# ユリウス日への変換アルゴリズムを用いた暦日妥当性チェック関数の提案と実装*

山泉 貴之†
日本アイ・ビー・エム 東京基礎研究所†

## 1 はじめに

コンピュータで業務用のデータを扱う場合、暦日の処理が事実上必須である。しかし、暦日の妥当性を確認する関数は開発者が個々に仕様を決定して実装する必要があるため、暦日妥当性の関数の実装を確認するためのテストをする必要があった。このテストを行なう場合には、

- 妥当な日付 (例: 2004 年 3 月 10 日)
- 妥当でない日付 (例: 2004 年 6 月 31 日、 2004 年 4 月 -2 日、 2004 年 13 月 4 日)

などといったいろいろな日付をテスト用の入力として与える必要があるために、一般にかなりの手間を要する。

本論文では暦日の妥当性をチェックするためのコードに対するテストを省くことを目的として、天文学や年代学で日数計算に用いられるユリウス日と暦日との間の変換ルーチンを使用した日付妥当性をチェックする関数を提案する。また実装した関数についても実用的な範囲で検証を行なったので、その結果も記述する。

## 2 ユリウス日と暦日の変換

### 2.1 ユリウス日

ユリウス日は紀元前 4713 年 1 月 1 日の正午 (GMT) を起点 (0.0 日) として積算した日数である [1]。例えば、 2004 年 3 月 10 日午前 0 時 (GMT) のユリウス日は 2453074.5 である。特に小数点以下の数字が必要でない場合には、正午 (GMT) のユリウス日をその日付を代表するユリウス日 (整数値) として使用することが多い。本論文ではユリウス日としてはこの整数値を使用して議論する。

### 2.2 ユリウス暦と グレゴリウス暦

現在使用されている暦 (太陽暦) は紀元前 45 年から採用された 4 年に 1 度の閏年を置いて 1 年を 365.25 日とするユリウス暦を元にしているが、実際の 1 年 (1 太陽年 = 約 365.2422 日) との間に約 0.0078 日のずれを生じる。このずれが無視できなくなってきたために、 1582 年に閏年を置くルールとして「西暦年号で 100 で割れる年については 400 で割り切れる年にのみ閏年を置く。」を追加した上で、 1582 年 10 月 4 日 (木曜日) の次の日を 1582 年 10 月 15 日 (金曜日) としたのが現在使用されているグレゴリウス暦である[1]。

Java[2] では JDK1.1 から java.util.GregorianCalendar クラスの実装が開始されていて、ユリウス暦とグレゴリウス暦を扱うことができるようになっている。このクラスではメソッドによりユリウス暦からグレゴリウス暦への切替えが行なわれた日付を設定できるようになっているが、日付の妥当性が 1 回のメソッド呼び出しによって直接扱えるメソッドは実装されていない [2]。業務で使用するシステムではグレゴリウス暦のみを考慮すれば十分であると考えられるが、 Java での実装状況を考慮し、ユリウス暦とグレゴリウス暦の両方を扱うこととする。以下の議論では、 1582 年 10 月 4 日以前の日付はユリウス暦で、 1582 年 10 月 15 日以降の日付はグレゴリウス暦であると考えることにする。

### 2.3 配列を用いない月と日数の関係と閏年の表現法

暦日妥当性のチェック関数を実装する場合によく行なわれるアプローチとして、各月に対応する日数を配列として定義して参照するという方法がある。しかし、この方法では月として存在しない数が入力されても正しく動作することを確認するためのテストが必要になる。本論文では年の始めを 0 日として数えた時の累積日数を $p$, それに対応する月を $n$ としたときに、 $1 \leq p \leq 366$ を満たすすべての $p$ について、 (1) 式で正しく月 $n$ を求めることのできる $x$ を求めることを考える。なお、 $g(x)$ は $x$ を超えない最大の整数を表す。

$$n = g(p/x) + 1 \tag{1}$$

ところが、年の始めを 1 月としている限り、 (1) 式を満たす $x$ は存在しない。そこで、 3 月の前に 1ヵ月が 30.5 日程度の仮想的な「月」を何ヵ月か定義し、その先頭の月の 1 日の前の日 (以下、「基準日」と書く。) を 0 日として数えた時の累積日数 (以下、単に累積日数と書く。) と月の関係を考えることにする。すなわち、追加した月の数を $a_m$, 追加した日数を $p_m$ とするとき、 $1 \leq p \leq 366$ を満たすすべての整数 $p$ について、 (2) 式を満たす $x$ を求めることにする。なお、 $n = 13, 14$ となる値が求まった場合は、次の年の 1 月, 2 月となるように補正する。

$$n = g((p + p_m)/x) + 3 - a_m \tag{2}$$

(2) 式を満たす $x$ は、 $a_m \leq n \leq a_m + 11$ を満たすすべての整数 $n$ について以下の不等式 (3) を満たす。ここで、 $n$ は基準日からの累積日数 $p$ が属する月 (先頭の月を 0 月として数える。) を、 $n$ 月の末日までの累積日数を $p_n$ とする。

$$\frac{p_n + p_m}{n+1} \leq x < \frac{p_n + p_m + 1}{n+1} \tag{3}$$

*Proposal and implementation of function for date validation with conversion routine between Julian day and Civil date.

