情報技術の基礎 1

情報技術の基礎 1

第 1 章　コンピュータ技術の基礎

1-1 データ表現と基数

(a) コンピュータと 2 進数

コンピュータが内部で扱う物理データはスイッチの ON・OFF で生じる電圧で表現される。情報は、電圧の Hi （High：通常 ＋5V）と Lo（Low：0V）で伝えられ、ディジタル回路で処理される。データ処理では、一本の配線に伝えられている Hi または Lo が最小のデータ単位となり、これを 1 ビット（bit）という。通常データは 8 ビットを単位としてまとめて処理され、これを 1 バイト（byte）という。

データは、数値として処理される場合、1 ビットを 1 桁とした 2 進数となる。また、数値としてではなく、それぞれのビットの値を真（= 1）または偽（= 0）という属性として扱う場合は論理値となる。ただし、数値・論理値の区別はあくまでも意味づけであり、コンピュータ内部では、「1, 0」の 2 進データとして処理される。

(b) 情報処理で用いる基数変換

0 と 1 からなる 2 進数はコンピュータには処理しやすいが、普段、0 から 9 までを使っている私たちにとってはなじみにくい。そこで、2 進数をよりわかりやすくするために、2 進数を 3 桁ずつにまとめて 8 進数で表したり、4 桁ずつにまとめて 16 進数で表したりする。2 進数、8 進数、16 進数と 10 進数の関連を理解し、いつでも相互に変換ができるようにすることが必要である。

(c) *m*進数と桁の重み

一般に *m* 進数というとき、*m*を基数という。たとえば 10 進数では 10 が基数である。*m* 進数で小数点の位置から左方向に数えて *n* 桁目の数字を *K* とすると、その桁の値は 10 進数では、$K \times m^{n-1}$ となり、この m^{n-1} が桁の重みである。たとえば、10 進数 385 は、次のようになる。

$385 = 3 \times 10^2 + 8 \times 10^1 + 5 \times 10^0$

その桁の値が基数になると桁上がりする。

1-2 基数変数

(a) 整数の基数変換

m 進数の整数 *R* を *k* 進数に変換するには、まず *m* 進数の演算で、*R* を *k* で割り、余りを a_1 とする。さらに、その商を *k* で割り、余りを a_2 とする。以下同様に、a_3, a_4 ……と商が 0 になるまで繰り返し、余りを、a_1 を最下位として（……$a_4a_3a_2a_1$）と並べると *k* 進数になる。

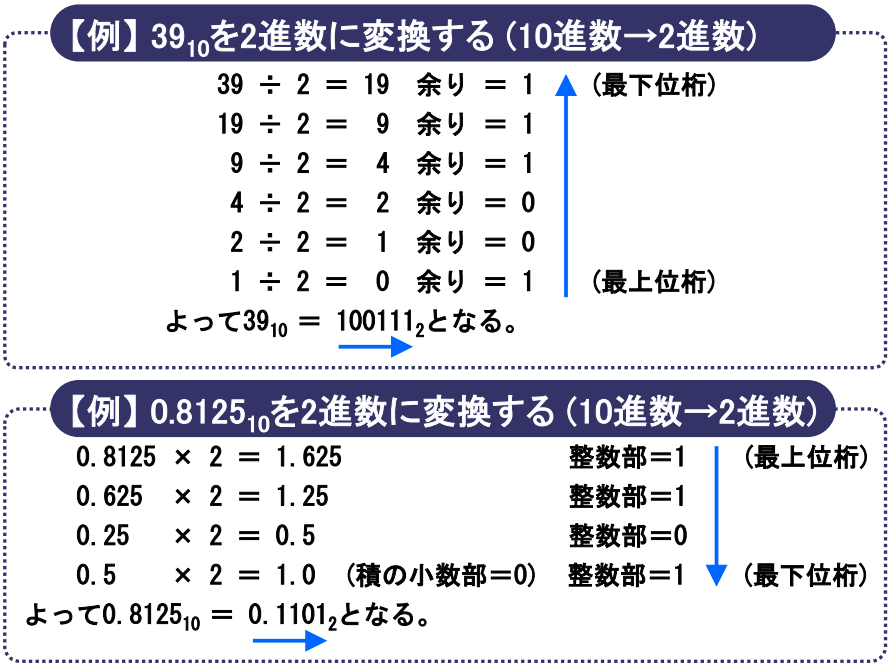

図 1　基数変換

(b) 小数の基数変換

m進数の小数（整数部を含まない）Sをk進数に変換するには、m進数の演算でSをk倍し、整数への桁上がり値をb_1とする（桁上がりがなければ0）。さらにその積の小数部のみをk倍し、整数への桁上がり値をb_2，b_3，b_4 ……と小数部が0になるまで繰り返すと、$b_1b_2b_3b_4$の数列がb_1を小数第1位（最上位桁）とするk進数の小数になる。しかし、積の小数部が0にならない場合（循環小数など）もあり、その場合は近似値で打ち切る。

