グラフとアルゴリズムとプログラムのやさしいおはなし

渡邉敏正

2021 年 8 月 26 日

第 5 回

1　うるう年を考えてみよう

太陽系の中心にある太陽のまわりを地軸の傾いた地球が自転しながら一周することで春夏秋冬の 1 年が生じることはよく知られていることです。今回はこの一周にかかる時間と暦のうえでの 1 年との関係を取り上げます。

「うるう年」と聞くと私などは「4 年に一度 2 月 29 日がある年で、1 年が 366 日だ」と考えるだけでした。しかし、ネット検索などでよく調べてみると間違いではないのですが、厳密には正確とは言えないことがわかりました。閏年（じゅんねん、あるいは、うるうどし）と書く場合もあるようですが、ここでは「うるう年」ということにします。なお、365 日の年は「平年」、2 月 29 日は「うるう日」とよばれます。

1.1　うるう年とグレゴリオ暦

ここから、24 時間を 1 日として説明をします。太陽の周囲を地球が 1 周する時間は実は一定ではありませんので、何周かする総時間をその周回数で割った平均の時間のことになります。しかし、これもどの時点で平均を取るかで変わってきます。このようなことを言っていると話が進みませんので、1 周する時間は 365.24219 日＝ 365 日 5 時間 48 分 45.216 秒 とします[*1]。仮に暦の 1 年を 365 日 と決めたとすると、暦の上で 1 年（365 日）過ぎたときに実際に地球がある位置は 1 周する手前約 6 時間のところです (図 30 参照)。

この設定では、4 年後には 4 周目の終わりのほぼ 1 日前にいることになります。ですからこのような暦で 400 年経過したとすると、実際には地球は 400 周目の完了 100 日ほど前にいることになります。暦では冬至となっても季節はまだ 9 月の 中旬です。年とともに暦と季節感がずれていき、農耕などでは困ることになります。実際、365 日を 1 年としていた古代エジプトなどでは暦は当てにせず星の位置で農耕作業を決めていたそうです。

このような不都合を改善する試みはずっと続いたようで、現在各国で使われているグレゴリオ暦が 1582 年に制定されました。ポイントは

400 年間に 97 回だけうるう年（366 日を 1 年とする年）を設ける

[*1] 国立天文台ホームページ「よくある質問」3. 暦に関する質問:質問 3-6) どの年がうるう年になるの？ (https://www.nao.ac.jp/faq/a0306.html)

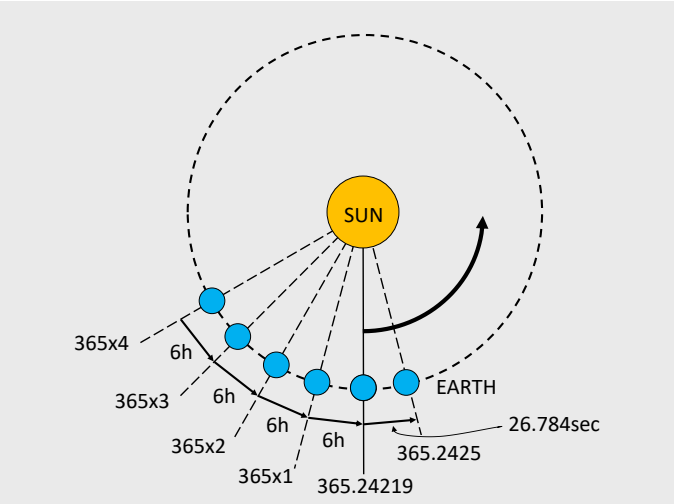

図 30　地球が太陽を一周する平均時間 365.24219 日 と 365 日 (1 日を 24 時間として) および 365.2425 日の位置関係 (イメージ図)

ということです。このように設定すると、400 年を 暦の上で見ると

$$(365 \times 303 + 366 \times 97)/400 \;=\; 365 + 97/400 \;=\; 365.2425$$

という計算で、平均で 1 年が 365.2425 日 ＝ 365 日 5 時間 49 分 12 秒 となります。一周する平均時間 365.24219 日＝ 365 日 5 時間 48 分 45.216 秒 と比べれみても 26.784 秒 長いだけです。暦の上で 1 年経過したと考える地球の位置は、実際に地球が一周した位置よりも平均で 27 秒程度進んでいるだけで、両者が非常に近くなることがわかります。(この誤差は 3000 年あまりで 1 日補正すればいい程度、ということです。) グレゴリオ暦が採用されている理由がわかると思います。

1.2　うるう年の判定方法

では、具体的にどのような規則で西暦の年号（西暦年）をうるう年と決めているのでしょうか。インターネット上[*2]に次のような規則を見つけました。(「原則として」の語句がない記述もありました。)

1. 西暦年が 4 で割り切れる年は (原則として) うるう年。
2. ただし、西暦年が 100 で割り切れる年は (原則として) 平年。
3. ただし、西暦年が 400 で割り切れる年は必ずうるう年。

さて、この規則を使って次の年号がうるう年かどうかを判定してみてください。

[*2] https://ja.wikipedia.org/wiki/閏年

1600　関ヶ原の戦い
1700　17 世紀最後の年
1736　オイラーがケーニヒスベルクの橋の問題を解く
1800　アメリカの首都がワシントン D.C. に
1868　明治元年
1900　パリ五輪（第２回夏季オリンピック大会）
1978　FIFA ワールドカップ（アルゼンチン大会）
2000　シドニーオリンピック

素早く判定できたでしょうか。私は「原則として」の扱いに悩みました。皆さんはいかがでしたか。上述の規則をそれぞれ条件１，条件２，条件３とよぶことにしましょう。私の中で、条件１、条件２、条件３と進みながらどのように年号が「うるう年」の候補として取捨選択されたかを図に示してみます。

