Objektorientierte Programmierung mit C++ (WS 2001/2002)

Abgabetermin: 6. Dezember 2001

Beispiellösung

6 E-Mails auf indisch (10+10 Punkte)

Shrandra Pradesh, der Inder in ihrem Team bei der aufstrebenden Softwareschmiede BadPointer.Com, mußte leider wegen eines Krankheitsfalls seines Lieblingselefanten auf Firmenkosten für 14 Tage zurück in sein Heimatland reisen. „Sie springen für Shrandra wegen der Fertigstellung von Emilia ein!“, sagte ihr Chef kurz in der morgendlichen Besprechung. „Das Programm muß Donnerstag nächste Woche an die Leute von SaiTutoren.com ausgeliefert werden! Unser Ruf steht auf dem Spiel.“ (Schon wieder!). Sie wissen, was das heißt: Einarbeiten in den kaum dokumentierten Code von Shrandra, dessen spärlichen Kommentare meist auf Hindu sind ... nur ein paar englische und deutsche Erklärungen über das Projekt „Emilia“ hat er in den C++ Code eingestreut ... das kann ja Eiter werden. Egal - nix wie ran!
Sie finden alle von Shrandra bisher erstellten Programme auf dem *Web*. Worum geht es bei dem Projekt? Emilia ist ein E-Mailreader, der Spams (unerwünschte Werbemails) filtern soll. Shrandra hat schon ganz gut vorgearbeitet. Der Aufruf ist: emilia verzeichnis. Dadurch werden alle Header von E-Maildateien auf dem angegeben Verzeichnis in den Reader geladen. Die Annahme ist, das genau eine Mail in einer Datei abgespeichert ist. Ein Objekt der Klasse email hält eine einzelne Mail und in einem Objekt der Klasse mailbox finden sich alle Mails eines Verzeichnisses wieder. Ihr Chef gibt ihnen ziemlich freie Hand beim OO-Design (oder sinnvollen Änderungen des bestehenden) - nur will er, dass bis nächste Woche die folgende Funktionalität in das Programm integriert wird:

- Der komplette Header einer bestimmten E-Mail sollte am Schirm angezeigt werden können. Sie implementieren deshalb zum Aufwärmen `void mailbox::showheader(int emailnum)` (1 Punkt)

- SaiTutoren.com will eine Funktion, mit der bei Angabe der Nummer einer E-Mail nur mehr E-Mails mit identischen Absender wie die ausgewählte E-Mail angezeigt werden. (1 Punkt) `void mailbox::showsamesender(int email)`

- Mit der Option '-d' soll als Zusatzinformation neben Name und Subject auch das Empfangsdatum (Text hinter 'Date: ' im Header) der Email in der Infozeile der E-Mail angezeigt werden. (3 Punkte)
- Spamfilter: (5 Punkte)
  Bei Aufruf der Option '-s' sollen alle von ihnen als Spam identifizierten E-Mails nicht mehr angezeigt werden. Bei der Option '-S' werden jedoch nur die Spams angezeigt! Ohne die S-Optionen werden alle E-Mails angezeigt. Eine E-Mail ist mit hoher Wahrscheinlichkeit eine Spam, wenn
  - die Subject-Zeile aus mehreren Wörtern besteht, die aber alle GROß geschrieben sind („ACHTUNG: GRATIS AUDI A4 FUER DICH“).
  - im Subject lästige Schlüsselwörter wie z.B. „Sex“, „$$$$“, „Money“ etc. vorkommen.
  - (...) Sie können diese Kriterienliste beliebig erweitern!
- Das Programm sollte bei Aufruf mit der Option '-n' nur mehr die Header der 20 neuesten E-Mails (zeitlich) absteigend sortiert auflisten. (Zeile „'Date:“ in der Mail). (6 Bonuspunkte = optional)
- Bei der Option '-b' soll auch im eigentlichen Mailtext nach den Spamschlüsselwörtern gesucht werden. (4 Bonuspunkte = optional)