1-3 いろいろな基数変換

(a) 2進・10進変換（2進数→10進数）

2進数から10進数への変換も先の例と同様にしてできるが、各桁の重みを利用したほうが簡単なので普通はこの方法で行う。n桁目の2進数の値は、2^{n-1}である。

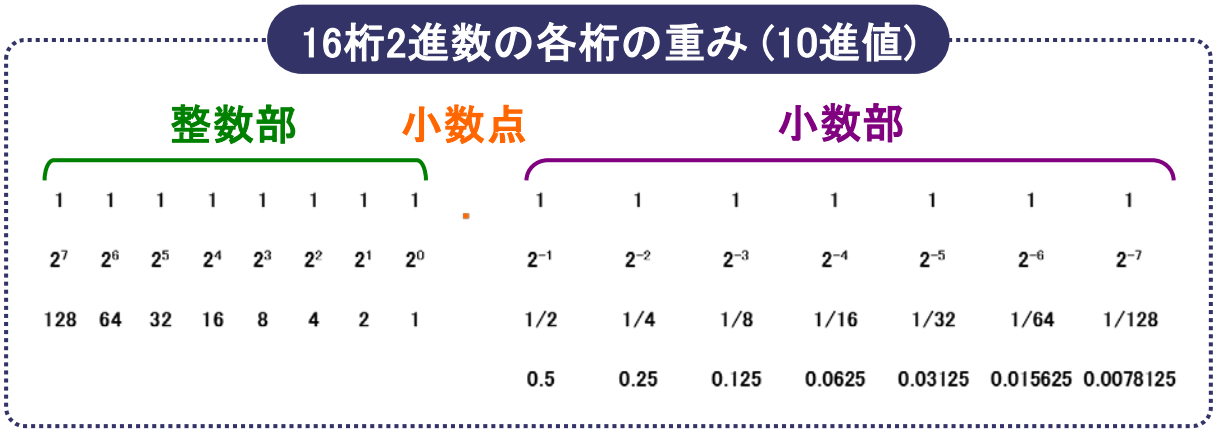

図2　2進・10進変換

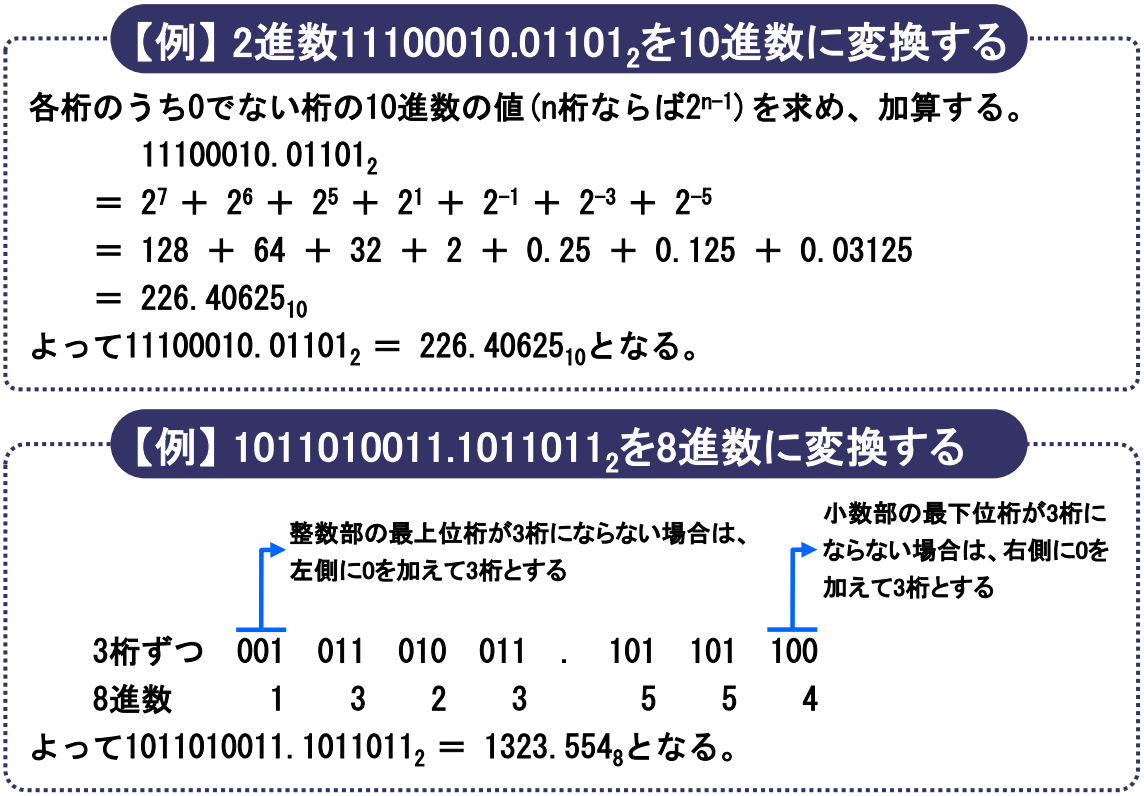

図3　いろいろな変換

(b) 2進・8進変換

① 2進数→8進数

2 進数を小数点から両側に 3 桁ずつのブロックに区切り、8 進数を求める。

② 8 進数→2 進数

8 進数の各桁を 3 桁の 2 進数に変換し、対応させる。

(c) 2 進・16 進変換

① 2 進数→16 進数

2 進数を小数点から両側に 4 桁ずつのブロックに区切り、その 10 進数を計算して 16 進数を求める。10 進数の 10～15 は 16 進数では A～F で表すことに注意すること。

② 16 進数→2 進数

16 進数の各桁を 4 桁の 2 進数に変換し対応させる。2 進数から 16 進数に変換したときのように 16 進数→10 進数→2 進数と順に計算してもよい。

【例】2471.53_8を2進数に変換する

8進数	2	4	7	1	.	5	3
2進数	010	100	111	001	.	101	011_2

よって2471.53_8 ＝ 10100111001.101011_2となる。

【例】10110100.11101101_2を16進数に変換する

4桁ずつ	1011	0100	.	1110	1101
10進数	11	4	.	14	13
16進数	B	4	.	E	D

よって10110100.11101101_2 ＝ $B4.ED_{16}$となる。

【例】$B0.3C_{16}$を2進数に変換する

16進数	B	0	.	3	C
(10進数	11	0	.	3	12)
2進数	1011	0000	.	0011	1100_2

よって$B0.3C_{16}$ ＝ 10110000.00111100_2となる。

図 4　いろいろな変換

1-4　1 の補数と 2 の補数

n 進表現された数に加算すると、一つ桁が上がって $100\cdots0_n$ となる値（たとえば、10 進数では 3 に対する 7）を n の補数または真補数という。また、加算すると、桁数は変わらずに、$(100\cdots0-1)_n$ となる値を（$n-1$）の補数または擬補数という。たとえば、10 進数を例にとると、6 の 10 の補数（真補数）は 4 であるが、9（＝10－1）の補数（擬補数）は 3 である。（$n-1$）の補数に 1 を加算すると n の補数になる。2 進数では 2 の補数と 1 の補数がそれである。

(a) 1 の補数

2 進数では足して 1 になる数は、0 には 1、1 には 0 である。したがって、1 の補数は、各桁の数の 0 と 1 を反転した値である。

【例】　1011 0010_2 の 1 の補数
　　0100 1101

(b) 2 の補数

2 の補数は加算すると 2、つまり一つ桁が上がって 10 となる値である。したがって、11 の 2 の補数は 01 である（11＋01＝100）。この関係を次の例で理解してほしい。2 の補数を求めるには、1 の補数を求めて 1 を足すとよい。

【例】　1011 0010_2 の 2 の補数
　　0100 1101 ＋ 1 ＝ 0100 1110
　　（1011 0010 ＋ 0100 1100 ＝ 1 0000 0000）

(c) 2 の補数と負の表現

2 進数では、2 の補数を使って負を表現する。コンピュータでは、限られた桁数（ビット数）以上の表現はないので、0－1 = -1 は 8 ビットの 2 進数で表現すると、0000 0000－0000 0001 = 1111 1111 となる。この 1111 1111 は 0000 0001 の 2 の補数で、最上位桁が 1 の数は負となる。

次の数は、0 前後の数を 8 桁の 2 進数で表したものである。

0000 0010　+2（-2 の 2 の補数）
0000 0001　+1（-1 の 2 の補数）
0000 0000　 0
1111 1111　-1（+1 の 2 の補数）
1111 1110　-2（+2 の 2 の補数）

次に、8 桁で表現できる最大・最小の値付近を示す（最上位は符号ビット）。

0111 1110　+126（-126 の 2 の補数）
0111 1111　+127（-127 の 2 の補数）
1000 0000　-128
1000 0001　-127（+127 の 2 の補数）
1000 0010　-126（+126 の 2 の補数）

1-5 固定小数点表現

(a) 符号なし 2 進数（符号なし整数）

1 と 0 の組合せは、4 桁では 16、8 桁では 256 通りといったように、2 進数 n 桁の組合せは 2^n 通りである。

(b) 符号付き 2 進数（符号付き整数）

符号付き 2 進数では、同じ桁数で表現できる数値の絶対値が約半分になる。しかし、前述したように負側が 1 だけ多くなる。

(c) 固定小数点表現

2 進数での固定小数点表現には、ワード単位で、最上位ビット（MSB）の左側を小数点位置とする形が使われる。最下位ビット（LSB）の右側を小数点位置とすると整数表現になる。

①符号と絶対値表現

最上位ビットを符号ビットとし、以下を絶対値で表す。わかりやすいのが特徴である。n ビットで表現できる範囲は、-2^{n-1}～2^{n-1} である。

【例】　-5（10 進数）= 1000 0101（8 ビット 2 進数）

②補数表現

1 の補数または 2 の補数を用いて表す。特に、2 の補数は加減算の区別がなく、加算のみでできるのでよく用いられる。n ビットで表現できる範囲は

1 の補数　　-2^{n-1}～2^{n-1}
2 の補数　　-2^{n-1}～$2^{n-1}-1$

③バイアス表現

2 の補数表現の負の値に、あらかじめ負の範囲の下限値をバイアスとして加算し、非負の値として表現する。元に戻すにはバイアス値を減じる。加減算が 2 の補数同様、加算のみで行え、大小の関係も見やすいのが特徴である。

