Subject: Date is broken for years below one.
Posted by Novo on Mon, 10 May 2010 14:06:15 GMT
View Forum Message <> Reply to Message

```
const Date date1(0, 1, 1);
ASSERT(date1.IsValid());
const int n = date1.Get();

Date date2;
date2.Set(n);

ASSERT(date1 == date2);
```

I couldn't find description of an algorithm used in UPP, but other libraries (like BOOST) use Julian Day Number to convert date to a number of days. Algorithm for calculation seems to be much simpler in this case.

http://beta.boost.org/doc/libs/1_42_0/boost/date_time/gregor ian_calendar.ipp

http://www.faqs.org/faqs/calendars/faq/part2/

http://en.wikipedia.org/wiki/Julian_day

Subject: Re: Date is broken for years below one.
Posted by dolik.rce on Mon, 10 May 2010 16:15:14 GMT
View Forum Message <> Reply to Message

Hi Novo,

Quite similar topic was discussed here few months ago, see this thread.
As someone noted Quote: not many dates are known so detailed from the years before 1582
I don't think this is a serious problem. Most of the algorithms using date should work just fine.

If you really need to count days, you can try to fix Date::Get() and Date::Set(). They contain just plain arithmetics, but it gets tricky when year<=0 It is because of how the integral division works.
For starter, I *think* that Date::Get)) should be int Date::Get() const {

```
 return year * 365 + s_month_off[month - 1] + (day - 1) +
        (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400 + (year>0) +
        (month > 2) * IsLeapYear(year);
}
```

For Date::Set() it is bit more complicated, there you would have to add similar correction and additionally also add few more if blocks, because now all the ifs for counting leap years check just for (d>=...).

Best regards,
Honza