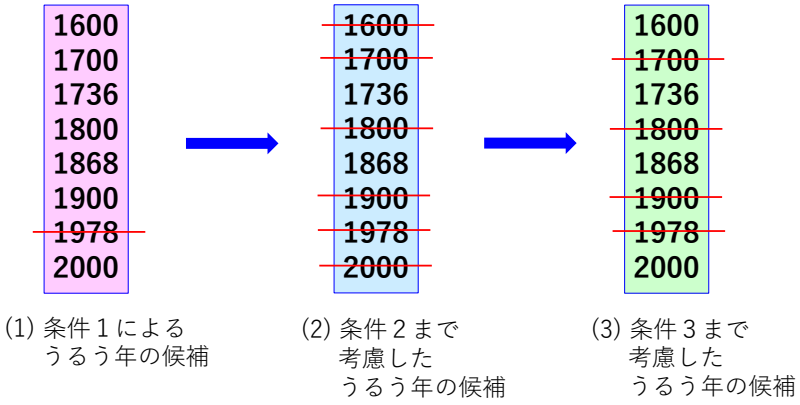

図 31　条件 1〜条件 3 による「うるう年」の判定過程 (赤線が除外を示す)

1978 以外の年号はどれも条件 1 を満たしますので、図 31(1) となります。「これらのうちで」条件２により候補から除外されるものを 100 で割り切れる年号とすると、候補として残るものは同図 (2) となります。さらにここで条件３を考えて、400 で割り切れるものを「復活させる」ことにして結局は同図 (3) になる、ということを示しています。

しかし、やはり「原則として」、「ただし」という記述と前後の「うるう年」候補との関係が分かり難いですね。いま述べた私の操作では、条件２については「これらのうちで」と対象を補足し、さらに条件３については、条件２で削除した年号も操作の対象として判定し復活させました。この辺りが条件 1 から条件３の記述で分かり難いと感じさせる原因かと思います。

4 年に一度「うるう年」を設けることを基本にしますので、4 で割り切れる年を「うるう年」の候補とすることは自然なことと思います。しかし、400 年のうち 3 回は「平年」とする必要がありますので、それをどの年にするかです。西暦 1 年〜400 年で考えてみると、最も簡単なのは、4 で割り切れる年の中で、100 年, 200 年, 300 年を平年とすればこの条件に合致します。このことを 401〜800, 801〜1200, ... と続けてみればグレゴリオ暦の条件を満たす「うるう年」の決め方になります。これで、

- 4 で割り切れる年を「うるう年」の候補とし、
- この候補も中から 100 で割り切れる年は「平年」として除外の対象とする一方で、

- 400 で割り切れる年は「うるう年」として残す

という判定方法に辿り着きます。

注 5.1　これから、西暦年がうるう年かどうかを判定するアルゴリズムを作っていきますが、その具体的なアルゴリズムはすでに書籍やインターネットなどにも示されています。ここでは、どのようにアルゴリズムは形作られていくのか、その過程の説明に重点を置いており、これが本稿の大きな目的です。合わせて、できたアルゴリズムを実際に C 言語というプログラミング言語を使ってプログラムとして記述し、それを実行するところまで説明しようと思います。

1.3　判定アルゴリズムを考える

ここで、上記の条件 1 から条件 3 のかわりに、操作の対象を明確にし、さらに削除した要素を復活させるといった後戻り的な操作を排除した形の「うるう年」の判定法を考えてみましょう。

1.3.1　流れ図

そのために、まず条件 1 から条件 3 の操作が明確になるように**流れ図** (flow chart) とよばれる図として書いてみましょう。図 32 が流れ図です。

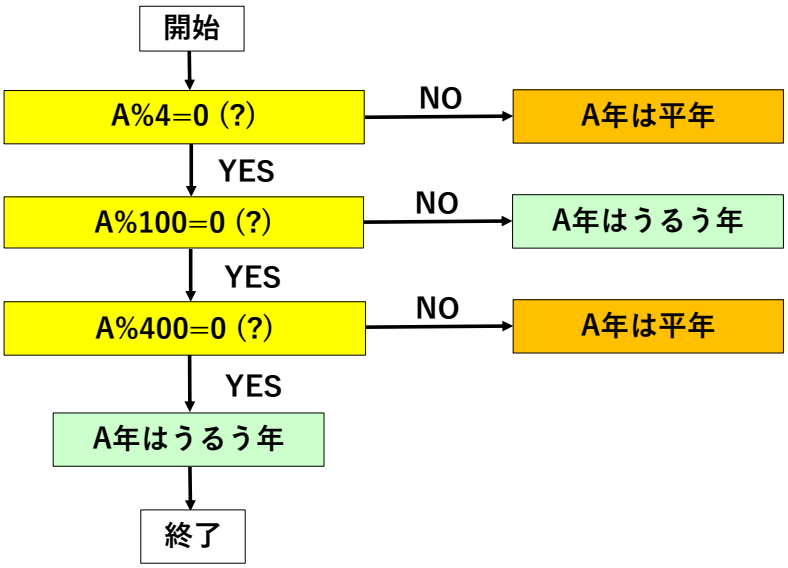

図 32　「うるう年」の判定操作を表す流れ図

この流れ図の作り方を説明しましょう。まずいくつかの準備をします。2 つの整数 p, q について、「p が q で割り切れるか否か」という判定はコンピューターでは、$p \div q$ のあまり (剰余とも言います) を求める算術演算 (この場合は除算) と、あまりが 0 に等しいかどうかの判定操作の組み合わせで実行できます。ここで、$p \div q$ のあまりを $p\%q$ という記号で表すことにします。なお、この除算は商を整数に限定します。例えば、$10 \div 3$ については整数商が 3 で剰余が 1 というです。したがって、$10\%3 = 1$ です。このとき「p が q で割り切れる

か否か」ということは

$$p\%q = 0 \text{ か否か}$$

と表現できます。この判定操作を

$$p\%q = 0\,(?)$$

と問いかけの形で表現し、その答を

$$p\%q = 0 \text{ のとき YES}, \quad p\%q \neq 0 \text{ のとき NO}$$

と表現することにします。このような記号と考え方で西暦 A 年が「うるう年」が否かの判定条件を明確に記述します。なお、「原則として」や「ただし」の部分は後で対応を考えますので、まずはそれ以外の部分を各条件文の下側に記述します。

