# Mail log

| | |
|---|---|
| **Termín odevzdání:** | **21.04.2019 23:59:59** |
| **Pozdní odevzdání s penalizací:** | **30.06.2019 23:59:59** (Penále za pozdní odevzdání: 100.0000 %) |
| **Hodnocení:** | **5.5000** |
| **Max. hodnocení:** | **5.0000** (bez bonusů) |
| **Odevzdaná řešení:** | 1 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání) |
| **Nápovědy:** | 0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu) |

Úkolem je vytvořit třídu `CMailLog`, která bude zpracovávat logy z poštovního subsystému.

Při analýze e-mailového provozu je potřeba získat informace o tom, kteří uživatelé, kdy a komu poslali e-mail. K tomu je potřeba analyzovat logu poštovního serveru. Třída `CMailLog` toto bude umožňovat.

Vstupem pro třídu je log poštovního serveru. Poštovní server má logy ve formě:

```
month day year hour:minute:sec relay_name mailID: message
```

tedy například:

```
Mar 29 2013 14:55:31.456 relay.fit.cvut.cz KhdfEjkl247D: from=PR-department@fit.cvut.cz
Mar 29 2013 14:50:23.233 relay.fit.cvut.cz ADFger72343D: mail undeliverable
Mar 29 2013 14:58:32.563 relay.fit.cvut.cz KhdfEjkl247D: subject=Meeting this afternoon
Mar 29 2013 15:04:18.345 relay.fit.cvut.cz KhdfEjkl247D: to=CEO@fit.cvut.cz
```

Význam polí je zřejmý: měsíc (anglická zkratka, 3 písmena, první velké), den měsíce, rok, hodina, minuta a sekunda (s přesností na milisekundy), jméno serveru (DNS jméno), identifikátor e-mailu (textový řetězec, písmena a čísla) a zpráva. Poštovní server loguje zprávu při každé zajímavé události (např. přijetí e-mailu od klienta, doručení e-mailu do schránky, ...). Je běžné, že jeden zaslaný e-mail vygeneruje několik řádek logu, řádky logu, které se týkají zpracování jednoho e-mailu mají stejný identifikátor.

Cílem při zpracování logu je propojit odesilatele, subjekt a příjemce a zaznamenat si pro každý takový e-mail čas doručení. Proto nás budou zajímat zprávy začínající `from=` a k nim odpovídající `subject=` a `to=`. Zprávy, které nezačínají ani `from=`, `subject=` ani `to=` přeskakujte. Pokud v logu není k dispozici `subject=`, pak ve výpisech vyplňte subjekt jako prázdný řetězec.

Z ukázky je vidět, že logování jde téměř chronologicky, ale občas je některá logovací hláška opožděná. Můžete se spolehnout, že opožděně jsou vypsané pouze hlášení typu `to=` (tedy pro každý e-mail je dodržená sekvence: nejprve `from=`, potom `subject=` a nakonec jedno či více `to=`). Dále je vidět, že mohou být promíchaná hlášení, která se týkají paralelně zpracovávaných e-mailů. Konečně, je vidět, že jeden e-mail může vygenerovat více hlášení typu `to=`, například pokud byl doručen více příjemcům. Toto budeme považovat za dva e-maily, navíc každý doručený v jiný čas.

Vytvářená třída bude umět takový log zpracovávat, ukládat a generovat z něj reporty. Musí se začlenit do rozhraní popsaného níže. Část rozhraní je implementovaná v testovacím prostředí (třídy `CTimeStamp` a `CMail`), proto jsou tyto třídy v bloku podmíněného překladu. Tyto třídy tedy nemusíte celé implementovat, nicméně pro účely testování budete muset dodělat jejich alespoň zjednodušenou implementaci ("mocking"). Aby šel výsledný program zkompilovat, je NUTNÉ tyto dvě třídy a jejich implementaci ponechat uvnitř bloku podmíněného překladu. Plně pak musíte implementovat třídu `CMailLog` a tu naopak ponechat mimo blok podmíněného překladu.

```
class CTimeStamp
{
  public:
    CTimeStamp ( int       year,
                 int       month,
                 int       day,
                 int       hour,
                 int       minute,
                 double    sec );

    int Compare ( const CTimeStamp & x ) const;
    friend ostream & operator << ( ostream     & os,
                                   const CTimeStamp & x );
  private:
    ...
};

class CMail
{
  public:
    CMail  ( const CTimeStamp & timeStamp,
```

```
            const string     & from,
            const string     & to,
            const string     & subject );
    int CompareByTime ( const CTimeStamp & x ) const
    int CompareByTime ( const CMail      & x ) const
    const string & From ( void ) const;
    const string & To   ( void ) const;
    const string & Subject ( void ) const;
    const CTimeStamp & TimeStamp ( void ) const;
    friend ostream & operator << ( ostream     & os,
                                   const CMail & x );
  private:
    ...
};
#endif /* __PROGTEST__ */

class CMailLog
{
  public:
    // default constructor
    // destructor
    int            ParseLog          ( istream & in );
    list<CMail>    ListMail          ( const CTimeStamp & from,
                                       const CTimeStamp & to ) const;
    set<string>    ActiveUsers       ( const CTimeStamp & from,
                                       const CTimeStamp & to ) const;
  private:
    ...
};
```

## CTimeStamp

Tato třída implementuje časové razítko. Má následující metody:

- konstruktor, který vyplní vzniklou instanci,
- přetížený operátor <<, který zobrazí časové razítko v ISO formátu (YYYY-MM-DD HH24:MI:SS.UUU). Pro vlastní testování tato funkce není nezbytná, hodí se ale pro ladění,
- Metodu pro porovnání `Compare`. Pro volání `a . Compare (b)` bude vráceno kladné číslo pro `a > b`, 0 pro `a == b` a záporné číslo pro `a < b`.