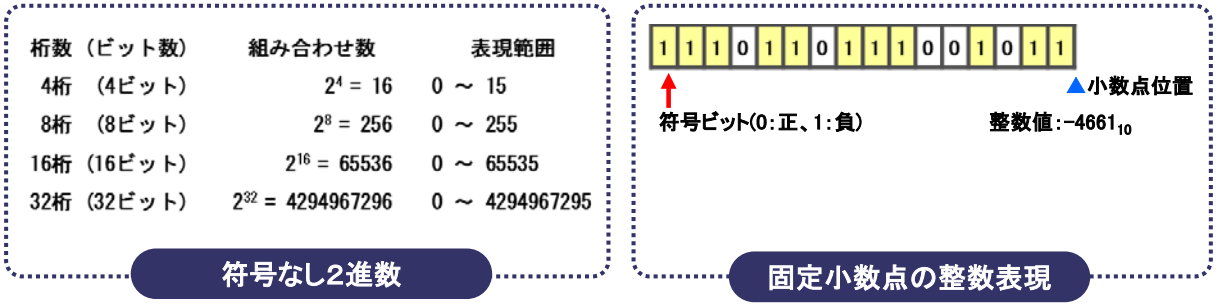

図 5　2 進数の正負、固定小数点表現

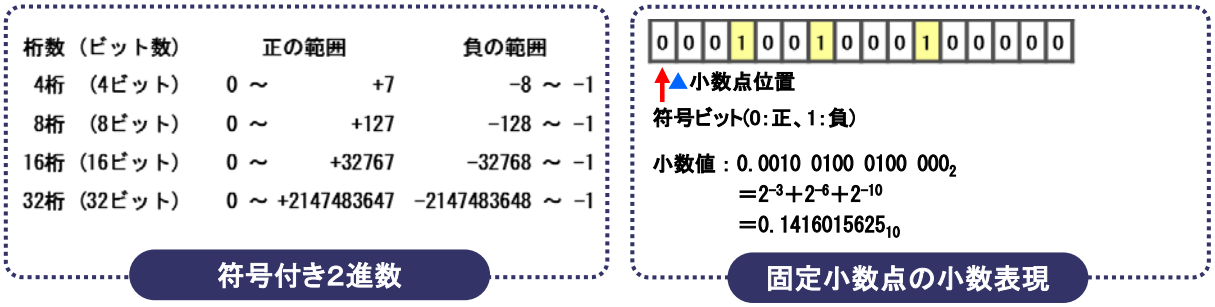

図 6　2 進数の正負、固定小数点表現

1-6　10 進数表現

(a) 2 進化 10 進数（BCD）

10 進数の各桁を 4 ビットで表したものである。153971 を BCD で表すと、次のようになる。

（BCD）　0001 0101 0011 1001 0111

（10 進数）　1　5　3　9　7

(b) ゾーン 10 進数

ゾーン 10 進数は、BCD の各桁（4 ビット）にゾーンという 4 ビットを加えて 10 進数の 1 桁 8 ビットで表したものである。最下位のゾーンには符号を入れる（+：1100, －：1101）。ゾーン 10 進数は、BCD と、標準コードの ASCII、JIS、EBCDIC などとの中間的性格を持つ。

(c) パック 10 進数

パック 10 進数は、ゾーン 10 進数からゾーンを除いたもので、10 進数 1 桁を 4 ビットで表現する。この点は BCD そのものと同じだが、最後に符号を表す 4 ビットが加わる。ゾーン 10 進数からパック 10 進数に変換することをパックするといい、その逆をアンパックするという。

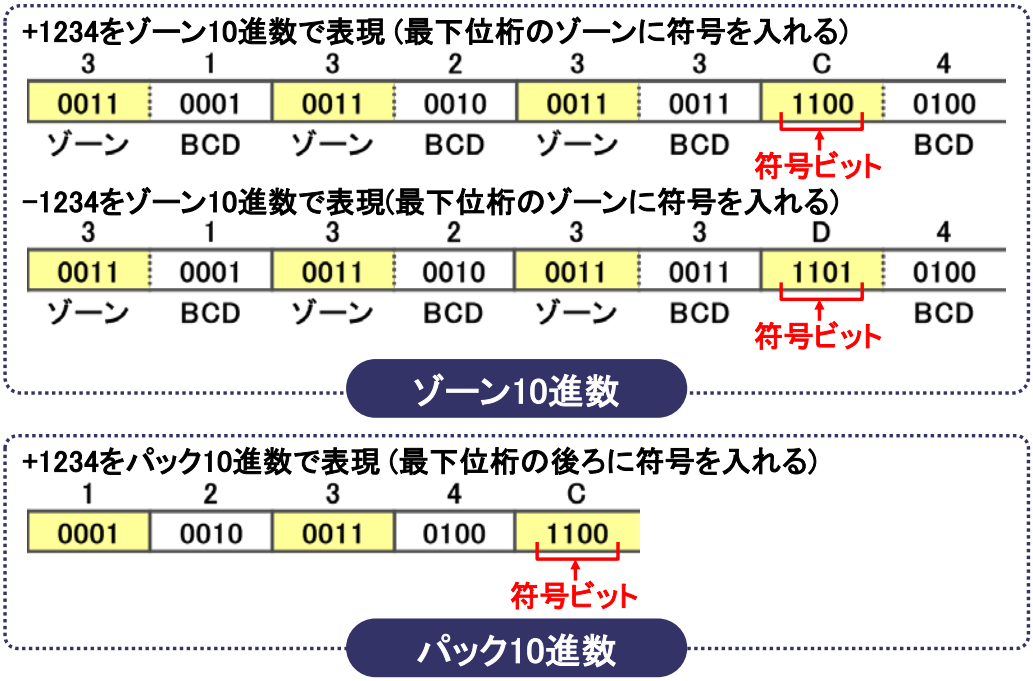

図 7　10 進数表現

1-7 数値データ

(a) 整数

整数は固定小数点の整数表現の形をとる。1 バイト型、2 バイト型、4 バイト型が用いられるが、符合なしと符号付きでは表現できる範囲が異なる。

(b) 実数

実数は浮動小数点数で表記する。4 バイトの単精度、8 バイトの倍精度が用いられる。コードサイズが大きいものほど、仮数部が大きく、精度がよくなる。

ここでは、実数演算において発生しうる誤差について述べる。

①丸め誤差

計算した結果のある桁以下を四捨五入、切上げ、切捨てなどを行ったときに生じる誤差。対策には丸めにより発生する四捨五入、切上げ、切捨てなどの状況を正確に把握して適切に使い分ける。

②情報落ち誤差（データ落ち）

絶対値が大きい値に対し、極端に小さい値を加減算すると、小さいほうの値が計算結果から欠落する誤差。　この誤差への対策としては、多数の数値の和をとる場合、小さい値から順に加算するようにする。

③桁落ち誤差

実数計算で、全体の値に対して差の小さな値を減算すると、演算結果の有効桁数が小さくなり、精度が下がる。　この誤差への対策としては、差が極端に小さい値どうしの加減算が起こらないようアルゴリズムを工夫する。

④打切り誤差

循環小数など無限小数の計算を、途中で打ち切ったために発生する誤差。前述した 10 進数の 2 進変換などで発生する。対策としては演算結果への影響が最小限となるよう、打切り点を検討する。

⑤オーバーフロー誤差

演算結果の絶対値が、コンピュータの表現できる範囲を超えたときに発生する誤差。実数演算では、仮数の有効桁数を超える桁数が発生した場合に見られる。

⑥アンダーフロー誤差

演算結果の絶対値が、コンピュータの表現できる最小値以下になるときに発生する誤差。実数演算では、仮数の絶対値が表現できる最小値以下になると発生する。

図 8　誤差

1-8 文字データ

文字データは、英語の場合はアルファベットなので、1 バイトコード（256 種類）でよいが、日本語では 2 バイトコードを用いる。

(a) ASCII コード、EBCDIC コード、JIS コード

ASCII コード（アスキーコード）はアメリカの標準キャラクタコードで、制御コード、記号、数字、アルファベット文字を 7 ビットで表現し、パリティビットの 1 ビットを加えて 1 バイトとする。ASCII コード体系はそのまま国際標準（ISO）コードに採用されている。一方、JIS コードは半角カタカナ文字に加えて、パリティなしの 8 ビットで表現し、記号類も日本向けに少し手直しされている。また、EBCDIC コードは 10 進コードによる汎用機用のコードである。

(b) 日本語の表現

漢字を中心とする日本語の表現には、ISO のコード拡張規定に基づいて定められた JIS 漢字コードが用いられる。これは、JIS コードを引き継いだ第 1 バイトと付加された第 2 バイトからなり、使用時に JIS コード（半角）と区別するために漢字コードの前と後ろにエスケープシーケンスを挿入する。

この点を改良し、第 1 バイトを JIS コードの未定義な部分に割り当てたのがシフト JIS コードで、エスケープシーケンスを挿入する必要がなく、半角、全角の混在も自由なため、広く使われている。一方、UNIX 上で多国語表現をする EUC コード、パソコンを中心に 2 バイト以上の多バイトコード体系を標準化しようとする Unicode などの国際規格化が進んでいる。

1-9 そのほかのデータ形式

コンピュータの高速化、高機能化、それに伴うマルチメディア化、またインターネットを中心としたデータ通信網の発達に対応した種々のデータ形式が標準化され、広く使われている。ここでは静止画像データ、動画データ、音声データについて述べる。

(a) 静止画像データ

静止画像データは大きく分けて、ペイント系とドロー系に分けられる。ペイント系は、静止画像をビットマップに分解し、サンプリング、量子化などのプロセスを経て、ディジタル化したもので、データ圧縮方式などにより種々のデータ形式が採用されている。

