■目次（ほげが今回の範囲）

第 1 部と第 2 部では，コンピュータが全てを 0/1 で表して動作していることを学んだ．しかし，これまでに 0/1 で表す方法を学んだのは，整数のデータと機械語の命令のみである．一方，現実のコンピュータは，文字や音声，画像といった様々な情報を処理することができる．ということは，これらの情報も 0/1 で表すことができるはずである．今回からの第 3 部では，これらの情報をどのようにして 0/1 で表すか，という問題を考える．関連して，情報の量の測り方についても学ぶ．

★ 7　情報をどのように表現するか

★ 7.1　アナログ情報とディジタル情報

情報とは何だろうか (☆ 1)．国語辞典 (☆ 2) によると，

> **じょうほう【情報】** — ある事柄に関して伝達（入手）されるデータ（の内容)。

> **データ**〔data〕— (1) 推論の基礎となる事実。(2) その事柄に関する（関して集めた）個個の事実を、記号〔＝数字・文字・符号・音声など〕で表現したもの。

だそうである．この科目としては，「データ」については (2) の方を採り，「情報」とは「ある事柄に関して伝達・入手される個個の事実を数字や文字などの記号で表現したもの」と言ってよいだろう．

コンピュータを含む情報機器は，入力装置を介して外の世界から情報を受け取り，それを処理して新たな情報に加工し (☆ 3)，出力装置を介して外の世界へ送り出すものである．情報機器が扱う情報には，温度や物体の位置のように，その値が連続的な数量で表される**アナログ情報**と，整数やキーボードのどのキーを押したかのように，離散的な（とびとびの）値で表される**ディジタル情報**とがある．

ディジタル情報は，その値を 0/1 のならびに対応付けることで，ビットパターンに変換することができる．この変換過程を**符号化**という．あるディジタル情報の取りうる値が N 通りあるならば，自然数 n を $n-1 < \log_2 N \leq n$ を満たす数として，n ビットのビットパターンを用いて符号化することができる．

> **Q1.** 英語のアルファベット 26 文字（大文字のみ考える）をビットパターンに符号化するなら最小何ビット必要か．0 から 9 の数も含めたらどうか．

これまで学んできたように，現代のコンピュータは，0/1 の切り替わりで動作し，扱うデータも命令も 0/1 のならびである．このようなコンピュータは，符号化されたディジタル情報を処理できるので，**ディジタルコンピュータ**と呼ばれる (☆ 4)．ディジタルコンピュータでアナログ情報を処理したい場合，そのままでは扱えないので，何らかの方法でディジタル情報に変換する必要がある (☆ 5)．

☆ 1)「もう「情報処理の基礎」の授業も 10 回目やっちゅうのにいまさら何言うてんねん」って感じですが (^_^;

☆ 2)「新明解国語辞典第五版」三省堂.

☆ 3) 細かく言えば，補助記憶装置に記憶しておくこともあるし，加工を加えずそのまま出力することもある.

☆ 4) 昔むかしは，電気回路等でアナログ情報をそのまま処理するアナログコンピュータというものもあった.

☆ 5) その方法については次回解説予定.

★ 7.2 文字情報の符号化

正負の整数以外のものをビットパターンに符号化してコンピュータで扱えることを具体的に示すために，文字列を符号化する方法について説明する (☆ 6)．日本語や英語などの文書の情報をディジタル化＋符号化する場合，文書を構成する各文字をビットパターンに符号化し，それを並べて表すのが一般的である．文字とビットパターンとの対応関係を**文字コード**といい，文字コードによって符号化されたディジタル情報を**テキストデータ** (☆ 7) という．

以下は，**ASCII**(☆ 8)(**アスキー**) の文字コード表である．アルファベットや数字，記号を扱うことのできる最も基本的な文字コードの規格であり，コンピュータその他の情報機器で広く用いられている．ASCII は元々米国で定められた規格であり，アルファベット 26 種類 ×2（大文字と小文字），数字 10 種類，その他の記号数十種類の，計 100 種類ほどの文字が扱えれば十分ということで，1 文字を 7 ビットで表して $2^7 = 128$ 種類の文字を指定できるように作られている．

☆ 6) 文字の大きさ，書体（フォント）などの違い，章，節，箇条書きといった文書の構造，などは考えない．

☆ 7) 例えば，C 言語のソースプログラムや電子メイルはテキストデータである．Emacs のようにテキストデータを編集するソフトウェアは，**テキストエディタ**と呼ばれる．

☆ 8) American Standard Code for Information Interchange の頭文字．

ASCII のコード表．例えば文字 'A' は 10 進数の 65 に相当するビットパターンに対応する．

0 NUL	1 SOH	2 STX	3 ETX	4 EOT	5 ENQ	6 ACK	7 BEL	8 BS	9 HT	10 LF	11 VT	12 FF	13 CR	14 SO	15 SI
16 DLE	17 DC1	18 DC2	19 DC3	20 DC4	21 NAK	22 SYN	23 ETB	24 CAN	25 EM	26 SUB	27 ESC	28 FS	29 GS	30 RS	31 US
32 SP	33 !	34 "	35 #	36 $	37 %	38 &	39 ’	40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
96 ‘	97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL

ASCII の最初の 32 文字（コード 0 から 31 まで（ここでは 10 進数で表す））と最後の 1 文字（コード 127）は**制御文字**と呼ばれるもので，文字を表示するためではなく，文字を出力する機器（プリンタ等）を制御するために用いられる．例えば，コード 10 の LF (Line Feed) は改行を表す．また，コード 32 の SP (Space) は空白文字（スペース）である．一方，コード 33 から 126 までは**印字可能文字**と呼ばれ，英数字や記号に対応している．

上述のように ASCII は本来 7 ビットの文字コードであるが，現代のコンピュータは 8 ビット単位でビットパターンを扱うものがほとんどであるため，最上位に 0 を付加して 8 ビットのコードとみなすことも多い．さらに，8 ビットで最上位ビットが 1 となる値（符号なし 10 進数で 128 から 255 に相当）も使用するように ASCII を拡張した文字コード体系も広く使用されている (☆ 9)．

Q2. 以下は，上述のようにして ASCII を 8 ビットにした文字コードによって，ある文字列を符号化したものである．元の文字列を答えなさい (☆ 10)．

01001000 01001111 01000111 01000101

☆ 9) 例えば，ヨーロッパで広く用いられてきた 8 ビット文字コードの規格 ISO/IEC 8859 の中の一つ ISO/IEC 8859-1 では，コード 241 に'á', 251 に'ë' 等が割り当てられており，ドイツ語やスペイン語に対応している．日本で用いられてきた同様の規格 JIS X 0201 では，同様にしてカタカナを符号化している（いわゆる「半角カナ」）．実は JIS X 0201 では，コード 92 が '¥', 126 が '‾' となっており，ASCII と違っている．

☆ 10) 見やすくするため 8 ビットごとに空白を入れてある．

C 言語では，ASCII を利用して文字情報を扱うことができる．以下は，そのようなプログラムの例である．

char.c

```
 1  #include <stdio.h>
 2
 3  int main(void)
 4  {
 5    char c;  // char 型は「8 ビット符号あり整数」
 6    int i;   //  ==> 0 以上の数（127 まで）だけ使って文字を表す
 7
 8    // 変数 c は文字 A を表す数から 1 ずつ大きくなっていく
 9    for(c = 'A'; c < 'A' + 26; c++){
10      // %d は 10 進整数として，%c は文字として扱う
11      printf("code %d corresponds to '%c'\n", c, c);
12    }
13    printf("-----\n");
14
15    while(1){
16      i = getchar(); // getchar() は 1 文字読み込む関数
17      c = (char)i;   // 戻り値が int 型なので char 型に変換
18      printf("code %d corresponds to '%c'\n", c, c);
19    }
20
21    return 0;
22  }
```

実行結果

```
$ ./a.out
code 65 corresponds to 'A'
code 66 corresponds to 'B'
code 67 corresponds to 'C'
code 68 corresponds to 'D'
code 69 corresponds to 'E'
code 70 corresponds to 'F'
code 71 corresponds to 'G'
code 72 corresponds to 'H'
     :
code 88 corresponds to 'X'
code 89 corresponds to 'Y'
code 90 corresponds to 'Z'
-----
HOGE!     <== HOGE!と入力して Enter
code 72 corresponds to 'H'
code 79 corresponds to 'O'
code 71 corresponds to 'G'
code 69 corresponds to 'E'
code 33 corresponds to '!'
code 10 corresponds to '
'          LF が出力された ↑
```

ASCII やその拡張はたかだか 8 ビットで 1 文字を表すものであるため，漢字のようにたくさんの種類の文字を符号化することはできない．そこで，日本語などのために定められた文字コードの規格では，より多くのビットを使うようになっている．以下は，日本語の文字コード規格としてよく使われてきたものである．

JIS コード JIS（日本工業規格）で定められたもの．国際規格化されており，その名称 (ISO-2022-JP) で呼ばれることもある．

Shift_JIS Microsoft Windows などで広く使われてきた文字コード．

日本語 EUC EUC(Extended Unix Code) という，UNIX(☆ 11) で使われてきた文字コードの日本語版．EUC-JP とも．

近年では，世界中のあらゆる文字を共通の文字コード体系で表すことを目指した **Unicode（ユニコード）**もよく使われる．Unicode には符号化の方式がいくつかあるが，日本で一般的な PC の環境では，**UTF-8** が使われることが多い (☆ 12)

異なる文字コードでは，同じ文字でも異なる符号が割り当てられているのが普通であるし，ある文字コードに存在する文字が別のコードには存在しないこともある．テキストデータをやりとりするときは，使用する文字コードに注意が必要である (☆ 13)．