## CMail

Tato třída reprezentuje jeden doručený e-mail. Má následující metody:

- konstruktor, který vyplní vzniklou instanci,
- metody pro čtení členských proměnných (getter) - `From`, `Subject`, `To` a `TimeStamp`,
- metody pro porovnání časových razítek v záznamech e-mailu, návratová hodnota odpovídá výsledku porovnání, který vrací `CTimeStamp::Compare`,
- přetížený operátor pro výstup, opět pouze pro účely ladění.

## CMailLog

Tato třída reprezentuje zpracovaný log poštovního serveru. Má následující metody:

- implicitní konstruktor, který vytvoří prázdnou instanci,
- destruktor, který uvolní alokované prostředky,
- metodu `ParseLog`, která zpracuje předaný log. Log je přístupný v podobě vstupního streamu. Metoda zpracovává vstup a ukládá informace o doručených e-mailech. Pokud je nějaká řádka na vstupu neplatná, metoda ji přeskočí a pokračuje ve zpracování další řádky logu. Návratovou hodnotou metody je počet doručených e-mailů, které byly při zpracování logu nalezené (tedy ne počet řádek logu),
- metoda `ListMail` vrátí seznam nalezených e-mailů doručených v zadaném období (od-do, včetně). Vrácený seznam bude seřazený podle času doručení vzestupně, pokud by dvojice mailů měla stejný čas doručení, zachová se pořadí z logu,
- metoda `ActiveUsers` uvažuje všechny e-maily, které byly v zadaném období (od-do, včetně) doručené. Pro tyto e-maily vrátí množinu e-mailových adres uživatelů, kteří byli odesilateli či adresáty.

Odevzdávejte soubor, který obsahuje implementovanou třídu `CMailLog`. Třída musí splňovat veřejné rozhraní podle ukázky - pokud Vámi odevzdané řešení nebude obsahovat popsané rozhraní, dojde k chybě při kompilaci. Do třídy si ale můžete doplnit další metody (veřejné nebo i privátní) a členské proměnné. Odevzdávaný soubor musí obsahovat jak deklaraci třídy (popis rozhraní) tak i definice metod, konstruktoru a destruktoru. Je jedno, zda jsou metody implementované inline nebo odděleně. Odevzdávaný soubor nesmí obsahovat vkládání hlavičkových souborů a funkci `main`. Funkce `main`, vkládání hlavičkových souborů a třídy `CTimeStamp` a `CMail` mohou zůstat, ale pouze obalené direktivami podmíněného překladu jako v ukázce výše.

Při řešení úlohy využijte STL. Využijte STL tak, abyste byli schopni log zpracovávat a vyhledávat v něm rychleji než lineárně.

Váš kód bude umístěn ve jmenném prostoru, aby nekolidoval s ostatními identifikátory v testovacím prostředí. Hotové třídy `CTimeStamp` a `CMail` jsou umístěné v nadřazeném jmenném prostoru, aby byly využitelné vaším kódem (z Vašeho jmenného prosotru) i kódem referenčním (z jmenného prostoru reference). V přiloženém testovacím kódu je to naznačeno direktovou `namespace MysteriousNamespace` (skutečné jméno se bude jiné, náhodně generované). Do Vašeho jmenného prostoru patří implementace třídy `CMailLog` a případných podpůrných tříd a funkcí. Naopak, do Vašeho jmenného prostoru nepatří kód rozšiřující chování tříd `CTimeStamp` a `CMail` (ten by patřil do nadřazeného jmenného prostoru). Nesnažte se ze svého jmenného prostoru rozšiřovat rozhraní těchto tříd, spoléhejte se pouze na již existující rozhraní.

Řešení této úlohy nelze použít pro code review.

| | | |
|---|---|---|
| **Vzorová data:** | | Download |
| **Odevzdat:** | Choose File No file chosen | Odevzdat |

**Referenční řešení**

| | | |
|---|---|---|
| **1** | **14.04.2019 20:00:11** | Download |
| **Stav odevzdání:** | Ohodnoceno | |
| **Hodnocení:** | 5.5000 | |

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test s parametry podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 5.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.772 s (limit: 5.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test rychlosti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 1.624 s (limit: 5.000 s)
    - Úspěch v nepovinném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 1.00 * ( 5.00 + 0.50 ) = 5.50

**SW metriky:**

| | Celkem | Průměr | Maximum | Jméno funkce |
|---|---|---|---|---|
| Funkce: | **24** | -- | -- | -- |
| Řádek kódu: | **218** | 9.08 ± 11.44 | 52 | `main` |
| Cyklomatická složitost: | **55** | 2.29 ± 2.67 | 13 | `CTimeStamp::Compare` |