ドロー系は、画像を形成する線の情報をデータとして保存するもので、最大の特徴は、画像を拡大しても曲線や斜線にギザギザ（シャギー）が出ないことである。

代表的なペイント系静止画像データ形式

データ形式名	特徴その他
BMP形式（拡張子.bmp）	ビットマップ形式のWindows標準形式である。色数はフルカラーの16,777,216色から256色、グレースケールなど、自由に設定できるが、原則としてデータ圧縮しないため、データサイズが大きい。
TIFF形式（拡張子.tif）	画像データをイメージデータとして保存する形式で、他のOSへの移植性が考慮されている。データの保存、印刷などに広く用いられている。
JPEG形式（拡張子.jpg）	データ圧縮の程度を自由に選択できる点に特徴があり、インターネットで最も利用されている。人の目にはほとんど認識できない程度の高画質圧縮から、データサイズを数十分の一以下にできる高度圧縮までが可能であるが、その場合は当然画質が落ちる。
GIF形式（拡張子.gif）	データを圧縮して保存するデータ形式で、色数を制限したものでは圧縮効率が高い。この形式による簡易動画(GIFアニメーション)も可能で、インターネットではよく利用される。
PICT形式（拡張子.pct）	Macintoshの標準的な画像保存形式である。

図 9　静止画像データ形式

(b) 動画データ

動画は毎秒 14～30 コマを連続的に表示する必要があるため、データサイズが非常に大きく、データの圧縮は不可欠である。カラー画像の標準圧縮方式を ISO で定めたものが MPEG 方式で、CD、ハードディスクなどを対象とした MPEG1、DVD、衛星放送などを対象とした MPEG2、移動体通信用

の MPEG4 などがある。MPEG2 は高画質を目的としており、標準として 720 X 480 ドット、毎秒 30 コマがよく使われるが、これは変えることができる。また、Windows の標準規格では、AVI 形式（拡張子.avi.）が、Machintosh 標準規格では MOV 形式（拡張子.mov）が広く使われている。

(c) 音声データ

音声の波形を AD 変換（アナログ・ディジタル変換）してデータ化したもので、Windows では WAVE 形式（拡張子.wav）が用いられる。また、ISO が策定したオーディオ規格、MPEG1 Audio Layer III は音声の伝達フォーマットとして策定されたものだが、音質を劣化させることなく高い圧縮率が得られることから、通称 MP3（拡張子.mp3）の名で音声データの保存形式として普及している。

1-10 論理演算

(a) 論理値と論理演算

1 ビットの 2 進数値（1 または 0）を数値とみなさない場合、その値を論理値という。論理値 1 は「真」、0 は「偽」と定義することができる。論理値どうしの演算が論理演算である。最も基本的な論理演算として、論理積（AND、∧）、論理和（OR、∨）、論理否定（NOT、¯）、排他的論理和（XOR、⊕）がある。

(b) 論理式と真理値表

真（1）と偽（0）のすべての組合せとその演算結果を表にまとめたものを真理値表という。

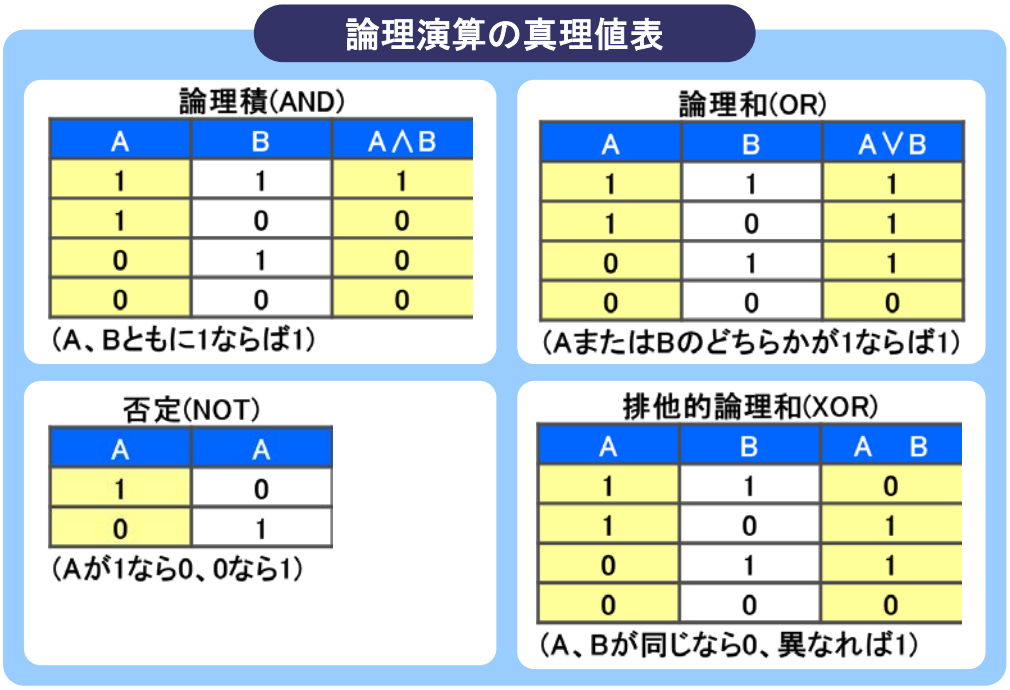

図 10　論理演算の真理値表

(c) 論理式の変形と「ド・モルガンの法則」

論理演算では 1 または 0 の 2 値のみなので、A、B、C の 2 進数について、図のような公式が成り立つ。これらの公式を使って式を簡略化することができる。

(d) 論理演算の応用

論理演算によるビット列処理の代表的なものをいくつかあげる。これらのビット演算は実際のプログラミングで多用されているので、覚えておくと便利である。

①特定ビットの消去（論理積）

数字の’6’は ASCII コードでは’00110110’であるが、上位の’0011’を消して数値 6 を取り出したい場合に用いる。

②BCD（2 進化 10 進数）を ASCII コードに変換（論理和）

①の逆の演算で、特定ビットをビット列で埋める。この演算は、2 進化 10 進数を ASCII コードに変換する際に用いられる。図の例は、マスタデータのパターンによってアルファベット、または数字に変換された例である。また、このパターンを変えることにより、特定のビットのみを 1 または 0 にすることもできる。