Lösung:

makefile

```
CC=g++
obj=main.o email.o mailbox.o date.o

emilia:   $(obj)
        g++  -o emilia $(obj)

main.o: mailbox.h
email.o: email.h date.h
mailbox.o: email.h mailbox.h
date.o: date.h
```

main.cc

```
// (c) Shrandra Pradesh sai uni ulm 2001
#include "mailbox.h"
#include <iostream>
#include <strings.h>

bool opt_s = false;
bool opt_S = false;
bool opt_b = false;
bool opt_d = false;
bool opt_n = false;
```

```
int main(int argc, char **argv)
{

        char *command = argv[0];
        --argc; ++argv;

        while (argc >= 1) {
            if (!strcmp(*argv, "-s"))           opt_s = true;
            else if (!strcmp(*argv, "-S"))      opt_S = true;
            else if (!strcmp(*argv, "-b"))      opt_b = true;
            else if (!strcmp(*argv, "-d"))      opt_d = true;
            else if (!strcmp(*argv, "-n"))      opt_n = true;
            else if (!strcmp(*argv, "--"))      { --argc; ++argv; break; }
            else
                break;

            --argc; ++argv;
        }

        if (opt_s && opt_S) {
                cerr << "options -s and -S cannot be set simultaneously" << endl;
                exit(1);
        }

        if (argc < 1) {
                cerr <<"usage: "<< command <<" {-s|-S|-b|-d|-n} [--] maildir"<<endl;
                exit(1);
        }
        mailbox mb(*argv);              // Mailboxverzeichnis einlesen

        if (!mb) {                      // Einlesen hat nicht geklappt
                cout <<"Mailbox error: "<< mb.errmesg()<<endl;
                return 1;
        }
        mb.dialog();
}
```

