Day from Date

Given a date

(day, month, year) -- Europe
(month, day, year) -- USA
(year, month, day) -- ISO (Intern'l Standards Org),

what week-day is it: Sun?, Mon? ...

Week-day, 1st day next century, (1, 1, 2001) = ?

Week-day , Christmas 2000, (25, 12, 2000) = ?

Problem Analysis

Input: day, month year. (d,m,y)

Output: day of the week

e.g. The date (31, 12, 1999) falls on a Friday, week-day 5

week-day coded Mod 7.

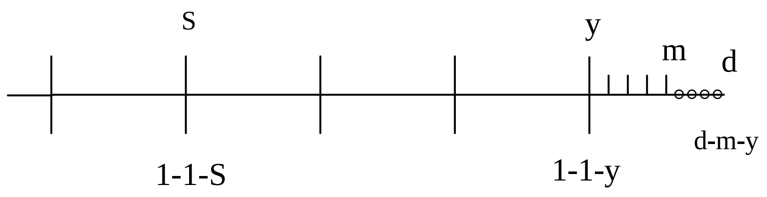
0	1	2	3	4	5	6
Sun	Mon	Tues	Wed	Thurs	Fri	Sat

Strategy

To find the day of the week of d-m-y, start with a known date, 1-1-S (first day of year S) and calculate the number of days from 1-1-S to d-m-y. Knowing the week-day of 1-1-S we can use 'mod 7' to find week-day of d-m-y. We use # for 'number of'

# days from 1-1-S to d-m-y

```
# days     =    d                                  -- # days in current month, m
                + days in months up to current month, m
                + (y-S)*365                        -- # days in years back to year S
                + (y-S) div 4                      -- add on leap days
                - ((y-S) div 100 - (y-S) div 400)  -- # correction to leap days.
```



Start Date
Consider date, 1-1-1, virtual beginning of Calendar.

tf. using expression above, (d,m,y) = (1,1,1) is day number 1, the first day.

tf. let start date, day zero, be one day before 1-1-1,

Using mod 7, we can calculate day of week from start date using

```
Num_days(d, m, y)  =   "# days from Start date"

                   =   d
                       + (sum k |  1 <= k < m : month_days(k,y)) -- y may be a leap year
                       + (y-1) * 365
                       + (y-1) div 4
                       - (y-1) div 100
                       + (y-1) div 400
```

tf.

```
date2day (d,m,y)     =     (Num_days(d,m,y)) mod 7
```

tf.
date2day(1,1,1) = 1, Monday,

Start date (day zero), the day before, is a Sunday -- week-day 0.

Knowing that the start date is a Sunday, we can use date2day to calculate the day of the week for any date.

Calculating month_days
Let days_in_month be an array indexed from 1 to 12, with value the associated number of days

days_in_month

1	2	3	4	5	6	7	8	9	10	11	12
31	28	31	30	31	30	31	31	30	31	30	31

If y is a leap year then we add 1 to number of days in February.

```
  is_leap_year(y)     =     "y is a leap year"
                            y divisible by 4 but (if y divisible by 100 then y divisible by 400)
                            (y mod 4 =0) and ((y mod 100  / = 0) or y mod 400 = 0)
```

Accumulating days, **month_day(k)**, when y is not a leap year, e.g. 1900 or 1999

1	2	3	4	5	6	7	8	9	10	11	12
31	59	90	120	151	181	212	243	273	304	334	365

Accumulating days, **leap_month_day (k)**, when y is a leap year, e.g. 2000

1	2	3	4	5	6	7	8	9	10	11	12
31	60	91	121	152	182	213	244	274	305	335	366

Eiffel functions for div and mod

Eiffel uses

n // d	for	n div d -- integer division e.g. 14 // 5 = 2
n \\ d	for	n mod -- remainder or ‘mod’ function e.g. 14 \\ 5 = 4

Eiffel Function for date2day(d,m,y)

```
date2day(d,m,y:INTEGER):INTEGER is
    do
        if is_leap_year(y) then
            Result := (d
                    + leap_month_days.item (m-1)
                    + (y -1) * 365
                    + (y-1) // 4
                    - (y-1) // 100
                    + (y-1) // 400) \\ 7
        else
            Result := (d
                    + month_days.item (m-1)
                    + (y -1) * 365
                    + (y-1) // 4
                    - (y-1) // 100
                    + (y-1) // 400) \\ 7
        end
    end -- date2day
```

```
is_leap_year(y:INTEGER):BOOLEAN is
    do
        if y \\ 100 = 0 then
            Result := y \\ 400 = 0
        else
            Result := y \\ 4 = 0
        end
    end
```

Eiffel Class for Date to Day calculation

```
class  DATE_DAY
feature
        month_days: ARRAY [INTEGER];
        leap_month_days: ARRAY [INTEGER];

        setup_months is
                local
                        k, sum: INTEGER;
                        days_in_month: ARRAY [INTEGER]
                do
                        from
                                days_in_month := <<31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31>>;
                                !! month_days.make (1, 12);
                                !! leap_month_days.make (1, 12);
                                month_days.put (31, 1);
                                leap_month_days.put (31, 1);
                                k := 2
                        until
                                k > 12
                        loop
                                sum := month_days.item (k - 1) + days_in_month.item (k);
                                month_days.put (sum, k);
                                leap_month_days.put (sum + 1, k);
                                k := k + 1
                        end
                end ; -- setup_months


        date2day (d, m, y: INTEGER): INTEGER is
                local
                        s, r: INTEGER
                do
                        setup_months;
                        s := simplify (d, y);
                        if  is_leap_year (y) then
                                r := leap_month_days.item (m - 1)
                        else
                                r := month_days.item (m - 1)
                        end ;
                        Result := (s + r) \\ 7
                end ; -- date2day
        simplify (d, y: INTEGER): INTEGER is
                do
                        Result := d + (y - 1) * 365 + (y - 1) // 4 - (y - 1) // 100 + (y - 1) // 400
                end ;

        is_leap_year (y: INTEGER): BOOLEAN is
                do
                        if  y \\ 100 = 0 then
                                Result := y \\ 400 = 0
                        else
                                Result := y \\ 4 = 0
                        end
                end ;

end  -- class DATE_DAY
```

```
class  GET_DAY
creation
        make
feature

        make is
                local
                        dd: DATE_DAY;
                        day, month, year: INTEGER
                do
                        get_date ("%NEnter day (1 <= day <= 31) : ");
                        day := num;
                        get_date ("%NEnter month (1 <= month <= 12) : ");
                        month := num;
                        get_date ("%NEnter year (1901 <= year <= 2099) : ");
                        year := num;
                        io.put_string ("%N The date is a ");
                        !! dd;
                        inspect  dd.date2day (day, month, year)
                        when  0 then
                                io.put_string ("Sunday")
                        when  1 then
                                io.put_string ("Monday")
                        when  2 then
                                io.put_string ("Tuesday")
                        when  3 then
                                io.put_string ("Wednesday")
                        when  4 then
                                io.put_string ("Thursday")
                        when  5 then
                                io.put_string ("Friday")
                        when  6 then
                                io.put_string ("Saturday")
                        end ;
                        io.new_line
                end ;

        num: INTEGER;

        get_date (msg: STRING) is
                do
                        io.put_string (msg);
                        io.read_integer;
                        num := io.last_integer
                end ;

end  -- class GET_DAY
```