③2 進数またはビット列の 0 クリア（排他的論理和）

値の同じビット列どうしの排他的論理和は常に 0 である。

④ビットチェック

特定ビットが真か偽かを調べる。

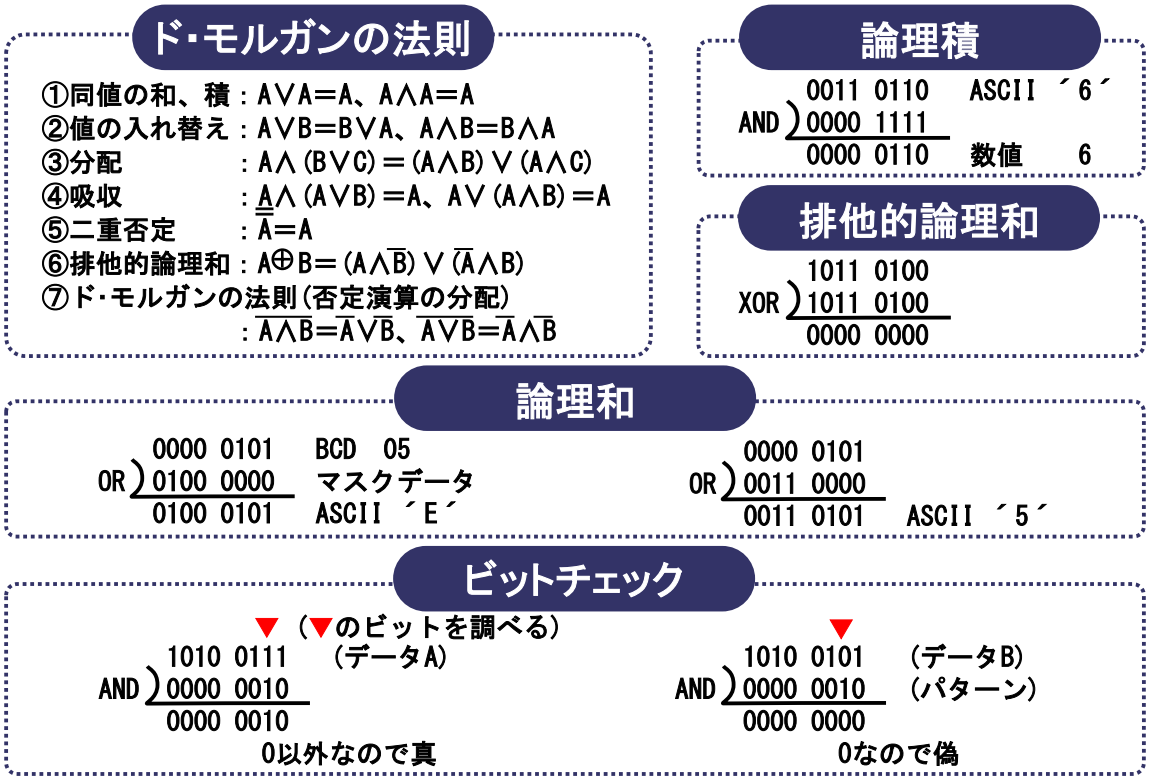

図 11　論理演算

1-11 集合

ある旅客機の乗客/搭乗員(T)が大人(X)と学生(Y)と幼児等(Z)であったとすると、T は全体集合、X、Y、Z は部分集合となる。また、X、Y、Z それぞれに男性（m）と女性(f)がいれば、全男性 Tm、全女性 Tf も部分集合となる。

部分集合 X が集合 T に属することを、次式で表す。

$$X \subset T$$

二つの集合 A と B の関係が A⊂B であり、同時に A⊃B であるとき、「集合 A と B は等しい」といい、次式で表す。

$$A = B$$

また、全体集合 A と部分集合 B の関係に、A = B の場合も含まれるときは、次式で表す。

$$A \supseteq B$$

a、*b*、*c*、*d* という個体からなる集合を A とするとき、

$$A = \{a,\ b,\ c,\ d\}$$

と表す。また、要素を一つも持たない集合を空集合といい、φ で表す。

要素 *c* が集合 A に属することを、次式で表す。

$$c \in A$$

集合の記述では集合 N がすべての整数からなることを、次式で表すこともある。

$$N = \{x \mid x \text{は整数}\}$$

1-12 符号理論の応用

(a) ハミングコードによる誤りの訂正

受信情報の誤りを検出し、訂正するためには、送受信情報の間に適当なビット差が必要になる。このビット差を与えるためのビット列を検査ビットといい、検査ビットを付加したものをハミングコードという。4 ビットの情報ビットを $x_1x_2x_3x_4$ をとして次の式によって求めた $c_1c_2c_3$ が検査ビ

ットである。

$$c_1 = x_1 + x_2 + x_3$$
$$c_2 = x_1 + x_3 + x_4$$
$$c_3 = x_2 + x_3 + x_4$$

4 ビットの情報ビットと検査ビット、情報ビットに検査ビットを付加したハミングコードを図に示す。図中で検査ビットの c_1、c_2、c_3は先に述べた検査ビットの式に基づいてつくられている。

7ビットのハミングコード

情報値	情報ビット				検査ビット			ハミングコード						
	x_1	x_2	x_3	x_4	c_1	c_2	c_3	x_1	x_2	x_3	x_4	c_1	c_2	c_3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	1	0	0	0	1	0	1	1
2	0	0	1	0	1	1	1	0	0	1	0	1	1	1
3	0	0	1	1	1	0	0	0	0	1	1	1	0	0
4	0	1	0	0	1	0	1	0	1	0	0	1	0	1
5	0	1	0	1	1	1	0	0	1	0	1	1	1	0
6	0	1	1	0	0	1	0	0	1	1	0	0	1	0
7	0	1	1	1	0	0	1	0	1	1	1	0	0	1
8	1	0	0	0	1	1	0	1	0	0	0	1	1	0
9	1	0	0	1	1	0	1	1	0	0	1	1	0	1
10	1	0	1	0	0	0	1	1	0	1	0	0	0	1
11	1	0	1	1	0	1	0	1	0	1	1	0	1	0
12	1	1	0	0	0	1	1	1	1	0	0	0	1	1
13	1	1	0	1	0	0	0	1	1	0	1	0	0	0
14	1	1	1	0	1	0	0	1	1	1	0	1	0	0
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1

図 12　ハミングコード

(b) CRC

情報と、検査のための冗長データを多項式で表し、その積を用いて誤りを検出する方式を CRC 方式という。CRC は簡単なシフト演算で誤りの検出、訂正ができることから、ハードウェアによる誤りの検出と訂正が可能であり、古くから行われている。

(c) パリティチェック

情報ビットに検査ビットを加えたデータを符号語と呼ぶ。符号の中にある 1 の個数が偶数個になるように冗長ビット（パリティビット）を与え、これを誤りの検出に使用するのがパリティチェックである。

(d) ハフマン符号とデータの圧縮

一連のデータの中で、最も出現頻度の高いパターンに最短のコードを与え、出現頻度が下がるにつれて、コードの長さを長くしていく手法をハフマン符号化という。たとえば、「東京特許許可局」をローマ字で表して

tokyotokkyokyokakyoku

という 21 字の文字列として考える。これは JIS コードで 21 バイトなのでこのままの符号長は、

21 x 8 = 168 ビット

である。ここに出現する文字を、出現頻度の高い順に並べ替えて、最も出現頻度の高い文字に 0、次が 10、以下 110、1110、11110 と符号化してハフマン符号を並べてみると

1110 10 0 110 10 1110 10 0 0 110 10 0 110 10 0 11110 0 110 10 0 111110
t o k y o t o k k y o k y o k a k y o k u

と表され、51 ビットとなる。JIS コードによりコード化した場合に比べ 30%に圧縮されている。

1-13 確率

ある試行により起きる事柄を事象という。この事象が起こる可能性を確率で表す。ある試行に

対する全事象を n、特定の事象 A の数を m とする。事象 A の起こる確率を P(A)とすると、P(A) = m/n となる。

【例題】　サイコロを振るという事象は全部で 6（全事象）ある。1～6 のどれも同様に出る可能性があるとすると、1 の目が出る（特定事象）確率は P（1 が出る）= 1/6 である。また、1～6 の中で、偶数は 2、4、6 の 3 個なので、偶数の目が出る（特定事象）確率は P（偶数が出る）= 3/6 = 1/2 となる。

(a) 和事象

二つの事象 A、B があり、A または B が起こる事象を和事象という。A の起こる確率を P (A)、B の起こる確率を P (B)とすると、事象 A または B の起こる確率 P (A∪B)は両者の確率の和で P (A∪B) = P (A) + P (B)となる。

【例題】　一つのサイコロを振ったとき、3 と 5 が同時に出ることはない。3 の出る確率、5 の出る確率ともに 1/6 なので、3 または 5 の出る確率は P（3 または 5 が出る）= 1/6 + 1/6 = 1/3 となる。

(b) 積事象

事象 A と B がともに起こることを積事象という。積事象の起こる確率 P (A∩B)は A、B それぞれの起こる確率の積で P (A∩B) = P (A) x P (B)となる。また、和事象において、A と B が同時に起こりえる場合は、同時に起こる確率を引く必要がある。

P (A∪B) = P (A) + P (B)－P (A∩B)

【例題】　サイコロを 2 回投げて、1 回目は偶数、2 回目に 3 が出る確率を求める。

【解答】　1 回目の試行で起こる事象の確率は P（偶数）= 3/6 = 1/2、2 回目の試行で起こる事象の確率は P（3）= 1/6。

偶数と 3 の出る確率は積事象なので P（偶数・3）= 1/2 x 1/6 = 1/12 となる。

(c) 余事象

事象 A の起こらないことを余事象といい A で表す。また、起こらない確率を「A の余事象の確率」という。

$$P(\bar{A}) = 1 - P(A)$$

1-14 確率分布

(a)　確率分布

ある変数により確率が決まる場合、その変数を確率変数という。確率変数を X とすると、k に対応する確率を $P(X = k)$ で表し、確率分布という。図のように確率分布をグラフで表したものを確率分布図という。