email.cc

```
// (c) Shrandra Pradesh sai uni ulm 2001
// die E-Mailklasse
#include <vector>
#include <string>
#include <fstream>
#include <strings.h>
#include "email.h"

email::email()
{
```

```
        validemail = false;
}

// Einlesen des E-Mail-Headers aus einer Datei
email::email(const char *file)
{
        ifstream in(file);      // Eingabestrom oeffnen

        validemail = false;
        if (!in) {
                cout << "can't open "<<file<<endl;
                return;
        }
        filename = file;        // Dateinamen merken
        for (int i=0; i<numkeys; i++)
                switch (i) {
                        case 0:
                                keywords[i].key = "From: ";
                                keywords[i].var = &from;
                                break;
                        case 1:
                                keywords[i].key = "Return-Path: ";
                                keywords[i].var = &replyto;
                                break;
                        case 2:
                                keywords[i].key = "Subject: ";
                                keywords[i].var = &subject;
                                break;
                        case 3:
                                keywords[i].key = "To: ";
                                keywords[i].var = &receiver;
                                break;
                        case 4:
                                keywords[i].key = "Date: ";
                                keywords[i].var = &date;
                                break;
                        default:
                                ;
                }
        bool isheader = true;

        while (!in.eof() && isheader) {
                string line;

                getline(in, line);      // komplette Zeile einlesen
                if (isheader && line.size()==0) {
                        isheader = false; // 1.Leerzeile? Header-Ende!
                        if (!from.empty() && !subject.empty())
                                validemail = true;      // echte E-Mail
```

```
            } else if (isheader) {  // Suche nach Schluesselworten
                                    // eine Headerzeile abspeichern
                        header.resize(header.size()+1, line);
                                    // Suche nach Keywors am Zeilenanfang
                        for (int i=0; i<numkeys; i++) {
                                char * hlp = keywords[i].key;
                                if (line.find(hlp, 0)==0)
                                        *keywords[i].var =
                                                line.substr(strlen(hlp));
                        }
            }
        }

        if (date.size() > 0) {
            int pos = date.find(",");
            if (pos != string::npos)
                date = date.substr(pos+1);
            try {
                d_date = Date(date.c_str());
            }
            catch (const char *s) {
                cerr << "Exception: " << s << endl;
                exit(1);
            }
        }

        in.close();
}

bool isAllUpper(string line)
{
    char *upper = new char[line.size()+1];
    strcpy(upper, line.c_str());
    char *s = upper;
    while (*s != 0) {
        if (*s >= 'a' && *s <= 'z')
            *s = *s - 'a' + 'A';
        s++;
    }
    int blocks = 0;
    s = upper;
    while (*s != 0) {
        if (*s >= 'A' && *s <= 'Z' && (blocks == 0 || blocks == 2))
            blocks++;
        if (!(*s >= 'A' && *s <= 'Z') && blocks == 1)
            blocks++;
        s++;
    }
    string lineUpper(upper);
```

```
    delete[] upper;

    // alles ist GROSS geschrieben UND es gibt mindestens 2 Worte
    if (line == lineUpper && blocks >= 3)
        return true;

    return false;
}

bool containsKey(string line)
{
    char *upper = new char[line.size()+1];
    strcpy(upper, line.c_str());
    char *s = upper;
    while (*s != 0) {
        if (*s >= 'a' && *s <= 'z')
            *s = *s - 'a' + 'A';
        s++;
    }
    string lineUpper(upper);
    delete[] upper;

    // ... oder es kommt eines der folgenden Schluesselworte
    // (in bel. Schreibweise -- d.h. GROSS oder klein) vor
    if (lineUpper.find("SEX") != string::npos)
        return true;
    if (lineUpper.find("$$$$") != string::npos)
        return true;
    if (lineUpper.find("MONEY") != string::npos)
        return true;
    if (lineUpper.find("OFFER") != string::npos)
        return true;

    return false;
}

extern bool opt_b;

bool email::isspam() {
    if (isAllUpper(subject) || containsKey(subject))
        return true;

    bool spam = false;

    if (opt_b) {

        bool bodyRead = body.size() > 0;

        getmail();
```

```
        for (int i = 0; i < body.size(); i++)
            if (containsKey(body[i])) {
                spam = true;
                break;
            }

        if (!bodyRead)
            body.resize(0);

    }

    return spam;
}

// Einlesen des eigentlichen E-Mailtextes einer E-Mail

void email::getmail()
{
        if (!validemail || body.size()>0)
                return;
        ifstream in(filename.c_str());  // Eingabestrom oeffnen

        if (!in) {
                cout << "can't open "<<filename.c_str()<<endl;
                return;
        }
        bool isheader = true;
        while (!in.eof()) {
                string line;
                getline(in, line);      // komplette Zeile einlesen
                if (isheader) {
                        if (line.size()==0)     // Ende des Headers
                                isheader = false;
                        continue;
                }
                body.resize(body.size()+1, line); // Zeile speichern
        }
}

void email::printbody(ostream& out) const
{
        if (!body.size())
                return ;

        for (int i=0; i< body.size(); i++) {
                out << body[i]<<endl;
                if (!((i+1)%23)) {
                        string dum;
```

```
                        getline(cin, dum);
                        system("clear");
                }

        }
}

void email::printaddress(ostream& out) const
{
        for (int i=0; i<numkeys; i++)
                cout << keywords[i].key <<*(keywords[i].var)<<endl;
}

void email::printheader(ostream& out) const
{
        if (!header.size())
                return ;

        for (int i=0; i< header.size(); i++) {
                out << header[i]<<endl;
                if (!((i+1)%23)) {
                        string dum;
                        getline(cin, dum);
                        system("clear");
                }

        }

}

string email::getemil()
{
    string sender = getfrom();
    int pos1 = sender.find("<");
    int pos2 = sender.find(">");
    if (pos1!=string::npos && pos2!=string::npos)
        sender = sender.substr(pos1+1, pos2-pos1-1);
    return sender;
}

// Ausgabe der kompletten E-Mail
ostream& operator<<(ostream& out, const email &e)
{
        if (!e.header.size())
                return out;

        e.printheader(out);
        out<<endl;              // Leerzeile
        e.printbody(out);
```

```
        return out;
}
```

