Zpracování logu

Termín odevzdání:	**21.04.2013 23:59:59**
Pozdní odevzdání s penalizací:	**12.05.2013 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**4.4000**
Max. hodnocení:	**4.0000** (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 odeslali 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 14:59:23.233 relay.fit.cvut.cz ADFger72343D: mail undeliverable
Mar 29 2013 14:58:32.563 relay.fit.cvut.cz KhdfEjkl247D: to=HR-department@fit.cvut.cz
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, odeslání 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 spárovat odesilatele a příjemce a zaznamenat si pro každou takovou dvojici čas odeslání. Proto nás budou zajímat zprávy začínající `from=` a k nim odpovídající `to=`. Pro každou takto nalezenou dvojici si zapamatujeme odesílatele, příjemce a čas odeslání e-mailu. Zprávy, které nezačínají ani `from=` ani `to=` přeskakujte.

Z ukázky je vidět, že logování jde přibližně chronologicky, ale záznamy mohou být "trochu" proházené v čase. Navíc 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 odeslán více příjemcům. Toto budeme považovat za dva e-maily, navíc každý odeslaný 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.

```
#ifndef __PROGTEST__
#include <iostream>
#include <iomanip>
#include <sstream>
#include <vector>
#include <map>
#include <set>
#include <list>
#include <algorithm>
using namespace std;

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 );
    int CompareTimeStamp ( const CTimeStamp & x ) const
    int CompareByTimeStamp ( const CMail      & x ) const
    const string & From ( void ) const;
    const string & To   ( 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 razítkov 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 odeslaný e-mail. Má následující metody:

- konstruktor, který vyplní vzniklou instanci,
- metody pro čtení členských proměnných (getter) - `From`, `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 odeslaných e-mailech. Pokud je nějaká řádka na vstupu neplatná, metoda ji přeskočí a pokračuje ve zpracování další řádky logu. Návratovou hodnotou metody je počet odeslaných e-mailů, které byly při zpracování logu nalezené (tedy ne počet řádek logu),
- metoda `ListMail` vrátí seznam nalezených odeslaných e-mailů v zadaném období (od-do, včetně). Vrácený seznam bude seřazený podle času odeslání vzestupně,
- metoda `ActiveUsers` uvažuje všechny e-maily, které byly v zadaném období (od-do, včetně) odeslané. 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. Nepoužívejte ale syntaxi C++0x (`auto`, uzávěry, ...), kompilátor tuto syntaxi ještě nezvládá. Využijte STL tak, abyste byli schopni v seznamu zpracovaných e-mailů vyhledávat rychleji než lineárně.

Ukázka práce třídy:

```
void showMail ( const list<CMail> & l )
 {
   for ( list<CMail>::const_iterator it = l . begin (); it != l . end (); ++it )
    cout << *it;
 }
void showUsers ( const set<string> & s )
 {
   for ( set<string>::const_iterator it = s . begin (); it != s . end (); ++it )
    cout << *it << endl;
 }
 ...
CMailLog    m;
int         cnt;
list<CMail> mailList;
set<string> users;
istringstream iss;