1. 西暦年が 4 で割り切れる年は (原則として) うるう年。
   $A\%4 = 0$ ならば (原則として) うるう年。
2. ただし、西暦年が 100 で割り切れる年は (原則として) 平年。
   ただし、$A\%100 = 0$ ならば (原則として) 平年。
3. ただし、西暦年が 400 で割り切れる年は必ずうるう年。
   ただし、$A\%400 = 0$ ならば必ずうるう年。

次に、「原則として」や「ただし」の扱いを考慮して操作の関係や順序を決めます。

(条件 1 について) $A\%4 \neq 0$ ならば「平年」が確定します。一方、$A\%4 = 0$ ならば条件 1 のみでは判定できず、条件 2 や条件 3 の判定が必要ですので、このときはまず条件 2 の判定に進むことにします。したがって図 33 となります。

(条件 2 について) ここでは、A は $A\%4 = 0$ なる条件を満たす「うるう年」の候補です。$A\%100 \neq 0$ ならば「うるう年」が確定します。一方、$A\%100 = 0$ ならば「うるう年」の候補ですがまだ「平年」の可能性が残っていますので、条件 3 の判定に進みます。したがって、図 34 となります。

(条件 3 について) ここでは、A は $A\%4 = 0$ かつ $A\%100 = 0$ なる条件を満たす「うるう年」の候補ですがまだ「平年」の可能性を持っています。ここで、$A\%400 = 0$ ならば「うるう年」ですし、$A\%400 \neq 0$ ならば「平年」となり、判定が確定します。したがって、図 35 となります。

図 35 に開始と終了の操作を追加すれば全体の流れ図が完成します。これが図 32 です。

1.3.2 判定アルゴリズム

流れ図から操作の流れ（関連性）がはっきりと見えると思います。西暦 A 年 が「うるう年」かどうかの判定がどのように進むかを具体例で見てみましょう。図 36 を見てください。ここでは、

$$A = 1600,\ 1736,\ 1800,\ 1978$$

を例としてやってみます。操作の流れを図に破線で示しておきますので進み方が見えると思います。なお、判定条件については以下のように $C1$〜$C3$ の記号を付けています。

図 33　条件 1 の判定

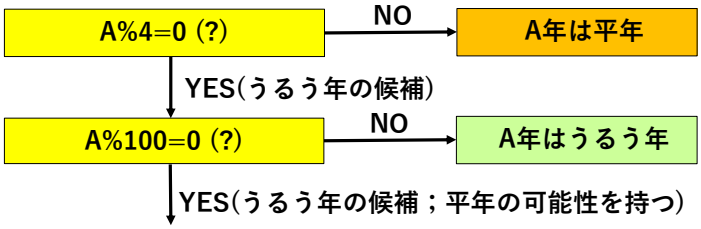

図 34　条件 2 の判定追加

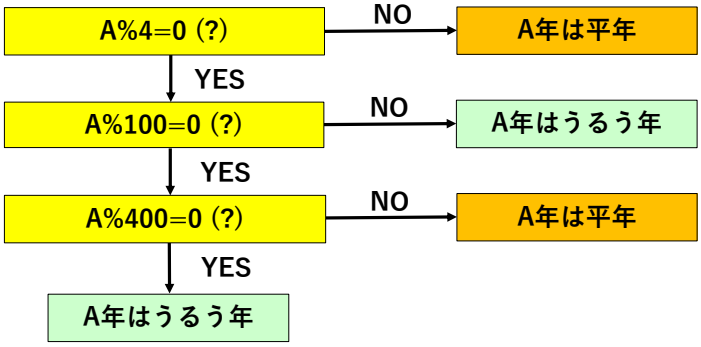

図 35　条件 3 の判定追加

$C1:\ A\%4=0 \quad C2:\ A\%100=0 \quad C3:\ A\%400=0$

また、判定結果 (うるう年や平年) および YES, NO にも識別用に記号を付けています。操作の動きを説明する場合にはこれらの記号を使うことがあります。

- $A=1978$ のとき: $A\%4 \neq 0$ ですので、$C1$ の判定で NO となって $R1$ に行って「平年」と判定されます。
- $A=1736$ のとき: $A\%4=0$ ですので $C1$ の判定で YES となって $C2$ に行きます。しかし、$A\%100 \neq 0$ ですので $C2$ の判定で NO となって、$R2$ に行って「うるう年」と判定されます。
- $A=1800$ のとき: $A\%4=0$ かつ $A\%100=0$ です。したがって $C1$, $C2$ どちらの判定も YES となって $C3$ に行きます。しかし、$A\%400 \neq 0$ ですので $C3$ の判定で NO となって、$R3$ に行って「平年」と判定されます。
- $A=1600$ のとき: $A\%4=0$ かつ $A\%100=0$ かつ $A\%400=0$ ですので、$C1$, $C2$, $C3$ すべての判定で YES となって、$R4$ に行って「うるう年」と判定されます。

