`date`, a C++ class for dates.

Bernt Arne Ødegaard

April 2007

Chapter 1

Date

This is the documentation for a C++ date class.

The date class is pretty rough. A date is stored as three integers (year, month, day). Functions for comparing dates, incrementing dates are provided. Also calculation of time between dates.

1.1 Setting up.

Compile date.cc, put into library libdate.a, make library available.

Put date.h on header file path.

1.2 Defining dates.

Examples of date definitions

`date d;`

`date d(19930624)`

`date d(24,6,1993)` Note the day, month, year sequence.

1.3 Operations on dates.

Let `d, d1, d2` be dates.

The following operations are defined:

Iteration: `d++`, `++d`, `d--`, `--d` .

Logical comparisions >, >=, <, <=, ! =, ==.

Some aritmetric operations are defined, which has a meaning that may seem strange at first, but they are natural for usage purposes.

Adding/Subtracting x number of days from a data:

`d=d1+7` gives `d` the date 7 days after date `d1`.

`d=d1-7` gives `d` the date 7 days before date `d1`.

Finding the number of days between 2 dates. `i=d1-d2`. Here `i` is an integer that is the number of days between `d1` and `d2`. `i` will be negative if `d1` $<$ `d2`.

Example:

```
date d1(30,6,1993); date d2(19930610);
i = d1-d2
```

Gives `i` the value 20, the number of days between 10 jun 93 and 30 jun 93.

```
// file: date.h
// author: Bernt A Oedegaard.

#ifndef _DATE_H_
#define _DATE_H_

#include <iostream>
using namespace std;

class date {
protected:
    int year_;
    int month_;
    int day_;
public:
    date();
    date(const int& d, const int& m, const int& y);

    bool valid(void) const;

    int day()  const;
    int month() const;
    int year() const;

    void set_day (const int& day );
    void set_month (const int& month );
    void set_year (const int& year );

    date operator ++(); // prefix
    date operator ++(int); // postfix
    date operator --(); // prefix
    date operator --(int); // postfix
};

bool operator == (const date&, const date&); // comparison operators
bool operator != (const date&, const date&);
bool operator < (const date&, const date&);
bool operator > (const date&, const date&);
bool operator <= (const date&, const date&);
bool operator >= (const date&, const date&);

ostream& operator << ( ostream& os, const date& d); // output operator

#endif
```

Header file 1.1: Header file

```
#include "date.h"

//////////////////////////// construction /////////

date::date(const int& d, const int& m, const int& y) {
   day_  = d;
   month_ = m;
   year_ = y; // this assumes year is given fully, not Y2K corrections
};

//////////////////////////// inline definitions /////////

date::date(){ year_ = 0; month_ = 0; day_ = 0;};

int date::day()  const { return day_; };
int date::month() const { return month_; };
int date::year() const { return year_; };

void date::set_day (const int& day) { date::day_ = day; };
void date::set_month(const int& month) { date::month_ = month; };
void date::set_year (const int& year) { date::year_ = year; };

bool date::valid() const {
// This function will check the given date is valid or not.
// If the date is not valid then it will return the value false.
//  Need some more checks on the year, though
    if (year_ <0)               return false;
    if (month_>12 || month_<1) return false;
    if (day_>31 || day_<1)   return false;
    if ((day_==31 &&
         ( month_==2 || month_==4 || month_==6 || month_==9 || month_==11) ) )
        return false;
    if ( day_==30 && month_==2) return false;
    if ( year_<2000){
        if ((day_==29 && month_==2) && !((year_-1900)%4==0)) return false;
    };
    if ( year_>2000){
        if ((day_==29 && month_==2) && !((year_-2000)%4==0)) return false;
    };
    return true;

};
```

C++ Code 1.1: Defining the basic operations

```
#include "date.h"

bool operator == (const date& d1,const date& d2){
// check for equality
    if (!d1.valid()) { return false; };
    if (!d2.valid()) { return false; };
    if( (d1.day()==d2.day())
        && (d1.month()==d2.month())
        && (d1.year()==d2.year())) {
        return true;
    };
    return false;
}

bool operator !=(const date& d1, const date& d2){
    return !(d1==d2);
}

bool operator < (const date& d1, const date& d2){
    if (!d1.valid()) { return false; }; // not meaningful, return anything
    if (!d2.valid()) { return false; }; // should really be an exception, but ?
    if (d1.year()<d2.year())    { return true;}
    else if (d1.year()>d2.year()) { return false;}
    else {                              // same year
        if (d1.month()<d2.month()) { return true;}
        else if (d1.month()>d2.month()) { return false;}
        else { // same month
            if ( d1.day()<d2.day()) { return true;}
            else { return false; }
        };
    };
    return false;
};

bool operator > (const date& d1, const date& d2) {
    if (d1==d2) { return false;}; // this is strict inequality
    if (d1<d2) { return false; };
    return true;
}

bool operator <=(const date& d1, const date& d2){
  if (d1==d2) { return true; }
  return (d1<d2);
}

bool operator >=(const date& d1, const date& d2) {
    if (d1==d2) { return true;};
    return (d1>d2);
};
```

C++ Code 1.2: Comparisons

```
#include "date.h"

inline date next_date(const date& d){
    date ndat;
    if (!d.valid()) { return ndat; };
    ndat=date((d.day()+1),d.month(),d.year()); if (ndat.valid()) return ndat;
    ndat=date(1,(d.month()+1),d.year()); if (ndat.valid()) return ndat;
    ndat = date(1,1,(d.year()+1));       return ndat;
}

inline date previous_date(const date& d){
    date ndat;
    if (!d.valid()) { return ndat; }; // return zero
    ndat = date((d.day()-1),d.month(),d.year()); if (ndat.valid()) return ndat;
    ndat = date(31,(d.month()-1),d.year()); if (ndat.valid()) return ndat;
    ndat = date(30,(d.month()-1),d.year()); if (ndat.valid()) return ndat;
    ndat = date(29,(d.month()-1),d.year()); if (ndat.valid()) return ndat;
    ndat = date(28,(d.month()-1),d.year()); if (ndat.valid()) return ndat;
    ndat = date(31,12,(d.year()-1));       return ndat;
};

date date::operator ++(int){ // postfix operator
   date d = *this;
   *this = next_date(d);
   return d;
}

date date::operator ++(){ // prefix operator
   *this = next_date(*this);
   return *this;
}

date date::operator --(int){ // postfix operator, return current value
   date d = *this;
   *this = previous_date(*this);
   return d;
}

date date::operator --(){ // prefix operator, return new value
   *this = previous_date(*this);
   return *this;
};

inline long long_date(const date& d) {
    if (d.valid()){ return d.year() * 10000 + d.month() * 100 + d.day(); };
    return -1;
};

ostream & operator << (ostream& os, const date& d){
    if (d.valid()) {       os << " " << long_date(d) << " " ; }
    else {       os << " invalid date "; };
    return os;
}
```

C++ Code 1.3: Iteration