```
Script started on Sat 13 Jul 2013 04:45:59 PM CDT
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ pwd
/home/jakob/cplusplus
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat date.
info
Name: Jakob Hansen
Class: CSC216
Lab: Date Class, assignment 2.23

Levels attempted:

3 - for doing the assignment?


Program Description:

A "Date" class to make working with dates from January 1, 1800 through December
31, 2500 straightforward and easy! Handles input/output of dates, difference
between dates, < and > operations as well as incrementing of dates by a certain
number of days. Might I also add: uses a snazzy modified binary search for
lightnin' fast searching within a range!
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat date.
h
#ifndef DATE_H
#define DATE_H


// Modified binary search. Returns the lower of two adjacent indices between
// which the searched-for object lies: returns n when a[n] <= findMe < a[n+1].
template <typename arrT>
arrT rangesearch(const arrT * a, long max, const arrT & findMe)
{
    arrT low = 0;
    arrT mid;
    arrT found = -1;

    while (low <= max && found == -1)
    {
        mid = (low+max)/2;
        if (a[mid] < findMe)
        {
            low = mid+1;
        }
        else if (a[mid] > findMe)
        {
            max = mid-1;
        }

        if (a[mid] <= findMe && findMe < a[mid+1])
        {
            found = mid;
        }
    }
    return found;
}


class Date
{
    long total_days, flags;
    static long WHOLE_MONTHS [];
    static long WHOLE_MONTHS_LEAP [];
    static long WHOLE_YEARS [];

    public:
    enum { day_bound = 1, month_bound = 2, year_bound = 4,
                      notation_fail = 8, fail_bit = 15, leap_bit = 16 };
    Date(void) : total_days(0), flags(0) { }
    Date(long days) : total_days(days), flags(0)
    {
        set_date(days);
    }
    Date(short day, short month, short year) : total_days(0), flags(0)
    {
        set_date(day, month, year);
    }
    Date(const Date & other) :
                      total_days(other.total_days), flags(other.flags) { }
    bool good(void) const
    {
        return (flags & fail_bit) == 0;
    }
    bool bad_days(void) const
    {
        return (flags & day_bound) != 0;
    }
    bool bad_month(void) const
    {
        return (flags & month_bound) != 0;
    }
    bool bad_year(void) const
    {
        return (flags & year_bound) != 0;
    }
    bool bad_notation(void) const
    {
        return (flags & notation_fail) != 0;
    }
    bool is_leap(void) const
    {
        return (flags & leap_bit) != 0;
    }
    void set_flag(long flagbit)
    {
        flags |= flagbit;
    }
    void unset_flag(long flagbit)
    {
        flags &= ~flagbit;
    }
    void clear_flags(void)
    {
        unset_flag(fail_bit);
    }
    void set_date(long days)
    {
        if (days+total_days > WHOLE_YEARS[700] || days+total_days < 0)
        {
            set_flag(year_bound);
        }
        else
        {
            total_days = days;
            unset_flag(year_bound);
        }
        if(good())
        {
            long year = get_year();
            if (((year % 4 == 0) && (year % 100 != 0)) ||
            ((year % 4 == 0) && (year % 100 == 0) && (year % 400 == 0)))
            {
                set_flag(leap_bit);
            }
            else
            {
                unset_flag(leap_bit);
            }
        }
```

```
        return;
    }
    void set_date(short month, short day, short year)
    {
        if (((year % 4 == 0) && (year % 100 != 0)) ||
            ((year % 4 == 0) && (year % 100 == 0) && (year % 400 == 0)))
        {
            set_flag(leap_bit);
        }
        else
        {