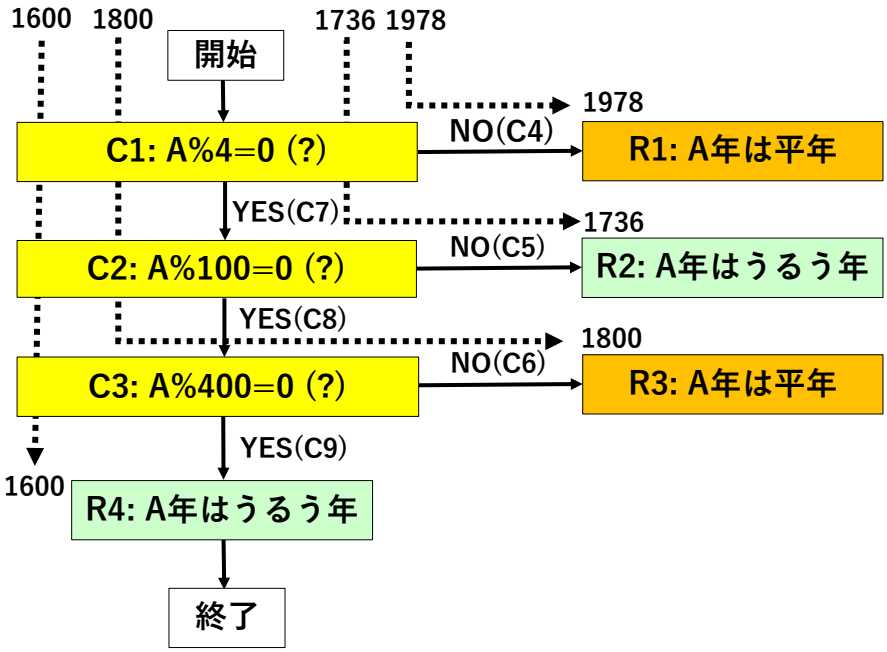

図 36　「うるう年」判定アルゴリズムの流れ図と具体的な動作

以上の流れを図 36 の破線を辿ることで確認してみてください。その他の年号についてもどのような判定になるかやってみてください。

さて、この流れ図に示したことは、西暦 A 年が「うるう年」か「平年」かを判定する操作の手順です。したがって、これは「うるう年」の**判定アルゴリズム**を表している、ということは理解していただけると思います。ただし厳密な言い方をすれば、この手順で「うるう年」かどうかをどのような年号についても正しく判定できること (**アルゴリズムの正当性** (correctness of algorithms) とよばれます) を示さねばなりません。先ほどやったことはいくつかの例について正しく判定できることを確認したにすぎないのです。しかし、一般的な正当性の証明にはいろいろな準備が必要ですので説明は別の機会に譲り、これで正しく判定できる、として話を進めることにします。

西暦 A 年が「うるう年」か「平年」かを判定するアルゴリズムの記述は以下のようになります。なお、/* ··· */の部分はコメント文と呼ばれ、アルゴリズムの動作には無関係な文章です。··· の部分に補足的な説明などを記述します。

うるう年の判定アルゴリズム

1. $A\%4 \neq 0$ ならば A 年 は平年として終了する;　/* $C1 \to$ NO(C4) $\to R1$ */
   A%4=0 ならば 2 に進む;　/* $C1 \to$ YES(C7) $\to C2$ */
2. $A\%100 \neq 0$ ならば A 年 はうるう年として終了する;　/* $C2 \to$ NO(C5) $\to R2$ */
   $A\%100 = 0$ ならば 3 に進む;　/* $C2 \to$ YES(C8) $\to C3$ */
3. $A\%400 \neq 0$ ならば A 年 は平年として終了する;　/* $C3 \to$ NO(C6) $\to R3$ */
   $A\%400 = 0$ ならば A 年 はうるう年として終了する;　/* $C3 \to$ YES(C9) $\to R4$ */

1.4 判定条件の整理・統合

図 36 は確かに操作をシンプルに表現していると思いますが、「A 年は平年」と「A 年はうるう年」が 2 つずつあります。できるだけシンプルに、という方向で各々を 1 つにまとめることを考えてみましょう。

1.4.1 条件の記述の仕方

準備として、条件の記述の仕方や用語の説明をしておきます。

西暦 A 年の値 A に対して、「$C1:\ A\%4 = 0$」の判定を考える場合に、$A\%4 = 0$ のときは

「A は $C1$ を満たす」　あるいは
「A について（A に対して）$C1$ が成り立つ」　あるいは
「A について $C1$ は正しい」

などと表現をします。なお、「A は」や「A について」は明らかなときには省略するときがあります。記述の内容によって適宜表現を選びますが、意味は同じです。一方、$A\%4 \neq 0$ のときには

「A は $C1$ を満たさない」　あるいは
「A について（A に対して）$C1$ が成り立たない」　あるいは
「A について $C1$ は正しくない」

などと表現します。以後、スペースの関係で「満たす」「満たさない」の表現のみ使います。

今後の説明の都合で、西暦 A 年が「うるう年」か「平年」かを判定するアルゴリズムを図 36 の識別記号と「満たす」「満たさない」を使って表現しておきます。

うるう年の判定アルゴリズム (識別記号での表現)

1. $C1$ は満たさないならば A 年 は平年として終了する;
   $C1$ を満たすならば 2 に進む；
2. $C2$ は満たさないならば A 年 はうるう年として終了する;
   $C2$ を満たすならば 3 に進む;
3. $C3$ は満たさないならば A 年 は平年として終了する;
   $C3$ を満たすならば A 年 はうるう年として終了する;

ただし、$C1$, $C2$, $C3$ は以下の通りです。

$$C1:\ A\%4 = 0 \quad C2:\ A\%100 = 0 \quad C3:\ A\%400 = 0$$

「$A\%4 \neq 0$」という記述を「$C1$ の**否定** (negation)」といい、$\neg C1$ という記号を使って

「$\neg C1:\ A\%4 \neq 0$」

と表記します。$\neg C1$ を使うと

「$C1$ を満たさない」 と 「$\neg C1$ を満たす」　は

同じ意味になります。すぐにわかると思いますが、

$\neg[\neg C1]$ は $C1$ と同じ意味です (記号で $\neg[\neg C1] \equiv C1$ と表記します)。

$C2$、$C3$ についても同様です。

1.4.2 「かつ」と「または」について

S, T を 2 つの記述とするときに、

「$[S$ かつ $T]$ を満たす」 は 「S を満たし、同時に T も満たす」 を

意味します。なお、「$[S1$ かつ $S2$ かつ $S3]$ を満たす」は「$S1$ を満たし、同時に $S2$ も満たし、さらに同時に $S3$ も満たす」を意味します。一方、

「$[S$ または $T]$ を満たす」は
「S を満たし同時に T も満たす」、
「S を満たすが T は満たさない」、
「S は満たさないが T を満たす」の 3 つのどれか 1 つ