(b) 期待値と分散

確率分布の様子を数値的に表現するには期待値（平均値）と分散が用いられる。確率分布の確率変数を x_1, x_2……, x_n とし、それぞれに対応する確率を p_1, p_2……, p_n とすると、確率変数の期待値 $E(X)$は

$$E(X) = x_1 p_1 + x_2 p_2 + \cdots\cdots + x_n p_n$$

で表される。

確率分布のばらつきの度合いを示すもう一つの数値が分散で、期待値との差の 2 乗の和で表す。期待値を E、確率を p_n とすると、分散 $V(X)$ は

$$V(X) = (p_1 - E)^2 + (p_2 - E)^2 + \cdots\cdots + (p_n - E)^2$$

となる。また、分散 $V(X)$ の平方根を標準偏差といい、$\sigma(X)$ で表す。

$$\sigma(X) = \sqrt{V(X)}$$

(c) 2 項分布

コインを 6 回投げるという試行で、表（おもて）が x 回出るという事象 A の起こる確率を p とすると、確率変数 x は 0, 1, 2, ……, 6 という値をとる確率分布となる。この場合、A の起こらない（裏が出る）確率 q は 1－p となり、このような確率分布を 2 項分布という。2 項分布の成立する確率分布をグラフにすると、左右対称なグラフになる。また、一般に、一つの確率分布における確率の総和は 1 である。

(d) 正規分布と標準正規分布

①正規分布

コインやサイコロのように離散型ではなく、確率変数が連続値をとる場合の最も一般的な分布は正規分布と呼ばれ、グラフにすると中央に山のできる左右対称な曲線になる。これを正規分布曲線という。

正規分布では、確率変数が連続値で与えられるため、対応する確率値も連続値である密度関数として与えられる。連続値の確率分布では、関数値である確率値も連続値となるため、その和はグラフの面積（積分値）として与えられ、その総和は 1 である。正規分布の密度関数を表す式は次のようになる。

$$P(X)=\frac{1}{\sqrt{2\pi}\,\sigma}\,e^{-\frac{(x-m)^2}{2\sigma^2}}$$

m :　平均
e :　自然対数の底
σ^2 :　分散
σ :　標準偏差

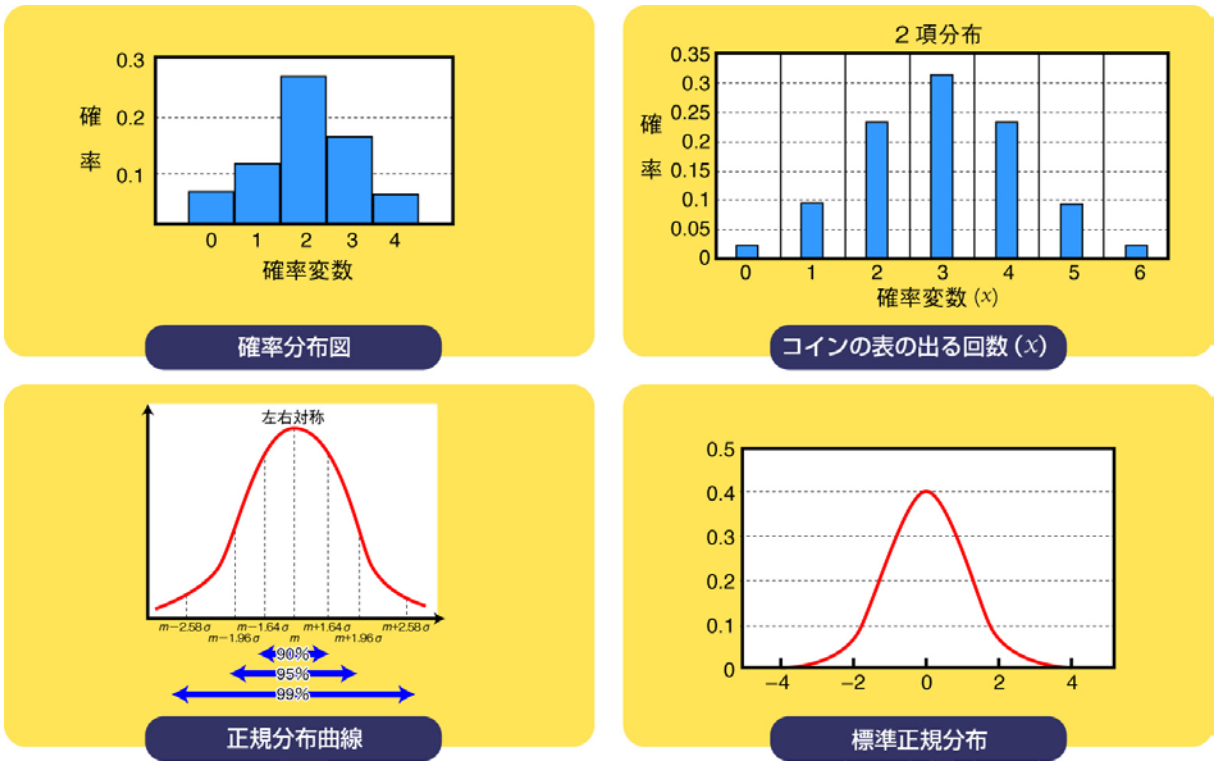

図 13　確率分布

②標準正規分布

正規分布において平均値を 0、分散・標準偏差を 1 としたものを標準正規分布という。実際のデータの正規分布を演算する場合、どうしても積分などの計算が必要だが、標準正規分布では標準的な密度関数値が積分値の数表として与えられるので計算が容易である。

実際には、実際の変数を x、平均値を m、標準偏差を σ としたとき、次式で標準正規分布の変数 u を算出する。

$$u = (x-m)/\sigma$$

(e) 一様分布

サイコロの目 1, 2, 3, ……, 6 を確率変数とする分布を考えると、各変数に対応する確率は常に 1/6 になる。このような分布を一様分布という。乱数の分布は一様分布である。

(e) ポアソン分布

ポアソン分布は、全事象の中で、事象 X の発生率を λ とすると、事象 X が n 回発生する確率 V(n) が次式で表されるような分布である。

$$V(n)=\left[\frac{\lambda^{n}}{n!}\right]e^{-\lambda}$$ （e =2.71828…：自然数対数の底）

これは、ランダムに発生する事象で、事象の発生する確率の低い試行を多数行った場合に見られる分布で、当てはまる例としては、全製品に対する不良品の発生数、一定時間の来客数などが挙げられる。λの数を増やしていくと、正規分布に近づく。

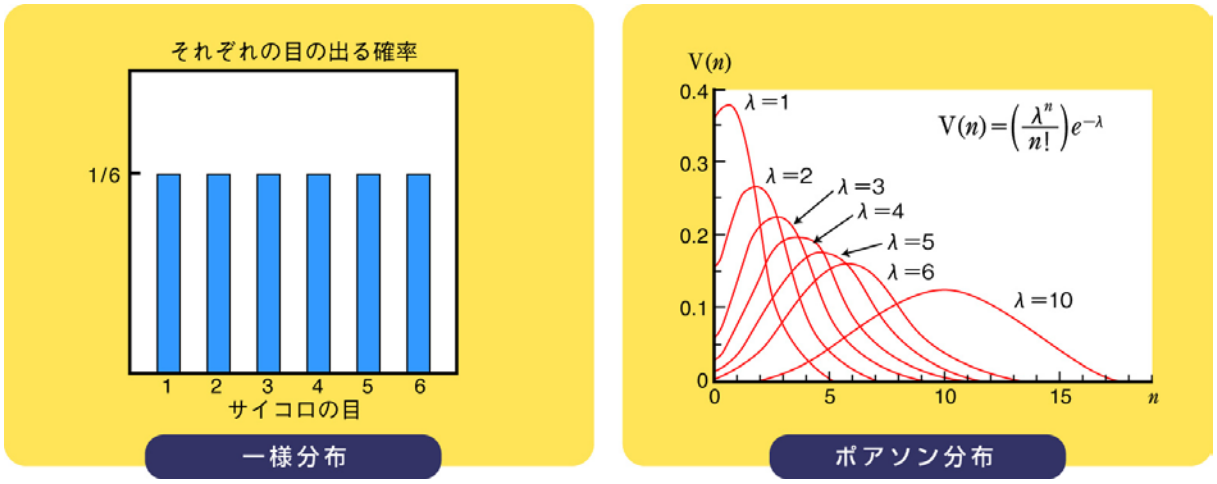

図 14　確率分布

1-15 アルゴリズムとは

(a) アルゴリズムとは

アルゴリズムは、ソフトウェアの処理実行という動的な面である「一連の実行手順」を意味する。コンピュータで物事を処理する場合、一般的には、内部ではさまざまな処理を組み合わせて実行している。この内部処理の順番によっては、無駄が発生したり、処理上のまちがいが起ったりすることもある。すなわち、事前に十分な検討を重ねてアルゴリズムを組み立てておくことは、効率的で、正しく動作するソフトウェアを設計する第一歩といえる。

