『プログラマーのための コンピュータ入門』
読者のための付録

―上記書購入者のみお読みいただけます―

付録 A では、n 進法、およびビットとバイトの基礎について説明します。
付録 B では、プログラムのコンパイルと実行について説明します。
付録 C では、1 バイト文字の文字コードについて説明します。

A n 進法、ビットとバイト

本書では、2 進法・10 進法・16 進法という言葉が何度も出てきました。ここで、それらについてまとめておきます。これらは総称して「n 進法」とも呼ばれる数を表記する方法です。n 進法で数を表現したものが n 進数で、こちらの用語のほうが馴染みが深いかも知れませんが、本書では原則として「n 進法」を使用しています。

第 10 章（「n 進法と整数」、本書 P.244）でも述べましたが、

```
int a;        /* int型（整数型）の変数aの宣言 */
a = 30;       /* 変数aに30を代入 */
```

という（C や Java の）コードがあったとき、「変数 a には 10 進数の 30 が代入される」と表現すると正確ではありません。

```
a = 30;       /* 10進法で表記された「30」を変数aに代入 */
a = 0x1e;     /* 16進法で表記された「1e」を変数aに代入 */
a = 036;      /* 8進法で表記された「36」を変数aに代入 */
```

これらのコードはすべて同じ値（2 進法で表した数、つまり 2 進数の「11110」）が a に代入されます。10 進数の 30 が代入されたり、16 進数の 1e が代入されたりするわけではありません。変数の内部では一般に 2 進法が使われています[*1]ので、これら 3 行はすべて同じ意味になるわけです。そこで本書では、原則として「n 進数」という言葉は使わずに「n 進法」で統一しました。

さて、10 進法で表記した数「123」は以下のように書けます。

$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

同じ 123 という値を「2 の何乗」を使った式にすると、以下のようになります。

$123 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

このようにすると、10 進法の「123」は 2 進法では「1111011」であることがわかります。この 2 進法の 1 桁のことをビットと呼びます。

10 進法と 2 進法の変換を人間が、特に暗算で行うのはかなりたいへんで

*1 言語やコンピュータによっては、10 進数をそのまま取り扱うことが可能な場合もあります。

す[*2]。そこで、プログラマ電卓という n 進法を簡単に変換できる電卓が古くからあり、ダンプを追うときなどに重宝したものです。いまは OS やスマホのアプリでもプログラマ電卓と同様のことができます。

コンピュータ内部のデータでは一般に 2 進法が使われているわけですが、この 2 進法のデータを人間が扱う（表示したり、入力したり、計算したりする）のは、桁数が多くなり、また数字として 0 と 1 しか使われないため、たいへん見づらく厄介です。人間は（たまたま両手の指が 10 本だったため）いにしえより 10 進法を使うことに慣れています。しかし、コンピュータが扱う 2 進法と、人間にとって自然な 10 進法の変換は、電卓などを使ったとしても、面倒くさいものです。そこで、コンピュータ内部のデータを表記するときには 2 進法をそのまま使うのではなく、2 進法の数桁（数ビット）をひとまとめにして表現することにします。まとめる単位は 3 桁（3 ビット）または 4 桁（4 ビット）が一般的です。

ここで例を挙げます。2 進法で表記された数「1111011」を右から 3 ビットずつに区切ります。3 ビットで表現できる値は 0 ～ 7 ですから、区切った部分をそれぞれ 0 ～ 7 の数に置き換えます（図 A-1）。これは、2 進法で表記された数を 8 進法に変換していることになります。同様に 4 ビットずつに区切れば 16 進法に変換することができます[*3]（図 A-2）。

このように、8 進法や 16 進法の表記は 2 進法表記との間で簡単に変換ができ、また 2 進法の「桁が多くなりすぎる」「0 と 1 ばかりで見づらい」という欠点を避けることができるため、よく使われます。なお、現在のコンピュータは 1 バイトが 8 ビットであるため、16 進法を用いれば、1 バイトが 16 進法 2 桁でちょうど表現できます。このため、16 進法がよく使われ、8 進法はあまり使われなくなりました[*4]。

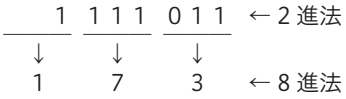
A

*2 この世界に長く生きていると、ある程度は覚えていたりもしますが、それでも限界があります。

*3 16 進法では、10 以上の数字として a ～ f または A ～ F のアルファベットを使います。これらのアルファベットは小文字を使う場合も大文字を使う場合もあります。C や Java では小文字を使うことが多いため、本書では原則として小文字で表記しました。

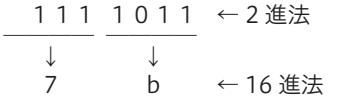
*4 過去には、1 バイトが 6 ビットや 9 ビットのコンピュータもあり、そのようなコンピュータでは 8 進法のほうが都合がよかったのです。

図 A-1　2 進法→ 8 進法の変換



2 進法	8 進法	10 進法
000	0	0
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7

図 A-2　2 進法→ 16 進法の変換



2 進法	10 進法	16 進法
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

2 進法	10 進法	16 進法
1000	8	8
1001	9	9
1010	10	a
1011	11	b
1100	12	c
1101	13	d
1110	14	e
1111	15	f