を意味します。(これらのどの 2 つについても 2 つが同時に正しいことはありませんので、どれか 1 つ、と言っています。) 逆に言えば、

3 つのうちのどれも「$[S$ または $T]$ を満たす」と表現できる

ということです。このように**「または」は複数の意味を持っている場合がある**、ということに注意してください。

1.4.3 判定条件を記述する

さて、上述の記述の仕方や「かつ」「または」を使って西暦 A 年が「うるう年」となる条件を記述してみると、1 つは

$$[C1:\ A\%4 = 0] \quad \text{かつ} \quad [\neg C2:\ A\%100 \neq 0]$$

を満たすときであり、もう 1 つは

$$\begin{aligned} & [C1:\ A\%4 = 0] \\ \text{かつ} \quad & [C2:\ A\%100 = 0] \\ \text{かつ} \quad & [C3:\ A\%400 = 0] \end{aligned}$$

を満たすときです。なお、2 条件を同時に満たすことはないです。結局、西暦 A 年が「うるう年」となるのは「または」を使って

$$[\ C1 \text{ かつ } \neg C2\ ] \text{ または } [\ C1 \text{ かつ } C2 \text{ かつ } C3\ ]$$

を満たすときとなります。

注 5.2 「平年」となるのは「この条件を満たさないとき」となりますが、この説明は省略します。

1.5　アルゴリズムの最終形に向けて

準備が長くなりました。図 36 に 2 つずつある「A 年 は平年」と「A 年 はうるう年」をそれぞれ 1 つずつにする、という簡単化の話に戻りましょう。詳しい説明は少し準備が必要になりますので、大まかな考え方と結論だけ伝えることにします。

1.5.1　最終的な判定条件

ポイントは、西暦 A 年が「うるう年」となる条件

$$[\ C1\ \text{かつ}\ \neg C2\ ]\ \text{または}\ [\ C1\ \text{かつ}\ C2\ \text{かつ}\ C3\ ]$$

を「A 年 はうるう年」が 1 つになるように改めることです。すぐに思いつくことは、共通している $C1$ を括り出して 1 つの条件式にできないか、ということです。共通の $C1$ を外に括り出せば、残りそうなのは $\neg C2,\ C2,\ C3$ だろうな、と予想できると思います。実際には、条件は次のようになります。

$$C1\ \text{かつ}\ [\ \neg C2\ \text{または}\ C3\ ]$$

ただし、$C1, \neg C2, C3$ は

$$C1:\ A\%4 = 0 \quad \neg C2:\ A\%100 \neq 0 \quad C3:\ A\%400 = 0$$

です。この新しい条件を使って、西暦 A 年が「うるう年」か「平年」かを判定するアルゴリズム (単純化したアルゴリズムとよぶことにします) を流れ図として示すと図 37 になります。確かに形として「A 年 は平年」と「A 年 はうるう年」が 1 つずつになっています。しかし、新しい条件では ($\neg C2$ は残っていますが) $C2$ が消えています。大丈夫でしょうか。新しい条件で大丈夫であることや $C2$ が不要であることの詳細な説明は別の機会に譲り、ここでは新しい条件の妥当性をいくつかの例で確認してみることにします。

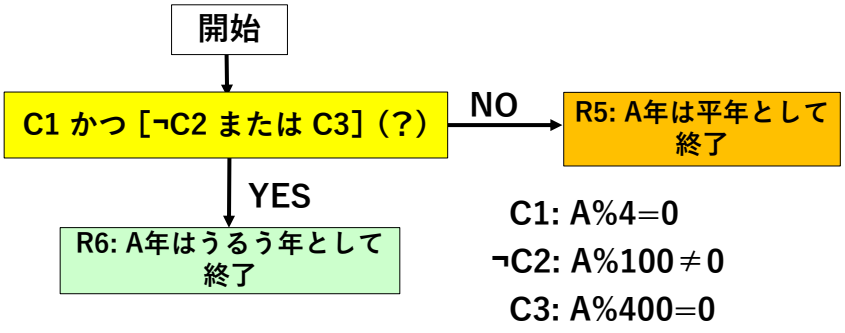

図 37　単純化したアルゴリズムの流れ図

1.5.2　「または」の否定と「かつ」の否定

準備として「S または T」全体の否定 (記号としては $\neg[\ S$ または $T\ ]$)、「S かつ T」全体の否定 (記号としては $\neg[\ S$ かつ $T\ ]$) の意味について述べておきます。なぜそうなるのかについても別の機会に説明すること

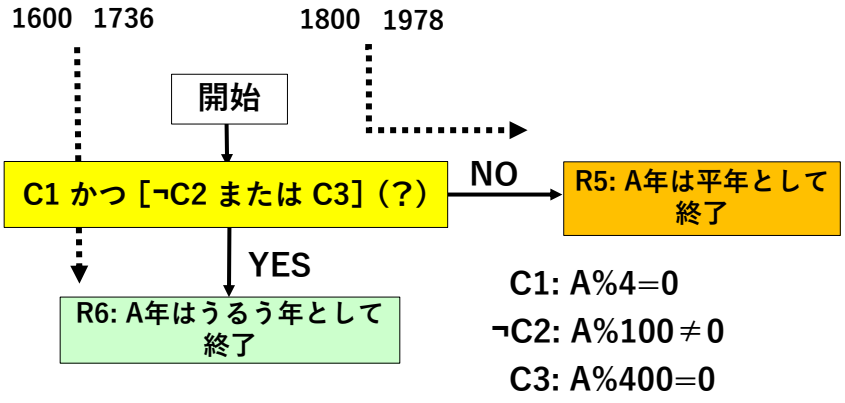

図 38　単純化したアルゴリズムの流れ図と具体的な動作

にします。これらに関して

$$\text{「}\neg[\,S \text{ または } T\,]\text{」} \quad \text{と} \quad \text{「}\neg S \text{ かつ } \neg T\text{」} \quad \text{は}$$
$$\text{「}\neg[\,S \text{ かつ } T\,]\text{」} \quad \text{と} \quad \text{「}\neg S \text{ または } \neg T\text{」} \quad \text{は}$$