            unset_flag(leap_bit);
        }
        if (month > 12)
        {
            set_flag(month_bound);
        }
        if (year > 2500 || year < 1800)
        {
            set_flag(year_bound);
        }
        if ((day > 30 && (month==4 || month==6 || month==9 || month==11)) ||

           (day > 31) ||
           ((day > 28 && month == 2 && !is_leap()) ||
             (day > 29 && month == 2 && is_leap())))
        {
            set_flag(day_bound);
        }
        if (good())
        {
            total_days += WHOLE_YEARS[year-1800];
            total_days += is_leap() ? WHOLE_MONTHS_LEAP[month-1]
                                    : WHOLE_MONTHS[month-1];
            total_days += day-1;
        }
        return;
    }
    long get_year(void) const
    {
        return rangesearch(WHOLE_YEARS, 700, total_days) + 1800;
    }

    long get_month(void) const
    {
        return rangesearch(is_leap() ? WHOLE_MONTHS_LEAP
                                     : WHOLE_MONTHS, 11,
                                total_days - WHOLE_YEARS[get_year()-1800])+1;
    }
    long get_day(void) const
    {
        return total_days - WHOLE_YEARS[get_year()-1800] -
                         (is_leap() ? WHOLE_MONTHS_LEAP[get_month()-1]
                                    : WHOLE_MONTHS[get_month()-1])+1;
    }
    Date & operator+=(long days)
    {
        set_date(total_days+days);
        return *this;
    }
    long operator-(const Date & right) const
    {
        return total_days - right.total_days;
    }
    bool operator<(const Date & right) const
    {
        return total_days < right.total_days;
    }
};

std::istream & operator>>(std::istream & in, Date & d);
std::ostream & operator<<(std::ostream & out, Date & d);

#endif

\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat date.
cpp
#include <iostream>
#include <climits>
#include "date.h"

using namespace std;

long Date::WHOLE_MONTHS [ ] = { 0, 31, 59, 90, 120, 151,
                                181, 212, 243, 273, 304, 334, 365 };
long Date::WHOLE_MONTHS_LEAP [ ] = { 0, 31, 60, 91, 121, 152, 182,
    213, 244, 274, 305, 335, 366 };



long Date::WHOLE_YEARS [ ] =
{ 0, 365, 730, 1095, 1460, 1826, 2191, 2556, 2921, 3287, 3652, 4017, 4382, 4748,
5113, 5478, 5843, 6209, 6574, 6939, 7304, 7670, 8035, 8400, 8765, 9131, 9496,
9861, 10226, 10592, 10957, 11322, 11687, 12053, 12418, 12783, 13148, 13514,
13879, 14244, 14609, 14975, 15340, 15705, 16070, 16436, 16801, 17166, 17531,
17897, 18262, 18627, 18992, 19358, 19723, 20088, 20453, 20819, 21184, 21549,
21914, 22280, 22645, 23010, 23375, 23741, 24106, 24471, 24836, 25202, 25567,
25932, 26297, 26663, 27028, 27393, 27758, 28124, 28489, 28854, 29219, 29585,
29950, 30315, 30680, 31046, 31411, 31776, 32141, 32507, 32872, 33237, 33602,
33968, 34333, 34698, 35063, 35429, 35794, 36159, 36524, 36889, 37254, 37619,
37984, 38350, 38715, 39080, 39445, 39811, 40176, 40541, 40906, 41272, 41637,
42002, 42367, 42733, 43098, 43463, 43828, 44194, 44559, 44924, 45289, 45655,
46020, 46385, 46750, 47116, 47481, 47846, 48211, 48577, 48942, 49307, 49672,
50038, 50403, 50768, 51133, 51499, 51864, 52229, 52594, 52960, 53325, 53690,
54055, 54421, 54786, 55151, 55516, 55882, 56247, 56612, 56977, 57343, 57708,
58073, 58438, 58804, 59169, 59534, 59899, 60265, 60630, 60995, 61360, 61726,
62091, 62456, 62821, 63187, 63552, 63917, 64282, 64648, 65013, 65378, 65743,
66109, 66474, 66839, 67204, 67570, 67935, 68300, 68665, 69031, 69396, 69761,
70126, 70492, 70857, 71222, 71587, 71953, 72318, 72683, 73048, 73414, 73779,
74144, 74509, 74875, 75240, 75605, 75970, 76336, 76701, 77066, 77431, 77797,
78162, 78527, 78892, 79258, 79623, 79988, 80353, 80719, 81084, 81449, 81814,