†Takayuki Yamaizumi, Tokyo Research Laboratory, IBM Japan

[1]実際に運用が開始された年は国によって異なる。

[2]Java およびすべての Java 関連の標語およびロゴは、米国 Sun Microsystems. Inc. の米国およびその他の国における商標または登録商標である。

(3) 式が成り立つような $x$ が存在する $a_m$ と $p_m$ の組合せはいくつか考えられる。たとえば、$x = 30.6001, a_m = 4, p_m = 122$ とすると、$1 \leq p \leq 366$ を満たすすべての整数 $p$ について、(2) 式を使って正しい $n$ の値を求めることができる。

### 2.4 暦日とユリウス日の間の変換

暦日 (西暦 $Y$ 年 $M$ 月 $D$ 日) からユリウス日 ($J_0$) への変換は図 1の方法で行なうことができる (変換例は [1], [3])。

```
if (M > 3) { Y = Y - 1; M = M + 12; }
S = g(Y*365.25) + g(30.6*(M + 1.0));
j0 = S + D + 1720994.5;
if (j0 > 2299159.5) {
  A = g(Y / 100.0); B = 2.0 - A + g(A / 4.0);
}
j0 = j0 + B;
```

図 1: 暦日からユリウス日への変換アルゴリズム

また、ユリウス日 ($d_0$) から暦日 (西暦 $y$ 年 $m$ 月 $d$ 日) への変換は図 2の方法で変換できる [3]。これは、$x = 30.6001$, $a_m = 4, p_m = 122$ として計算している[3]。

```
s = g(d0+0.5); a = g(s-1867216.25)/36524.25;
if (s > 2299161.0) {
  A = s + 1.0 + a - g(a / 4.0);
} else {
  A = s;
}
B = A + 1524; C = g((B - 122.1) / 365.25); D = g(365.25 * C);
E = g((B - D) / 30.6001); d = B - D - g(30.6001*E);
if (E > 13.5) { m = E - 13; } else { m = E - 1; }
if (m > 2.5) { y = C - 4716; } else { y = C - 4715; }
```

図 2: ユリウス日から暦日への変換アルゴリズム

## 3 暦日妥当性チェック関数の実装と検証

### 3.1 関数の実装

ここで図 1 のアルゴリズムに着目すると、このアルゴリズムは計算を途中で打ち切る事がないので、暦日としては実際には存在しない日付を $Y$, $M$, $D$ として与えても何らかの値をユリウス日の出力として得られる。一方、図 2 のアルゴリズムは $d_0$ として任意の実数を入力として与えれば、暦日としては妥当な日付を得られる。

したがって、図 3 のような処理を行なう暦日妥当性チェック関数を構成できる。

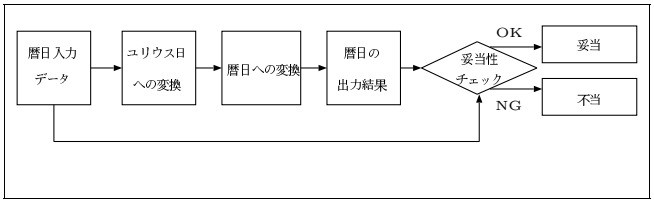


図 3: 暦日妥当性チェック関数の構成

すなわち、図 1 のアルゴリズムの出力を図 2 のアルゴリズムの入力として与えることによって得られた暦日 $y$, $m$, $d$ と図 1 のアルゴリズムの入力として与えた暦日 $Y$, $M$, $D$ を以下の方法で比較して妥当性のチェックを行なう。

- $y = Y$, $m = M$, $d = D$ がすべて成り立つ場合: 暦日として妥当。
- $y = Y$, $m = M$, $d = D$ の少なくとも 1 つが成り立たない場合: 暦日として不当。

なお、図 1, 2 のアルゴリズムをそのまま使用すると、ユリウス暦からグレゴリウス暦への切替えにともない飛ばされた 1582 年 10 月 5 日から 14 日までの日付を不当な日付として判定することができる。

### 3.2 関数の検証

3.1 で提案したアルゴリズムが業務上入力されうると考えられる日付に対して機能するかどうかを確認するために C 言語の関数として実装した。また、$1601 \leq Y \leq 2400, -10 \leq M \leq 40, -100 \leq D \leq 100$ を満たす整数 $Y$, $M$, $D$ の組合せを与えて実装した関数を呼び出すプログラムを作成して検証した。その結果、OK となっている暦日が 292194 日となった。これは 800 年に相当する日数 (365.2425 日 $\times$ 800) と一致する。また、1601 年から 2400 年の 2 月 29 日といった暦日と 2000 年から 2099 年までの暦日については正しく妥当性の判定を行なうことができることが確認できた。

## 4 むすび

本論文ではユリウス日と暦日の変換関数を組み合わせた暦日妥当性チェック関数を提案し、その有効性について検証した。最近は開発すべきシステムが高度化、複雑化しているため、日付の妥当性を扱う関数のような基本的な関数の実装とテストで作業時間を消費するのはシステムの開発上得策ではない。したがって、仕様が明らかな関数としてモジュール化することで大幅な作業効率の向上が期待できると考えている。

本論文では業務用システムに対して入力されうる値に対してのみ検証を行なった。歴史的な理由によりグレゴリウス暦への切り替えが遅れた国や地域があるが、その点については実用性が低いと思われるため一切考慮しなかった。本論文で提案した手法をグレゴリウス暦への切り替えが遅れた国の暦に対しても適用する場合には暦日として入力され得る値の範囲に合わせた条件式の修正が必要であろう。

## 参考文献


[1] こよみ, 東京大学公開講座 70, 東京大学出版会, 1999

[2] Java 2 Platform Standatd Edition v 1.4.2 API specification, http://java.sun.com/j2se/1.4.2/docs/api/, Sun Microsystems, 2003

[3] Numerial recipes in C (日本語版), Willam H. Press 他著, 丹慶 勝市他訳, 技術評論社, 1993


[3]なお、図 2のコードの 7 行目の $C$ を計算する式に 122.1 という数字が現れているが、この計算式は経過年数を計算するものであり、0.1 をプラスしているのは丸め誤差が入り込むのを防ぐためのものである。