それぞれ同じ意味であることが知られています。このことは記号 $\equiv$ を使って

$$\neg[S \text{ または } T] \equiv \neg S \text{ かつ } \neg T$$
$$\neg[S \text{ かつ } T] \equiv \neg S \text{ または } \neg T$$

と表されます。ここで「$[\neg C2$ または $C3]$ は満たさない」に着目てみましょう。記号で書くと

$$\neg[\neg C2 \text{ または } C3] \equiv \neg[\neg C2] \text{ かつ } \neg C3$$
$$\neg[\neg C2] \text{ かつ } \neg C3 \equiv C2 \text{ かつ } \neg C3$$

という関係があります。つまり、

「$[\neg C2$ または $C3]$ は満たさない」　と
「$\neg C2$ は満たさず $C3$ も満たさない」　と
「$C2$ を満たすが $C3$ は満たさない」　は

同じ意味であるということです。このことは、このあと図 32 と図 37 (あるいは図 36 と図 38) の対応を見るときに使います。

1.5.3　単純化したアルゴリズムが正しいこと

上述の説明も合わせて、判定条件「$C1$ かつ $[\,\neg C2$ または $C3\,]$」と $A = 1600, 1736, 1800, 1978$ について図 37 に対する動作を見ていきます。図 38 を参照してください。破線でそれぞれの値に対する動作を示しています。その結果、これらについて図 38 と図 36 の動作がきっちりと対応することがわかり、これで単純化したアルゴリズムが正しいことを確かめた、ということにします。

- $A = 1978$ のとき: $C1$ を満たさないので NO で $R5$ に行って「A 年 は平年」となります。
  (図 36 の $C1 \to$NO(C4)$\to R1$ に対応)
- $A = 1736$ のとき: $C1$ を満たし、同時に $\neg C2$ も満たします (つまり $C2$ は満たさない) ので、YES で $R6$ に行って「A 年 はうるう年」となります。
  (図 36 の $C1 \to$YES(C7)$\to C2 \to$ NO(C5) $\to R2$ に対応)

- $A = 1600$ のとき: $C1$ を満たします。$\neg C2$ は満たさない (つまり $C2$ を満たす) が、$C3$ を満たすので YES で $R6$ に行って「A 年 はうるう年」となります。

  (図 36 の $C1 \rightarrow$YES(C7)$\rightarrow$ $C2$ $\rightarrow$ YES(C8) $\rightarrow$ $C3$ $\rightarrow$ YES(C9)$\rightarrow$ $R4$ に対応)
- $A = 1800$ のとき: $C1$ を満たします。一方、$C2$ を満たすが $C3$ は満たさないので、上述の

  $$\neg[\neg C2 \text{ または } C3] \quad \equiv \quad [C2 \text{ かつ } \neg C3]$$

  という結果から「$[\neg C2$ または $C3]$ は満たさない」となり、NO で $R5$ に行って「A 年 は平年」となります。

$A = 1800$ 以外については、括弧内に示したように図 38 と図 36 の動作が対応していることは明らかだと思います。$A = 1800$ については、上述の通り $\neg[\neg C2$ または $C3] \equiv [C2$ かつ $\neg C3]$ ですので、$C1$ を満たすことを合わせれば図 36 の

$$C1 \rightarrow \text{YES(C7)} \rightarrow C2 \rightarrow \text{YES(C8)} \rightarrow C3 \rightarrow \text{NO(C6)} \rightarrow R3$$

に対応します。

以上で、これらの年号について図 38 と図 36 の動作が対応しており、単純化したアルゴリズムの正しさ（正当性）が確認できた、ということにします。

西暦 A 年が「うるう年」か「平年」かを判定する単純化したアルゴリズムの記述は以下のようになります。

うるう年を判定する単純化したアルゴリズム

[$C1$ かつ [$\neg C2$ または $C3$]] を満たすならば A 年 はうるう年として終了する;
[$C1$ かつ [$\neg C2$ または $C3$]] を満たさないならば A 年 は平年として終了する;

ただし、

$$C1 : A\%4 = 0 \quad \neg C2 : A\%100 \neq 0 \quad C3 : A\%400 = 0$$

です。

1.6 C 言語によるプログラム例

ここまで説明してきたうるう年の判定アルゴリズムをプログラムとして記述して実行することを考えてみましょう。判定アルゴリズム自体は簡単でプログラムを作らなくても暗算で判定できることです。ここでは、対象とする問題を明確にすること、アルゴリズムの作成と正当性のチェック、プログラムの作成と実行、という一連のステップを皆さんにわかりやすく説明する意図で、この簡単なアルゴリズムを取り上げています。

急にプログラムの話が出てきて驚かれる方もいるかと思いますが、分からないことは気にせずに気楽に目を通してみてください。プログラムを作って実行させるには、コンピュータの仕組や OS(Operating System: 基本ソフトウェア)、さらにはコンピュータ内部におけるプログラム処理の仕組、など多くのことを知る必要があります。これらについて別の機会に説明しますので、ここではプログラム作成と実行の流れが何となく分かる程度で差し支えありません。なお、OS としては Windows を知っている方が多いと思います。他にも macOS や UNIX などがあります。実は macOS は FreeBSD(UNIX と類似のもの) とほぼ同じです。以下では、コンピューターの仕組や OS にはあまり深入りせず、プログラムの作成や実行をできるだけ分かりやすい形で大まかに説明します。

```
1  //
2  //  leap-year.c created by Toshimasa Watanabe on 2021/06/30.
3  //
4
5  #include <stdio.h>
6
7  int main(void)
8  {
9      int A;
10     int div4=4, div100=100, div400=400;
11     int r1, r2, r3;
12
13    // input year
14     printf("\n\n");
15     printf("************************************************\n");
16     printf("        西暦A年がうるう年か否かを判定する\n");
17     printf("************************************************\n");
18     printf("\n");
19     printf("年号をキーボードから「半角で」入れてください:  A=");
20     scanf("%d",&A);
21     printf("\n");
22
23     // computing integer residues
24     r1=A%div4;
25     r2=A%div100;
26     r3=A%div400;
27
28     // determining whether or not A is a leap year
29     if ( (r1==0) && ((r2!=0)|(r3==0)) )
30     {
31         printf("西暦%d年はうるう年です \n\n", A);
32     }
33     else
34     {
35         printf("西暦%d年はうるう年ではありません \n\n  ", A);
36     }
37
38    return 0;
39 }
```