たとえば、東京から大阪に行くまでの経路を考えてみよう。交通は、新幹線や飛行機、自動車などさまざまな手段があり、経路も無数に存在する。この中から、移動時間や経費的な観点で、最善となる交通手段と経路を選択することは、日常生活の中で一般的に行われていることである。コンピュータの処理も同様で、全く同じ処理結果が得られるプログラムであっても、コンピュータに負荷をかけず、より高速に、無駄なリソースを使わないようなアルゴリズムを採用することが、プログラム設計者、およびプログラマの裁量となる。したがって、処理内容の順番や論理を検討したうえで、良いアルゴリズムを採用することが重要となる。なお、良いアルゴリズムを組み立てるためには、ここで扱う基本となるアルゴリズムのパターンを定石として覚えておくとよい。

(b) 良いアルゴリズムの条件

良いアルゴリズムは、図のような条件を満たしたものである。

(c) アルゴリズムとデータ構造

データ構造とは、データがどのように記録されているか、個々のデータがどのような関係で構成されているかを表すものである。処理されるべきデータのデータ構造によって、採用されるアルゴリズムも異なる可能性がある。したがって、事前に処理を行うデータのデータ構造についても調べておく必要がある。

データ構造も、アルゴリズムと同様にいくつかの基本的なデータ構造に大別できる。それぞれのデータ構造を理解して、それらをアルゴリズムの中でどのように扱うかということを理解すると効果的である。

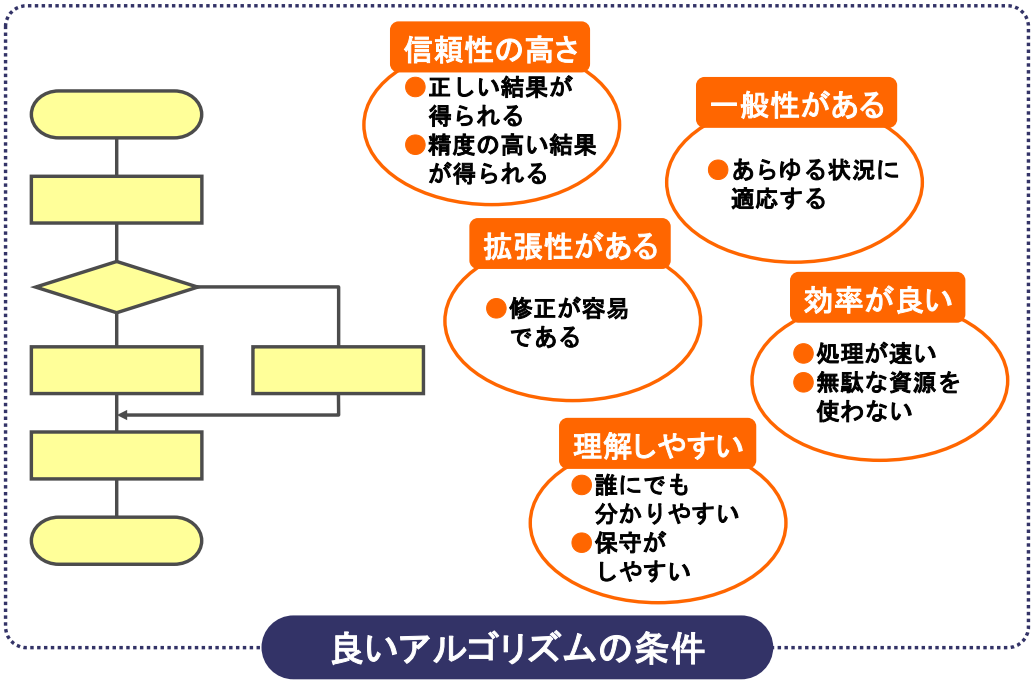

図 15　アルゴリズム

1-16 流れ図

(a) 流れ図とは

アルゴリズムを理解しやすく表記する方法が流れ図（フローチャート）である。これは、制御の流れを線で表し、動作させる装置や手続きを記号化して、その図の中に処理内容を記述することによって、視覚的にアルゴリズムを理解するものである。流れ図で用いられる記号は日本工業規格（JIS）によって定められている。

(b) 基本的な三つの制御構造

アルゴリズムには、基本となる制御構造が 3 種類ある。この制御構造を組み合わせて構成したアルゴリズムが、わかりやすく理解しやすいといわれている。基本制御構造とその流れ図は次のようになる。

①直線型（連続型）

与えられた手続きを、上から下に向かって連続して直線的に処理する。

②分岐型（選択型）

処理に伴う判定を行い、その結果によって、複数存在する処理の中から実行する処理を選択する。分岐型には、大別すると次の二つがある。

- 二分岐型（IF THEN ELSE 型）：一般的に、条件を満たすか、満たさないかという真偽による判定を行い、実行する処理を選択する。
- 多分岐型（CASE 型）：条件判定のとき、複数の分岐を取り得る。すなわち、一つの判定によって、複数の異なる処理を選択する。

③反復型（繰り返し型）

ある条件が満たされている間、あるいはある条件が満たされるまで、一定の処理を繰り返し実行する。なお、反復は繰り返し同じ処理を実行することで、ループ処理とも呼ばれる。反復型は、大別すると事前判定型と事後判定型の二つがある。両者は処理に対してループを抜ける判定位置が異なるため、アルゴリズムを考えるうえで、どちらを採用するか慎重に検討する必要がある。

- 事前判定型（DO WHILE 型）：処理を行う前に、処理を継続するかどうかを判定する。したがって、状況によっては、一度も処理を行わないままループを抜けることもあり得るのが特徴である。
- 事後判定型（REPEAT UNTIL 型）：処理を実行した後に、繰り返しの継続判定を行う。したがって、少なくとも 1 回は処理を実行することになる。