email.h

```
// die E-Mailklasse
#ifndef _email                          // Mehrfach-Include abstellen
#define _email

#include <fstream>
#include <vector>
#include <string>
#include "date.h"

const int numkeys = 5;
struct keys {
        char *key;                      // Text Zeilenanfang
        string *var;                    // Inhalt der Zeile
};

class email {
        vector<string> header;          // complete header
        vector<string> body;            // complete body
        string from;                    // sender
        string replyto;                 // reply to
        string receiver;                // receiver of e-mail
        string filename;                // file read from
        string subject;                 // subject line
        string date;
        Date d_date;
        struct keys keywords[numkeys];
        bool validemail;                // was file a valid email file?
public:
        email();
        email(const char *file);        // Einlesen des E-Mailheaders

        bool isspam();

        void printheader(ostream&) const; // Ausgabe Header
        void printbody(ostream&) const; // Ausgabe Body
        void printaddress(ostream&) const;// Ausgabe Adressierung
        void getmail();                 // Einlesen des eigentlichen Textes
        bool operator!() { return !validemail; }
        string& getfilename() { return filename; }
        string& getfrom() { return from; }
        string& getsubject() { return subject; }
        Date& getdate() { return d_date; }

        string getemil();
                                        // Ausgabe komplette E-Mail
```

```
        friend ostream& operator<<(ostream&, const email &);
};
#endif
```