図 39　C 言語プログラムの一例 (左端の行番号は説明用に入れています: プログラムに入れるとエラーになります)

図 38 のうるう年を判定する単純化したアルゴリズムを C 言語で記述したプログラムの例を図 39 に示します。また、$A = 1978, 1800, 1736, 1600$ のぞれぞれについて実行したときのディスプレイの表示を図 40〜図 43 に示します。なお、コンピュータによっては \ (バックスラッシュ) は ¥ と表示される場合もあります。

図 39 の左端の行番号は説明用に入れています。(プログラムに入れるとエラーになります。) また、//はこの記号より右側からその行の終わりまでが注釈文であることを示します。(注釈の書き方は、これ以外に/* ··· */もあります。)

```
Toshimasa-no-MacBook-Pro:repos watanabe$ gcc -o leap-year leap-year.c
Toshimasa-no-MacBook-Pro:repos watanabe$ ./leap-year


*********************************************
     西暦A年がうるう年か否かを判定する
*********************************************

年号をキーボードから「半角で」入れてください: A=1978

西暦1978年はうるう年ではありません
```

図 40　$A = 1978$ に対するプログラムの実行結果 (ディスプレイの表示)

```
Toshimasa-no-MacBook-Pro:repos watanabe$ ./leap-year


*********************************************
     西暦A年がうるう年か否かを判定する
*********************************************

年号をキーボードから「半角で」入れてください: A=1800

西暦1800年はうるう年ではありません
```

図 41　$A = 1800$ に対するプログラムの実行結果 (ディスプレイの表示)

1.6.1　C 言語でのプログラム作成と実行

C 言語でのプログラム作成と実行を、現在一般的に使われているパーソナルコンピュータの使用を前提として説明します。次のような手順で進みます。あとで語句も含めて個々の操作を説明をします。

(プログラム作成と実行)
ソースコード作成 → (コンパイル) → オブジェクトコード作成 → (リンクなど) →
実行ファイル作成 → (OS により実行)

- まず C 言語処理システムを決めます。システムの中心はエディタとコンパイラ（リンク機能を含む）の組み合わせです。今回は Mac を使いましたので、エディタは X-code、コンパイラは GCC(実行ファイル名 (コマンド) は gcc) を使用しました。Windows を OS とするコンピュータでは C 言語処理システムとしてエディタとコンパイラが統合した処理システムもあります。以後、コンパイラを gcc と表記します。
- **エディタ** (editor) は、コンピュータでの実行を意識しながらアルゴリズムを C 言語プログラムで記述する場合に使用します。(Windows や Mac のワード、Windows のワードパッドなどと同様に文章作成機能を持ちますが、テキストのみを扱う単純な機能に限定されています。一方で、プログラム開発に便

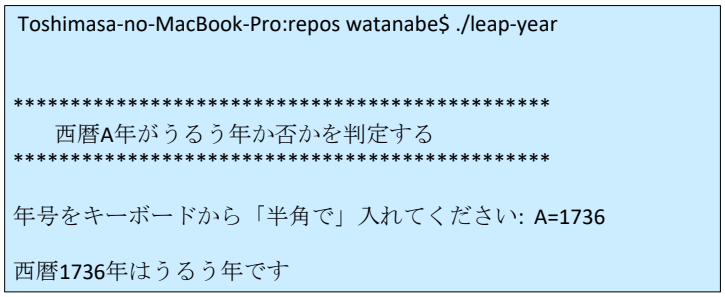

図 42　$A = 1736$ に対するプログラムの実行結果 (ディスプレイの表示)

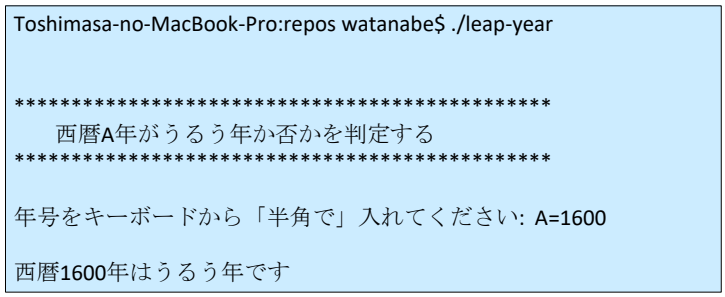

図 43　$A = 1600$ に対するプログラムの実行結果 (ディスプレイの表示)

利な機能が追加されています。) 作成したプログラム (**ソースコード** (source code) と言います) をファイルに保存します。この例ではファイル名を leap-year.c としています。拡張子は.c とします。これが図 39 のプログラムです。

- **コンパイラ** (compiler) は、ソースコード (今回では leap-year.c にあるプログラム) をコンピュータが読み取り可能な 0 と 1 の系列で書かれたプログラム (**オブジェクトコード** (object code) と言います) に変換する機能を持ちます。(この変換操作を**コンパイル** (compile) と言います。) オブジェクトコードは一般的にいくつかのオブジェクトモジュール (各々はまとまった一つの処理プログラム) の集合体になります。通常、gcc はこれらの実行順序を決め、実行に必要な種々の情報を付加して**実行可能ファイル** (executable file) とよばれるファイルを作成します。(この辺りの操作を**リンク** (link) と言います。) 今回はこの実行可能ファイルを leap-year というファイル名で保存しています。図 40 の最初の行 ($より右側の部分)

  gcc -o leap-year leap-year.c

  が leap-year.c にあるソースプログラムをコンパイル、リンクして得た実行可能ファイルを leap-year という名前のファイルとして保存せよ、というコンピュータへの操作の指示です。
- 実行可能ファイルを実行させれば、アルゴリズムで意図した動作が実際にコンピュータで実行されます。図 40 の 2 行目の ($より右側の部分)