☆ 11) コンピュータの OS（オペレーティングシステム）の一種．Linux や Mac OS X は UNIX 系の OS である．

☆ 12) 龍大計算機室の Linux 環境では基本的に UTF-8 を用いている．

☆ 13) 受け手が送り手の使ったのと違う文字コードで解釈しようとすると，いわゆる「文字化け」が起こる．

★ 7.3 情報・データの量とバイト

ディジタル情報は 0/1 のならびに符号化できるから，個々の情報の量を，その符号のならびに必要なビット数で測ることができる．この量を**データ量**という (☆ 14)．★ 7.2 でも述べたように，現代のコンピュータは 8 ビットを最小単位として情報を扱うように作られたものがほとんどであるため，データ量を測る際にも 8 ビットをひとかたまりとする**バイト (byte)**(☆ 15) という単位を用いることが多い．単位の記号としては B または byte を用いる．1B は 8bit に等しい．

☆ 14) 情報の量だから「情報量」と呼べばよい，と思うかもしれないが，実は「**情報量**」は「ある情報が本質的にどの程度の情報を持つか」の尺度として使われる用語なので，この授業では別の語を使います．「情報量」については，上の学年の授業で登場するかも．

Q3. ほげおくんは，友達 256 人のそれぞれに，英数字で 1 から 16 文字のアダ名をつけている．この友達全員のアダ名をコンピュータで扱うため，英数字 1 文字を 1 バイトで符号化する文字コードを用い，16 文字未満のアダ名も 16 文字の長さの文字列で表すことにする．全員分のアダ名のデータ量は何 B か．また，それは何 KiB か（KiB については次節参照）．

☆ 15) 本来は「そのコンピュータで英数字など 1 文字分を表すのに用いるビット数」で定められるものであり，昔むかしのコンピュータでは，6,7,9 ビットなど様々な場合があった．しかし，現代のコンピュータの世界ではほぼ 1 バイトは 8 ビットと考えて差し支えない．バイト単位で測ったデータ量は，その情報が英数字でおよそ何文字に相当するかを表すといえる．

★ 7.4 単位の接頭辞

この章の内容だけに関係する話ではないのだが，コンピュータの話で様々なところに登場する単位の接頭辞について，まとめておく．単位の接頭辞（接頭語ともいう）とは，単位の前に付けて大きな量や小さな量を表すための記号のことである．右の表に，国際単位系 (☆ 16) で定められた接頭辞（SI 接頭辞という）の一部を示す．表からわかるように，m（メートル）の 1000 倍の km や 1000 分の 1 の mm をつくる k や m は，このような接頭辞の一種である．

一般に用いる数量はほとんど 10 進法で表されるため，SI 接頭辞も 10 の整数乗で定められている．一方，コンピュータ関連の分野では 2 進法をよく用いるため，2 の整数乗を乗数とする場合がある．★ 7.3 で説明したデータ量の場合，単位にバイトを用いることが多い．このとき，$2^{10} = 1024$ が $10^3 = 1000$ と近いことから，1kB = 1024B のように，SI 接頭辞を使いながらも 2^{10} や 2^{20} を乗数とすることが慣習とされてきた．現在でも，コンピュータのメモリ容量はこの流儀で表すことが多い．一方，同じ記憶装置でもハードディスクドライブ等の補助記憶装置の容量は，10 のべき乗を乗数として表すことが多い．

SI 接頭辞

名前	記号	乗数
ペタ (peta)	P	10^{15}
テラ (tera)	T	10^{12}
ギガ (giga)	G	$10^9 = 1000\ 000\ 000$
メガ (mega)	M	$10^6 = 1000\ 000$
キロ (kilo)	k	$10^3 = 1000$
		$10^0 = 1$
ミリ (milli)	m	$10^{-3} = 0.001$
マイクロ (micro)	μ	$10^{-6} = 0.000\ 001$
ナノ (nano)	n	$10^{-9} = 0.000\ 000\ 001$
ピコ (pico)	p	10^{-12}
フェムト (femto)	f	10^{-15}

2 進接頭辞

名前	記号	乗数
ペビ (pebi)	Pi	2^{50}
テビ (tebi)	Ti	2^{40}
ギビ (gibi)	Gi	$2^{30} = 1073741824$
メビ (mebi)	Mi	$2^{20} = 1048576$
キビ (kibi)	Ki	$2^{10} = 1024$

kibi = kilo binary

☆ 16) 国際的に取り決められた単位の体系．時間，長さ，質量等の基本単位やそれらを組み合わせた単位を定めている．

このような乗数の違いは混乱のもと（$2^{40} = 1099511627776$ だから，これを乗数とした 1TB は，10^{12} を乗数とした 1TB より 1 割近く大きい）なので，上記に示す 2 進接頭辞 (☆ 17) が新たに定められた．しかし，あまり普及していない．

☆ 17) IEC（International Electrotechnical Commission，国際電気標準会議）によって 15 年ほど前に定められたもの．

Q4. ウェブでいまどきの PC のクロック周波数，メモリ容量，補助記憶装置の容量を調べよう（メーカーの直販サイトやいろんな通販サイトを覗いてみたらよいだろう）．それぞれ，どの接頭辞が主に使われているだろうか．