mailbox.cc

```
// (c) Shrandra Pradesh sai uni ulm 2001
#include "mailbox.h"
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>

bool mailbox::operator!()                       // hatte das Einlesen geklappt?
{
        return validbox==false;
}

extern bool opt_s, opt_S, opt_n;

// Erzeugen einer Mailbox: einlesen aller Mails auf dem betreffenden Verzeichnis
mailbox::mailbox(char *path)
{
        validbox = false;
        DIR *dp = opendir(path);
        if (!dp) {
                errmessage = new char(strlen(path)+1+30);
                sprintf(errmessage, "could'nt open: >%s<", path);
                return ;
        }
        struct dirent *ep;
        int mails = 0;
        while (ep = readdir(dp)) {
                if (!strcmp(".", ep->d_name) || !strcmp("..", ep->d_name))
                        continue;

                validbox = true;
                char *file = new char[strlen(ep->d_name)+strlen(path)+2];
                sprintf(file, "%s/%s", path, ep->d_name);
                cerr <<"reading "<< ++mails<<"\r";
                cerr.flush();
                email newmail(file);
                delete [] file;

                if (!newmail || (opt_s && newmail.isspam()) || (opt_S && !newmail.isspam()))
                        ;
                else {
                    if (!opt_n)
                        emails.resize(emails.size()+1, newmail);
                    else {
                        int i;
```

```
                        for (i = 0; i < emails.size(); i++)
                            if (emails[i].getdate() <= newmail.getdate())
                                break;
                        emails.insert(&emails[i], newmail);
                        if (emails.size() > 20)
                            emails.resize(20);
                    }
                }
        }
        closedir(dp);
}

extern bool opt_d;

// Anzeige einer E-Mailzeile
static void statusline(string sender, string subject, Date d, int num)
{
        int pos = sender.find("<");
        if (pos!=string::npos)
                sender = sender.substr(0, pos);

        sender.resize(20, ' ');
        char hlp[10];
        sprintf(hlp, "%*d",4, num+1);
        cout << hlp;
        cout << " " << sender.substr(0, 20) << " ";
        if (opt_d)
            cout << d << " ";
        cout << subject.substr(0, opt_d ? 40 : 50)<<endl;
}

// Anzeigen der Kopfzeilen aller eingelesenen E-Mails
void mailbox::showheader()
{
        for (int i=0; i<emails.size(); i++) {
                statusline(emails[i].getfrom(), emails[i].getsubject(), emails[i].getdate(), i);
        }
}

// Holen eines E-Mailtextes

void mailbox::getmail(int num)
{
        if (num<emails.size())
                emails[num].getmail();
}

// Anzeigen einer E-Mail der Box
```

```
void mailbox::showmail(int num, ostream& out=cout)
{
        if (num<emails.size())
                emails[num].printbody(out);
}

// Anzeigen einer E-Mail der Box

void mailbox::showheader(int num)
{
        if (num<emails.size())
                emails[num].printheader(cout);
}

// Mails mit dem selben Absender wie E-Mail Nr. email anzeigen
void mailbox::showsamesender(int email)
{
    for (int i = 0; i < emails.size(); i++)
        if (i != email && emails[i].getemil() == emails[email].getemil()) {
            emails[i].getmail();
            system("clear");
            emails[i].printbody(cout);
            string dummy;
            getline(cin, dummy);
        }
}

// interaktiver Dialog zur E-Mailanzeige

void mailbox::dialog()
{
        string eingabe;
        int start = 0;
        int select = 0;
        bool header = false;
        while (eingabe!="q") {
                system("clear");
                if (select == 0)
                    for (int i=start; i<start+23 && i<emails.size(); i++)
                        statusline(emails[i].getfrom(), emails[i].getsubject(), emails[i].getdate(), i);
                else
                    for (int i=0; i<emails.size(); i++)
                        if (emails[i].getemil() == emails[select-1].getemil())
                            statusline(emails[i].getfrom(), emails[i].getsubject(), emails[i].getdate(), i);
                cout <<"q=quit; u=up; d=down; h=header; s=sender; enter number to display: ";
                getline(cin, eingabe);
                int hlp;
                switch (*(eingabe.c_str())) {
                        case 'q':       break;
```

```
                        case 'u':       if ((start -= 23)<0)
                                                start = 0;
                                        break;
                        case 'd':       if ((start + 23)<emails.size())
                                                start+=23;
                                        break;
                        case 'h':       header = !header;
                                        break;
                        case 's':       if (eingabe.size() == 1)
                                            select = 0;
                                        else {
                                            select = atoi(eingabe.c_str()+1);
                                            if (select <= 0 || select > emails.size())
                                                select = 0;
                                        }
                                        break;
                        default:
                                hlp = atoi(eingabe.c_str())-1;
                                if (hlp>=0 && hlp<emails.size()) {
                                        emails[hlp].getmail();
                                        system("clear");
                                        if (!header)
                                            emails[hlp].printbody(cout);
                                        else
                                            showheader(hlp);
                                        string dummy;
                                        getline(cin, dummy);
                                }
                }
        }
}
```

mailbox.h

```
#ifndef _mailbox
#define _mailbox

#include <vector>
#include "email.h"

class mailbox {
        vector<email> emails;
        bool validbox;                  // hat das Einlesen geklappt?
        char *errmessage;
        bool issamesender(int i, int j);
public:
        mailbox(char *path);            // Einlesen aller E-Mails auf dem Path
        bool operator!();               // Abpruefen auf Erfolg beim Einlesen
        const char *errmesg()           // Fehlermeldung beim Einlesen
        { return errmessage;    }
```

```
        void showheader();                      // Anzeigen aller Kopfzeilen
        void getmail(int number);       // hole E-Mail Nr. number
        void showmail(int, ostream& out=cout);  // zeige Text der E-Mail an
        void showheader(int);                   // zeige Header der E-Mail an
        void showsamesender(int);               // EMails mit demselben Absender anzeigen
        void dialog();                  // interaktive Mailanzeige
};
#endif
```