  ./leap-year

がコンピュータへの実行の指示です。なお、このプログラムは一つの A の値について判定すると終了しますので、$A = 1978, 1800, 1736, 1600$ のぞれぞれについて実行しています。これが図 40〜図 43 で、ディスプレイの表示を示しています。図 41〜図 43 では 1 行目にある ./leap-year で実行を開始しています。

注 5.3　ソースコードは人間にとってある程度読める形の文章表現というところですが、コンピュータへの指示にはなりません。コンピュータが読めるのは 0,1 の 2 値のみですので、ソースコードをコンピュータが読める形の 2 値データに変換する必要があります。この意味で、コンパイルを翻訳ということがあります。

注 5.4　上の説明では「コンパイルして、そのあとで実行可能ファイルを実行させれば」と書きましたが、現実にはこの段階でエラー処理に時間がかかります。

1. まず、プログラム記述の違反 (C 言語の文法に違反) をコンパイラがチェックして表示します。これを修正しないとコンパイル自体が終了しません。
2. 次のエラーの可能性は、文法上のエラーではない記述間違いです。例えば乗算を除算として書いた場合、記述が文法に違反していなければコンパイルは通りますが、実行結果は正しくないでしょう。
3. もう一つのエラーの可能性はアルゴリズム自体です。これが間違っていると、コンパイルがうまく完了しても、実行すると意図しない結果が出てしまいます。文法上のエラーではない記述間違いやアルゴリズムのエラーなどはコンピュータによるチェックは無理で人間の介在が必要になります。

このような記述のエラー、アルゴリズムの内容のエラー、など様々なエラーが絡んできて、その修正に手間と時間がかかるのが一般的です。このようなエラーを**バグ** (bug: 虫) と言い、エラーの修正操作を**デバッグ** (debug: 虫とり) あるいは**デバッギング** (debugging) といいます。一つのプログラミングにかかる全体の時間の 9 割以上はデバッグにかかるという話をよく聞きます。今回のうるう年の判定は非常に簡単なプログラムですのでデバッグは殆どありませんでした。ディスプレイ上の表示形式を少し修正した程度でした。

1.6.2　プログラムの記述と動作について

以下に、図 39 のプログラムの記述や動作について少し補足しておきます。

- 大切な注意です。C 言語プログラムの中での = は等号ではなく代入を表します。= の右側部分で決まる値を = の左側の変数に代入せよ、という命令になります。24 行目で説明しますと、(10 行目で $div4$ の値は 4 と設定していますので)$A\%div4$ の値は整数除算 $A \div 4$ のあまりとなりますが、これを $r1$ という変数に代入しなさい（格納しなさい)、ということです。
- C 言語では等号は==と (=を 2 つ続けて) 表記します。&& は「かつ」を、 || は「または」をそれぞれ表す C 言語の記号です。なお、$\neq$ (等しくない) は != と表記します。
- (14 行目〜21 行目) コンピュータのディスプレイにタイトルを表示したり、西暦 A 年の A の値をキーボードから打ち込んでコンピュータ内部のメモリに格納させる操作です (ディスプレイにその操作の指示を表示します)。20 行目の

  scanf(”%d” ,&A)

  がデータ読み込みの指示です。キーボードから打ち込まれた整数を変数 A に格納しなさい、という指示です。プログラムはこの値を使ってうるう年の判定をします。なお、このような読み込みのことを**データの入力** (data input) と言うことがあります。

- (24 行目〜26 行目) 3 つの整数除算のあまり $A\%4$, $A\%100$, $A\%400$ を計算します。なお上述した通り、これらの式にある $=$ は等号ではなく、代入です。
- (29 行目と 33 行目) この 2 行はペアになっています。29 行目の

  $(\ (r1 == 0)\ \&\&\ (\ (r2! = 0)\,|\,(r3 == 0)\ )\ )$

  は判定条件

  $[\,A\%4 = 0\,]$ かつ $[\ [\,A\%100 \neq 0\,]$ または $[\,A\%400 = 0\,]\ ]$

  を表します。 29 行目の **if** と条件式および 31 行目で、この条件が成り立つときには「西暦 A 年はうる年です」とディスプレイに表示させます。33 行目の **else** と 35 行目で、この条件が成り立たないときに「西暦 A 年はうる年ではありません」とディスプレイに表示させます。
- 現在のプログラムは、A の 1 つの値に対して判定すると終了する記述になっています。もう少し使い勝手が良くなるように、1 つの値に対して判定するとすぐに終了するのではなく、判定後に次の A の値の入力を待つ形式にし、次の値が入力されると判定する、という繰り返し形に変更することも選択肢です。ただ、この変更には繰り返し操作の記述方法などの説明なども必要になりますので、もう少しプログラムというものに慣れた段階で取り上げることにします。

ちょっと一言 今回は大雑把に言えば、アルゴリズムを作成し、その正当性を確認したあとでプログラミング言語によりプログラムとして記述して、コンピュータで実行させる、という流れを概説したということになります。実際には、アルゴリズムに基づいてまずメモリの使用量や処理時間を概算して「これで大丈夫」となってからプログラム作成に進むのが通常の姿です。またほとんどの場合で、作成した当初のプログラムには考え方やアルゴリズムの間違いあるいはプログラム記述のエラーなどが含まれます。これらを除去するためデバッグに長い時間を費やすことになり、デバッグが終了してはじめてプログラムの実行に移ることになります。

プログラム作成以後をプログラミングという場合がありますが、考え方やアルゴリズムの作成も含めてプログラミングという場合もあり、これらの区別は厳密ではありません。