82180, 82545, 82910, 83275, 83641, 84006, 84371, 84736, 85102, 85467, 85832,
86197, 86563, 86928, 87293, 87658, 88024, 88389, 88754, 89119, 89485, 89850,
90215, 90580, 90946, 91311, 91676, 92041, 92407, 92772, 93137, 93502, 93868,
94233, 94598, 94963, 95329, 95694, 96059, 96424, 96790, 97155, 97520, 97885,
98251, 98616, 98981, 99346, 99712, 100077, 100442, 100807, 101173, 101538,
101903, 102268, 102634, 102999, 103364, 103729, 104095, 104460, 104825, 105190,
105556, 105921, 106286, 106651, 107017, 107382, 107747, 108112, 108478, 108843,
109208, 109573, 109938, 110303, 110668, 111033, 111399, 111764, 112129, 112494,
112860, 113225, 113590, 113955, 114321, 114686, 115051, 115416, 115782, 116147,
116512, 116877, 117243, 117608, 117973, 118338, 118704, 119069, 119434, 119799,
120165, 120530, 120895, 121260, 121626, 121991, 122356, 122721, 123087, 123452,
123817, 124182, 124548, 124913, 125278, 125643, 126009, 126374, 126739, 127104,
127470, 127835, 128200, 128565, 128931, 129296, 129661, 130026, 130392, 130757,
131122, 131487, 131853, 132218, 132583, 132948, 133314, 133679, 134044, 134409,
134775, 135140, 135505, 135870, 136236, 136601, 136966, 137331, 137697, 138062,
138427, 138792, 139158, 139523, 139888, 140253, 140619, 140984, 141349, 141714,
142080, 142445, 142810, 143175, 143541, 143906, 144271, 144636, 145002, 145367,
145732, 146097, 146462, 146827, 147192, 147557, 147923, 148288, 148653, 149018,
149384, 149749, 150114, 150479, 150845, 151210, 151575, 151940, 152306, 152671,
153036, 153401, 153767, 154132, 154497, 154862, 155228, 155593, 155958, 156323,
156689, 157054, 157419, 157784, 158150, 158515, 158880, 159245, 159611, 159976,
160341, 160706, 161072, 161437, 161802, 162167, 162533, 162898, 163263, 163628,
163994, 164359, 164724, 165089, 165455, 165820, 166185, 166550, 166916, 167281,
167646, 168011, 168377, 168742, 169107, 169472, 169838, 170203, 170568, 170933,
171299, 171664, 172029, 172394, 172760, 173125, 173490, 173855, 174221, 174586,
174951, 175316, 175682, 176047, 176412, 176777, 177143, 177508, 177873, 178238,
```

```
178604, 178969, 179334, 179699, 180065, 180430, 180795, 181160, 181526, 181891,
182256, 182621, 182986, 183351, 183716, 184081, 184447, 184812, 185177, 185542,
185908, 186273, 186638, 187003, 187369, 187734, 188099, 188464, 188830, 189195,
189560, 189925, 190291, 190656, 191021, 191386, 191752, 192117, 192482, 192847,
193213, 193578, 193943, 194308, 194674, 195039, 195404, 195769, 196135, 196500,
196865, 197230, 197596, 197961, 198326, 198691, 199057, 199422, 199787, 200152,
200518, 200883, 201248, 201613, 201979, 202344, 202709, 203074, 203440, 203805,
204170, 204535, 204901, 205266, 205631, 205996, 206362, 206727, 207092, 207457,
207823, 208188, 208553, 208918, 209284, 209649, 210014, 210379, 210745, 211110,
211475, 211840, 212206, 212571, 212936, 213301, 213667, 214032, 214397, 214762,
215128, 215493, 215858, 216223, 216589, 216954, 217319, 217684, 218050, 218415,
218780, 219145, 219511, 219876, 220241, 220606, 220972, 221337, 221702, 222067,
222433, 222798, 223163, 223528, 223894, 224259, 224624, 224989, 225355, 225720,
226085, 226450, 226816, 227181, 227546, 227911, 228277, 228642, 229007, 229372,
229738, 230103, 230468, 230833, 231199, 231564, 231929, 232294, 232660, 233025,
233390, 233755, 234121, 234486, 234851, 235216, 235582, 235947, 236312, 236677,
237043, 237408, 237773, 238138, 238504, 238869, 239234, 239599, 239965, 240330,
240695, 241060, 241426, 241791, 242156, 242521, 242887, 243252, 243617, 243982,
244348, 244713, 245078, 245443, 245809, 246174, 246539, 246904, 247270, 247635,
248000, 248365, 248731, 249096, 249461, 249826, 250192, 250557, 250922, 251287,