date.h

```
#ifndef _DATE_H
#define _DATE_H

#include <iostream>
#include <sstream>
#include <stdio.h>

class Date {
private:
    int day, month, year;
    int hour, minute, second;

        // liefert true, falls year ein Schaltjahr ist
    static bool isLeapYear(int year);
        // liefert die Anzahl der Tage im angegebenen Jahr
    static int getDaysInYear(int year);
        // liefert die Anzahl der Tage im angegebenen Monat
    static int getDaysInMonth(int month, int year);
        // konvertiert das Datum in eine Integer
    static long long dateToInt(const Date &d);
        // konvertiert die Integer wieder in ein Datum
    static Date intToDate(long long val);
public:
        // legt ein neues Datum an
        // (ist das angegebene Datum nicht korrekt, so kommt
        // es zu einer Exception)
    Date(int day = 1, int month = 1, int year = 1900, int hour = 0, int minute = 0, int second = 0);
        // legt ein neues Datum an, wobei der
        // uebergebene String die Form dd.mm.yyyy haben muss
        // (ist das angegebene Datum nicht korrekt, so kommt
        // es zu einer Exception)
    Date(const char *date);
        // liefert den Tag des Datums
    int getDay();
        // liefert den Monat des Datums
    int getMonth();
        // liefert das Jahr des Datums
    int getYear();
    int getHour() const { return hour; }
```

```
    int getMinute() const { return minute; }
    int getSecond() const { return second; }
        // addiert seconds Sekunden zum Datum (inkl. Zeit) hinzu
        // (ergibt sich dabei ein Datum vor dem 1.1.1900,
        // so kommt es zu einer Exception)
    void add(int seconds);

    bool operator<=(Date&);
};

ostream &operator<<(ostream&, Date&);

#endif
```