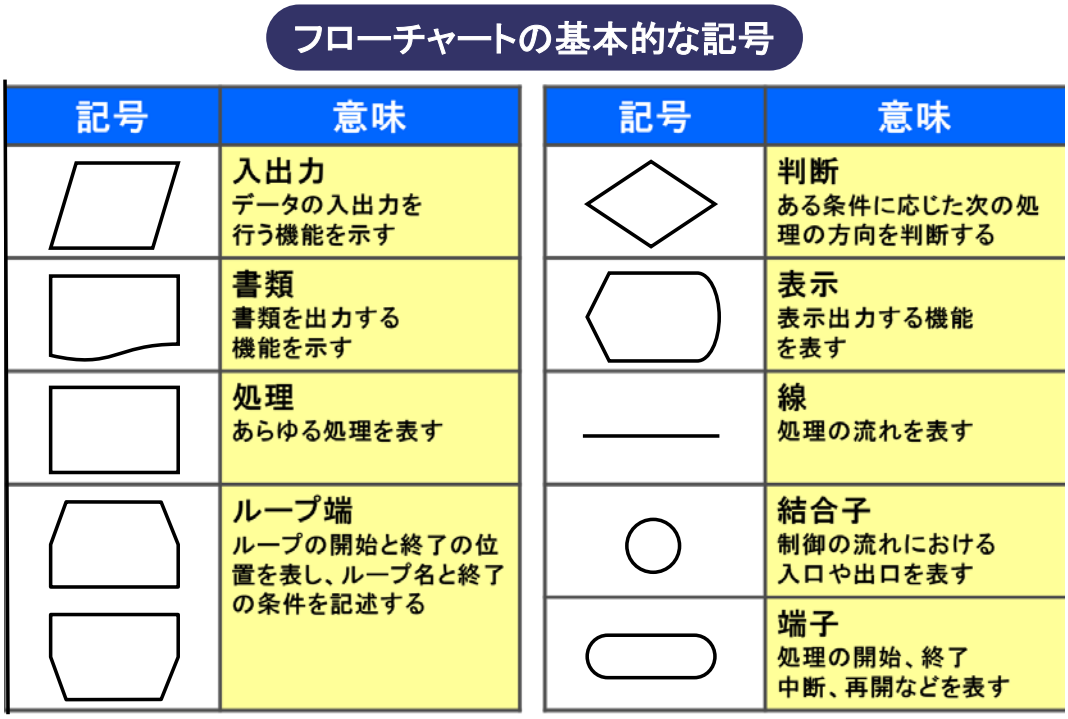

図 16　フローチャートの基本記号

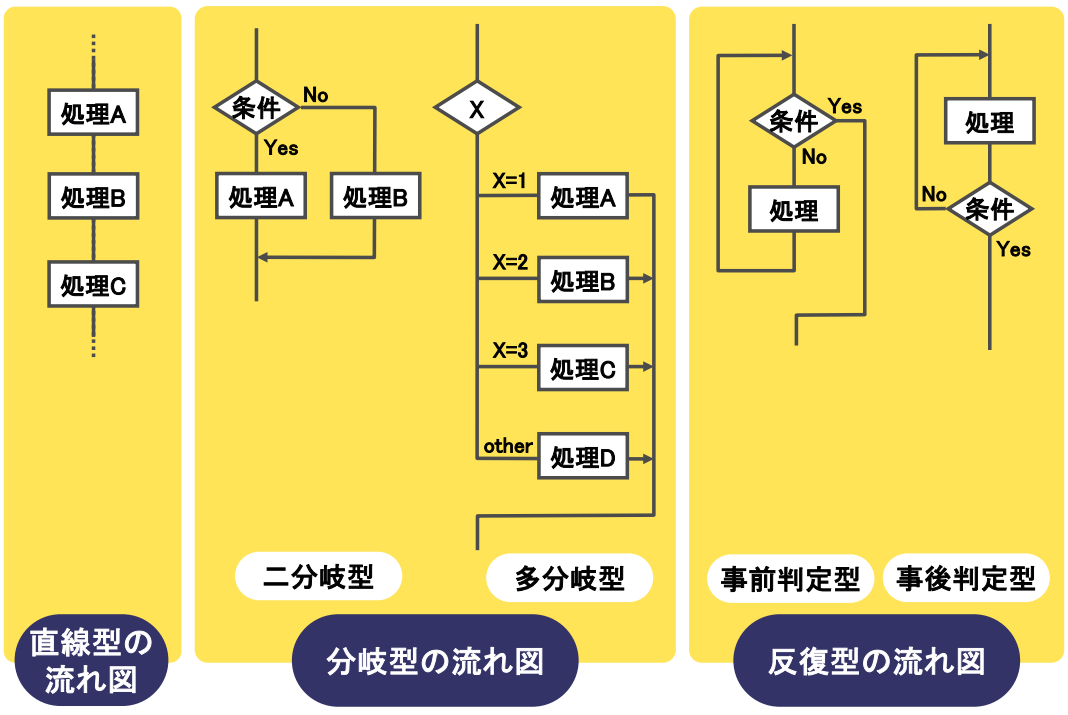

図 17　流れ図

用語・略語

BCD	Binary Coded Decimal
ASCII	American Standard Code for Information Interchange
JIS	Japan Industrial Standards：日本工業規格
EBCDIC	Extended Binary Coded Decimal Interchange Code
BMP	Bit MaP
TIFF	Tag Image File Format
JPEG	Joint Photographic Experts Group
GIF	Graphics Interchange Format
PICT	Macintosh PICT format
MPEG	Motion Picture Experts Group
BCD	Binary Coded Decimal：2 進化 10 進数
CRC	Cyclic Redundancy Check：巡回冗長検査
CPU	Central Processing Unit
ALU	Arithmetic and Logic Unit
クロックパルス	動作を合わせるために用いる周期的なパルス
PSW	Program Status Word
NMI	Non Maskable Interrupt
IRQ	Interrupt ReQuest
RAM	Random Access Memory
SRAM	Static RAM
DRAM	Dynamic RAM
ROM	Read Only Memory
マスク ROM	Masked Read Only Memory
PROM	Programmable Read Only Memory
EPROM	Erasable-Programmable Read Only Memory
フラッシュメモリ	ディジタルカメラのメモリカード。携帯情報端末のメモリ素子などを中心に急速に普及が進んでいる
MO	Magneto Optical disk：光磁気ディスク
CD	Compact Disc
CD-R	CD-Recordable：追記型光ディスク
CD-ROM	CD-Read Only Memory
CD-RW	CD-ReWritable
DVD	Digital Versatile Disk
DVD-ROM	DVD-Read Only Memory
DVD-R	DVD-Recordable
DVD-RAM	DVD-Random Access Memory
RAID	Redundant Array of Inexpensive Disks
ECC	Error Correcting Code：誤り訂正符号
OCR	Optical Character Reader：光学式文字読取り装置
OMR	Optical Mark Reader：光学式記号読取り装置
CCD	Charge Coupled Device：電荷結合素子
GUI	Graphical User Interface
LCD	Liquid Crystal Display
GP-IB	General Purpose Interface Bus
IEEE	Institute of Electrical and Electronics Engineers：米国電気電子技術者協会
デイジーチェーン	装置から装置へ次々に、いもづる式に接続すること
SCSI	Small Computer System Interface：スカジー
DTE	Data Terminal Equipment
DCE	Data Circuit terminating Equipment：回線終端装置
EIA	Electronic Industries Association：米国電子工業会
IDE	Integrated Device Electronics

USB	Universal Serial Bus
Ir-DA	Infrared Data Association
OS	Operating System：Windows や UNIX などハードウェアの管理やユーザインタフェースの提供を行うソフトウェア
CASE	Computer Aided Software Engineering
CAD	Computer Aided Design：コンピュータ支援設計
CAM	Computer Aided Manufacturing：コンピュータ支援製造
CAI	Computer Aided Instruction：コンピュータ支援教育
CMI	Computer Management Instruction：コンピュータ支援教育管理
JCL	Job Control Language
クライアント/サーバ	厳密にはサービスを提供するプログラムがサーバで、サービスを要求するプログラムをクライアントと呼んだほうが正確であるが、現在では同義として扱われる場合が多い
スプーリング	spooling：アプリケーションを実行しながら、周辺機器とのデータのやり取りをする
データベースサーバ	セキュリティ上、検索のみに限定して使われる場合もある
GUI	Graphical User Interface
メインフレーム	汎用コンピュータのこと
DSU	Digital Service Unit
RPC	Remote Procedure Call：遠隔手続き呼出しともいう
シームレス	透過的
LPC	Local Procedure Call
ANSI	American National Standards Institute
ISO	International Organization for Standardization：国際標準化機構 国際的な規格の制定を行う国際機関
23h	“h”は hexa の略で、16 進数を表す
JIS C 6220	JIS C 6220 はその後、JIS 全体のグループ変えに伴い、JIS X 0201 という規格書番号に変更されている。JIS C 6220 ・JIS X 0201 では、そこで規定している 2 種類の文字コードを”7 単位符号”および”8 単位符号”と呼んでいる
JIS C 6226	JIS C 6226 はその後、JIS 全体のグループ変えに伴い、JIS X 0208 という規格書番号に変更されている
システム外字	ユーザが定義する一般的な外字に対して、OS があらかじめ用意している外字。機種依存文字とも呼ぶ
現地化(ローカライズ)	コンピュータの世界では、主にアメリカで開発された技術や規格などを自国向けに変更することを”L10N”と、また、個別のローカライズではなく、国際的に適用する汎用のしくみとして進化させることを”I18N”と呼ぶことがある
ISO 10646	国際化時代の新たなるスタンダードとなることを象徴した規格番号といえる
オクテット	パソコンの世界では 8 ビットを 1 つのまとまりとした単位を”バイト(Byte)”と呼ぶのが通例だが、このバイトは 8 ビット以外の値を定義する際にも使われるため、厳密に 8 ビットを単位として表したいときには、”オクテット(Octet)”という言葉を使う
ISO 10646-1	正式名称は、”ISO/IEC 10646-1: 1993 Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane”
CJK 漢字統合	その後、ベトナムで使われていた漢字の文字集合も対象となったので、現在では”CJKV 統合漢字”と呼ばれることもある

引用文献:「基本情報処理技術者 標準教科書」 オーム社
引用文献:「図解でわかる文字コードのすべて」 日本実業出版社