicy dni w poszczególnych miesiącach:

```
static int Data::dniwmiesiacach[2][13] = {
    {0,31,28,31,30,31,30,31,31,30,31,30,31}, // lata zwykłe
    {0,31,29,31,30,31,30,31,31,30,31,30,31} // lata przestępne
};
```

Konstruktor bezargumentowy powinien pobierać bieżącą datę systemową. Klasa ta powinna też posiadać konstruktor kopiujący i przypisanie kopiujące (mogą być domyślne); zastamów się czy potrzeba implementować przenoszenie w tej klasie.

W definicji klasy umieść prywatną metodę instancyjną do obliczania ile dni upłunęło od pewnej ustalonej daty (na przykład od wirtualnego dnia *1 stycznia 0 roku*) do daty podanej jako parametr. Przy obliczaniu wyniku w tej metodzie nie używaj pętli tylko skorzystaj z zasady włączeń i wyłączeń; wykorzystaj w tej metodzie statyczną tablicę z ilością dni, które upłynęły od początku roku do końca danego miesiąca:

```
static int Data::dniodpoczroku[2][13] = {
    {0,31,59,90,120,151,181,212,243,273,304,334,365}, // lata zwykłe
    {0,31,60,91,121,152,182,213,244,274,305,335,366} // lata przestępne
};
```

Następnie zdefiniuj składowy wirtualny operator odejmowania `-` do obliczania jaka jest różnica (wyrażona w dniach) pomiędzy dwiema datami — parametrem tego operatora niech będzie referencja do stałej daty. Tutaj do obliczeń wykorzystaj metodę, o której mowa była przed chwilą. Uzupełnij tą klasę o operatory inkrementacji `++` i dekrementacji `--` (przesunięcie o jeden dzień do przodu i do tyłu) oraz składowe operatory dodawania `+=` i odejmowania `-=` zadanej liczby dni od bieżącej daty.

Zadanie 2.

Zdefiniuj klasę `DataGodz` do przechowywania daty i godziny (punkt czasowy), dziedzicząc po klasie `Data`. Klasa ta ma być finalna.

Zaprojektuj tą klasę tak, aby chronione pola z danymi (godziny, minuty, sekundy) były udostępniane przez odpowiednie gettery. Zdefiniuj w tej klasie konstruktor do inicjalizacji daty i godziny konkretnymi wartościami (użyj parametrów domyślnych) i konstruktor bezargumentowy inicjalizujący obiekt bieżącą datą i godziną systemową. Klasa ta powinna być kopiowalna.

Następnie zdefiniuj globalny operator odejmowania `-` do obliczania jaka jest różnica (wyrażona w sekundach) pomiędzy dwoma punktami czasowymi — parametrem tego operatora niech będą referencje do stałych punktów czasowych. Uzupełnij tą klasę o operatory inkrementacji `++` i dekrementacji `--` (przesunięcie o jedną sekundę do przodu i do tyłu) oraz składowe operatory dodawania `+=` i odejmowania `-=` zadanej liczby sekund od bieżącego punktu czasowego. Do kompletu zdefinuj dwa operatory relacyjne `<` i `==` porównujące punkty czasowe.

Zadanie 3.

Zdefiniuj klasę `Wydarzenie` do przechowywania punktu czasowego `DataGodzina` i skojarzonego z nim wydarzenia opisanego za pomocą łańcucha znakowego `string`. Zastosuj kompozycję zamiast dziedziczenia.

Zadanie 4.

Napisz program testujący zachowanie zdefiniowanych przez Ciebie klas. W skład testu niech wchodzi posortowanie kolekcji wydarzeń `vector<Wydarzenie>` (zdefiniuj operator porównujący wydarzenia pod względem chronologii i użyj go jako argumentu w standardowej funkcji sortującej); wypisz estetyczne wydarzenia po posortowaniu.

Elementy w programie, na które należy zwracać uwagę.

- Podział programu na pliki nagłówkowe i źródłowe.
- Składowe statyczne w klasie daty.
- Odejmowanie dat.
- Odejmowanie punktów czasowych.
- Operatory dodające i odejmujące w datach i potem w punktach czsowych.
- porównywanie punktów czasowych.