iss . clear ();
iss . str ( string (
   "Mar 29 2013 12:35:32.233 relay.fit.cvut.cz ADFger72343D: from=user1@fit.cvut.cz\n"
   "Mar 29 2013 12:37:16.234 relay.fit.cvut.cz JlMSRW4232Df: from=person3@fit.cvut.cz\n"
   "Mar 29 2013 12:38:45.043 relay.fit.cvut.cz Kbced342sdgA: from=office13@fit.cvut.cz\n"
   "Mar 29 2013 12:55:13.023 relay.fit.cvut.cz JlMSRW4232Df: to=user76@fit.cvut.cz\n"
   "Mar 29 2013 13:48:12.654 relay.fit.cvut.cz Kbced342sdgA: to=boss13@fit.cvut.cz\n"
   "Mar 29 2013 14:55:31.456 relay.fit.cvut.cz KhdfEjkl247D: from=PR-department@fit.cvut.cz\n"
   "Mar 29 2013 14:58:32.563 relay.fit.cvut.cz KhdfEjkl247D: to=HR-department@fit.cvut.cz\n"
   "Mar 29 2013 15:25:23.233 relay.fit.cvut.cz ADFger72343D: mail undeliverable\n"
   "Mar 29 2013 15:02:34.232 relay.fit.cvut.cz KhdfEjkl247D: to=CIO@fit.cvut.cz\n"
   "Mar 29 2013 15:04:18.345 relay.fit.cvut.cz KhdfEjkl247D: to=CEO@fit.cvut.cz\n"
   ) );
cnt  = m . ParseLog ( iss );
// cnt = 5
mailList = m . ListMail ( CTimeStamp ( 2013, 3, 28, 0, 0, 0 ),
                          CTimeStamp ( 2013, 3, 29, 23, 59, 59 ) );
showMail ( mailList );
/*
---8<---8<---8<---8<---8<---8<---
2013-03-29 12:55:13.023 person3@fit.cvut.cz -> user76@fit.cvut.cz
2013-03-29 13:48:12.654 office13@fit.cvut.cz -> boss13@fit.cvut.cz
2013-03-29 14:58:32.563 PR-department@fit.cvut.cz -> HR-department@fit.cvut.cz
2013-03-29 15:02:34.232 PR-department@fit.cvut.cz -> CIO@fit.cvut.cz
2013-03-29 15:04:18.345 PR-department@fit.cvut.cz -> CEO@fit.cvut.cz
---8<---8<---8<---8<---8<---8<---
*/
mailList = m . ListMail ( CTimeStamp ( 2013, 3, 28, 0, 0, 0 ),
                          CTimeStamp ( 2013, 3, 29, 14, 59, 59 ) );
```

```
showMail ( mailList );
/*
---8<---8<---8<---8<---8<---8<---
2013-03-29 12:55:13.023 person3@fit.cvut.cz -> user76@fit.cvut.cz
2013-03-29 13:48:12.654 office13@fit.cvut.cz -> boss13@fit.cvut.cz
2013-03-29 14:58:32.563 PR-department@fit.cvut.cz -> HR-department@fit.cvut.cz
---8<---8<---8<---8<---8<---8<---
*/
mailList = m . ListMail ( CTimeStamp ( 2013, 3, 30, 0, 0, 0 ),
                          CTimeStamp ( 2013, 3, 30, 23, 59, 59 ) );
showMail ( mailList );
/*
---8<---8<---8<---8<---8<---8<---
---8<---8<---8<---8<---8<---8<---
*/
users = m . ActiveUsers ( CTimeStamp ( 2013, 3, 28, 0, 0, 0 ),
                          CTimeStamp ( 2013, 3, 29, 23, 59, 59 ) );
showUsers ( users );
/*
---8<---8<---8<---8<---8<---8<---
CEO@fit.cvut.cz
CIO@fit.cvut.cz
HR-department@fit.cvut.cz
PR-department@fit.cvut.cz
boss13@fit.cvut.cz
office13@fit.cvut.cz
person3@fit.cvut.cz
user76@fit.cvut.cz
---8<---8<---8<---8<---8<---8<---
*/
users = m . ActiveUsers ( CTimeStamp ( 2013, 3, 28, 0, 0, 0 ),
                          CTimeStamp ( 2013, 3, 29, 13, 59, 59 ) );
showUsers ( users );
/*
---8<---8<---8<---8<---8<---8<---
boss13@fit.cvut.cz
office13@fit.cvut.cz
person3@fit.cvut.cz
user76@fit.cvut.cz
---8<---8<---8<---8<---8<---8<---
*/
```

Referenční řešení

1	**07.04.2013 15:52:04**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	4.4000	

- **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: 6.000 s)
    - Využití paměti: 12612 KiB (limit: 28009 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 1.843 s (limit: 6.000 s)
    - Využití paměti: 23424 KiB (limit: 28009 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 1.00 * (4.00 + 0.40) = 4.40

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**23**	--	--	--
Řádek kódu:	**278**	**12.09 ± 14.55**	**68**	`ParseLog(istream &)`
Cyklomatická složitost:	**84**	**3.65 ± 6.26**	**25**	`monthStrToInt(const string &)`