251653, 252018, 252383, 252748, 253114, 253479, 253844, 254209, 254575, 254940,
255305, 255670, 256035 };


istream & operator>>(istream & in, Date & d)
{
    short month, day, year;
    char c;
    in >> month >> c >> day >> c >> year;

    if (c != '/')
    {
        d.set_flag(Date::notation_fail);
    }

    d.set_date(month, day, year);

    in.clear();
    in.ignore(INT_MAX, '\n');

    return in;
}

ostream & operator<<(ostream & out, Date & d)
{
    out << d.get_month() << '/' << d.get_day() << '/' << d.get_year();
    return out;
}
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat datec
lass.cpp
#include <iostream>
#include <string>
#include <climits>
#include "date.h"

using namespace std;




int main(void)
{
    Date date1, date2;
    long incby;


    cout << "\nPlease Enter a first date (mm/dd/yyyy): ";
    cin >> date1;
    while (!date1.good())
    {
        if (date1.bad_days())
        {
            cout << "\nThere are too many days!!";
        }
        if (date1.bad_month())
        {
            cout << "\nInvalid Month!!";
        }
        if (date1.bad_year())
        {
            cout << "\nInvalid Year!!";
        }
        if (date1.bad_notation())
        {
            cout << "\nImproper Notation!!";
        }
        cout << "\nPlease Try again: ";
        date1.clear_flags();
        cin >> date1;
    }
    cout << "\nPlease Enter the second date (mm/dd/yyyy): ";
    cin >> date2;
    while (!date2.good())
    {
        if (date2.bad_days())
        {
            cout << "\nThere are too many days!!";
        }
        if (date2.bad_month())
        {
            cout << "\nInvalid Month!!";
        }
        if (date2.bad_year())
        {
            cout << "\nInvalid Year!!";
        }
        if (date2.bad_notation())
        {
            cout << "\nImproper Notation!!";
        }
        cout << "\nPlease Try again: ";
        date2.clear_flags();
        cin >> date2;
    }
    cout << "\nThe first date you entered was: " << date1 << '\n';
    cout << "\nThe second date you entered was: " << date2 << '\n';
    cout << "\nThe difference between these two dates is " << date2 - date1
         << " days.\n";
    if (date2 < date1)
    {
        cout << '\n' << date1 << " comes after " << date2 << '\n';
    }
    else
    {
        cout << '\n' << date2 << " comes after " << date1 << '\n';
    }
    cout << "\nBy how many days would you like to increment Date 1? ";
    cin >> incby;
    date1+=incby;
    while (!date1.good())
    {
        cout << "\nYou must have entered a bad amount of days to increment that"
             << " date by...Please try again: ";
        cin.ignore(INT_MAX, '\n');
        cin.clear();
        cin >> incby;
        date1+=incby;
    }
```

```
    cout << "\nOk, " << date1 << " plus " << incby
         << " days is: " << date1 << '\n';

    return 0;
}


\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ CPP date
dateclass
date.cpp...
dateclass.cpp***


\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ ./datecla
ss.out

Please Enter a first date (mm/dd/yyyy): 03/17/1986

Please Enter the second date (mm/dd/yyyy): 07132031

Invalid Month!!
Please Try again: 07/13/13

Invalid Year!!
Please Try again: 07/13/2013

The first date you entered was: 3/17/1986

The second date you entered was: 7/13/2013

The difference between these two dates is 9980 days.

7/13/2013 comes after 3/17/1986

By how many days would you like to increment Date 1? 234985623948

You must have entered a bad amount of days to increment that date by...Please try a
gain: 9980

Ok, 7/13/2013 plus 9980 days is: 7/13/2013
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ exit
exit

Script done on Sat 13 Jul 2013 04:47:11 PM CDT
```