date.cc

```
#include <iostream>
#include <sstream>
#include <stdio.h>
#include "date.h"

bool Date::isLeapYear(int year) {
        // ein Jahr ist Schaltjahr, falls es durch 4 teilbar ist UND
        // nicht durch 100 oder durch 400 teilbar ist
    return (year % 4) == 0 &&
           ((year % 100) != 0 || (year % 400) == 0);
}

inline int Date::getDaysInYear(int year) {
    return isLeapYear(year) ? 366 : 365;
}

inline int Date::getDaysInMonth(int month, int year) {
    const static int daysInMonth[] = {31,28,31,30,31,30,31,31,30,31,30,31};
        // Wert aus dem Array daysInMonth uebernehemen
        // AUSSER beim Februar in einem Schaltjahr
    return (month == 2 && isLeapYear(year)) ? 29 : daysInMonth[month-1];
}

long long Date::dateToInt(const Date &d) {
    long long val = 0;

    for (int i = 1900; i < d.year; i++)
        val += getDaysInYear(i);
    for (int i = 1; i < d.month; i++)
        val += getDaysInMonth(i, d.year);
    val += d.day;

    val = val * 24 + d.hour;
    val = val * 60 + d.minute;
```

```
    val = val * 60 + d.second;

    return val;
}

Date Date::intToDate(long long val) {
    Date d;

    d.second = val % 60;
    val /= 60;
    d.minute = val % 60;
    val /= 60;
    d.hour = val % 24;
    val /= 24;

    for (d.year = 1900; val > getDaysInYear(d.year); d.year++)
        val -= getDaysInYear(d.year);
    for (d.month = 1; val > getDaysInMonth(d.month, d.year); d.month++)
        val -= getDaysInMonth(d.month, d.year);
    d.day = val;

    return d;
}

Date::Date(int day = 1, int month = 1, int year = 1900, int hour = 0, int minute = 0, int second = 0) {
    if (year < 1900)
        throw "year must not be less than 1900";
    if (month < 1 || month > 12)
        throw "month must be between 1 and 12";
    if (day < 1 || day > getDaysInMonth(month, year))
        throw "days must be between 1 and 28/29/30/31";
    if (hour < 0 || hour > 23)
        throw "hour must be between 0 and 23";
    if (minute < 0 || minute > 59)
        throw "minute must be between 0 and 59";
    if (second < 0 || second > 59)
        throw "second must be between 0 and 59";

    this->day = day;
    this->month = month;
    this->year = year;
    this->hour = hour;
    this->minute = minute;
    this->second = second;
}

Date::Date(const char *date) {
    int day, month, year;
    int hour, minute, second;
```

```
    char s[100], s1[100], s2[100];

    int n;
    if ((n = sscanf(date, "%d %s %d %d:%d:%d %s %s", &day, s, &year, &hour, &minute, &second, s1, s2)) < 6)
        throw "date has not the right form";

    // Sonderbehandlung fuer 2-stellige Jahreszahlen
    if (year >= 0 && year < 50)
        year += 2000;
    else if (year >= 50 && year < 100)
        year += 1900;

    char *p = s;
    while (*p != 0) {
        if (*p >= 'a' && *p <= 'z')
            *p = *p - 'a' + 'A';
        p++;
    }

    if (!strcmp(s, "JAN"))              month = 1;
    else if (!strcmp(s, "FEB"))         month = 2;
    else if (!strcmp(s, "MAR"))         month = 3;
    else if (!strcmp(s, "APR"))         month = 4;
    else if (!strcmp(s, "MAY"))         month = 5;
    else if (!strcmp(s, "JUN"))         month = 6;
    else if (!strcmp(s, "JUL"))         month = 7;
    else if (!strcmp(s, "AUG"))         month = 8;
    else if (!strcmp(s, "SEP"))         month = 9;
    else if (!strcmp(s, "OCT"))         month = 10;
    else if (!strcmp(s, "NOV"))         month = 11;
    else if (!strcmp(s, "DEC"))         month = 12;
    else
        throw "month must be Jan, Feb, ..., Dec";

    bool pm = false;
    if (n >= 7 && (!strcmp(s1, "PM") || !strcmp(s1, "pm"))) {
/*
        if (hour > 11)
            throw "hour must not be greater than 11 when using pm";
*/
        pm = true;
    }

    if (n >= 8 && *s1 != '+' && *s1 != '-' && (*s2 == '+' || *s2 == '-'))
        strcpy(s1, s2);

    int inc_hour = 0, inc_min = 0;
    if (n >= 7 && (*s1 == '+' || *s1 == '-')) {
        int k = atoi(s1+1);
```

```
        inc_hour = k / 100;
        inc_min = k % 100;
        if (*s1 == '+') {
            inc_hour *= -1;
            inc_min *= -1;
        }
    }

    if (pm)
        inc_hour += 12;

    if (year < 1900)
        throw "year must not be less than 1900";
    if (day < 1 || day > getDaysInMonth(month, year))
        throw "days must be between 1 and 28/29/30/31";
    if (hour < 0 || hour > 23)
        throw "hour must be between 0 and 23";
    if (minute < 0 || minute > 59)
        throw "minute must be between 0 and 59";
    if (second < 0 || second > 59)
        throw "second must be between 0 and 59";

    this->day = day;
    this->month = month;
    this->year = year;
    this->hour = hour;
    this->minute = minute;
    this->second = second;

    int inc = (inc_hour * 60 + inc_min) * 60;
    add(inc);
}

inline int Date::getDay() {
    return day;
}

inline int Date::getMonth() {
    return month;
}

inline int Date::getYear() {
    return year;
}

void Date::add(int seconds) {
    long long val = dateToInt(*this) + seconds;

    if (val <= 0)
```

```
        throw "dates before 1.1.1900 are not supported";

    *this = intToDate(val);
}

bool Date::operator<=(Date &d) {
    return dateToInt(*this) <= dateToInt(d);
}

ostream &operator<<(ostream &out, Date &d) {
    char *months[] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
    out << d.getDay() << " " << months[d.getMonth()-1] << " " << d.getYear();
    char s[20];
    sprintf(s, " %2d:%02d:%02d", d.getHour(), d.getMinute(), d.getSecond());
    //out << s;
    return out;
}
```