B プログラムのコンパイルと実行

あるプログラミング言語でソースコードを書き、それを実行する場合、その言語の言語処理系が必要となります。インタープリタを使用する場合は、そのインタープリタが、動かそうとしているコンピュータの OS 上にインストールされていれば、それで実行可能です。

いっぽう、コンパイラを使用する場合には、そのコンパイラをはじめ、その他多くのプログラム・ライブラリなどが必要となります。

さらにいま実用的なプログラムを作成するような場合には、その開発を容易にするために IDE（統合開発環境、Integrated Development Environment）が一般的に利用されます。IDE は、コーディング、コンパイル、実行とデバッグ、GUI アプリケーションの画面インターフェース等の設計、作成されたコードのバージョン管理など、多くの機能を統合しています。さらに複数プログラマからなるチームによる開発支援など、多くの機能を持っています。第 1 章で出てきた Xcode（図 1-1、本書 P.2）も IDE です。ほかに有名な IDE としては、Eclipse、Microsoft Visual Studio などが挙げられます。

IDE の多くの機能は、ある程度以上の規模のプログラムを作成する上では、現代ではなくてはならない仕組みと言えます。ただ、プログラミング言語の勉強をしてみたい、ちょっと簡単なプログラムを作ってみたい、というような用途では、機能が豊富すぎる、IDE ごとに操作方法が異なっている、多くの便利機能が逆に勉強の妨げになることもある、など初心者には逆に厄介なこともあります（もちろん、学校や会社等で特定の IDE を使っていることが前提の場合は、それを利用することも悪いことではありません）。

ここでは、IDE のような大きな仕組みを利用せずに、プログラミングの勉強等ができる方法をいくつか挙げておきます。使用するコンピュータは PC（Windows）とします。

C、C++、アセンブリ言語

まず C や C++ のコンパイラ、またはアセンブラの場合です。本書で使用しているコンパイラは gcc（GNU Compiler Collection）です。これは UNIX 系の OS（Linux も含む）で利用可能なコンパイラです。Windows では動作しません。しかし、以下のいずれかの方法で動作させることが可能です。

① 仮想化ソフトウェアを利用

VMware Workstation Player 等の仮想化ソフトウェアを導入し、その上で仮想マシンを起動し Linux などの OS をインストール

② WSL（Windows Subsystem for Linux）を利用

Windows 10 には WSL という Windows 上に Linux をインストールする機能があり、それを利用して Linux を実行

③ Cygwin または MinGW を利用

Windows 上に UNIX のシステムコールを利用できる仕組みで、この上で gcc を動作可能

これらのうち、最も容易に gcc を動かすことができるのが③です。プログラミング言語の勉強をしたい、等の用途であれば③の方法が最も簡単です。①や②は、gcc を動かす前に Linux の知識が必要となりますので、ハードルはそれなりに高いと言えます。

Cygwin・MinGW は、以下のサイトよりインストーラをダウンロードできます。

https://www.cygwin.com/（Cygwin）
http://www.mingw.org/ （MinGW）

ポイント

Cygwin も MinGW も、基本的な仕組みは同じですが、Cygwin が UNIX 環境全般を Windows 上に再現する方向性なのに対して、MinGW は言語処理系などの開発環境に特化していて、どちらかと言えば小さめの仕組みです。

たとえば Cygwin のインストーラを起動すると、どのようなソフトウェア（パッケージ）をインストールするか尋ねられますので、gcc 等の開発環境を選択しインストールします。すると gcc.exe 等のプログラムが指定したディレクトリにインストールされます。その後、インストールされたプログラムのありかを OS に知らせるために、いわゆる「パスを通す」設定を行います。Windows 10 では以下のように操作します。

STEP 1 スタートメニューを右クリックし「システム」を選択する。
STEP 2 右上にある「システム情報」をクリックする。
STEP 3 左側の「システムの詳細設定」をクリックする。

STEP 4 「環境変数」ボタンをクリックする。

STEP 5 変数「Path」を選択し、「編集」ボタンをクリックする。

STEP 6 「新規」ボタンをクリックし、gcc.exe がインストールされたディレクトリのパス名（例：C:¥cygwin64¥bin）を入力する。

パスを通す設定が完了すると、gcc がコマンドプロンプトから利用できるようになります（**リスト B-1**）。

リスト B-1：Cygwin 上で gcc コマンドを実行した例

```
C:¥Users¥lepton>gcc --version
gcc (GCC) 7.4.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.


C:¥Users¥lepton>
```

Java

Java の実行環境は JRE（Java Runtime Environment）と呼ばれ、JavaVM 等が含まれています。JRE があれば Java のクラスファイルを実行することが可能です。しかし、これには Java コンパイラ等の開発環境は含まれていません。開発環境は JDK（Java Development Kit）と呼ばれ、いくつかの種類があります。いま無料で Java のプログラミングの勉強を行う用途ならば、OpenJDK を使用するのがいいでしょう。

OpenJDK は以下のサイトよりダウンロードできます。

https://openjdk.java.net/

Windows 用は ZIP ファイルになっていますので、展開して適当な場所にコピーします。その後、前述した「パスを通す」作業を行うことにより、Java の開発環境が使用可能になります（**リスト B-2**）。

リスト B-2：javac、java コマンドを実行した例

```
C:¥Users¥lepton>javac -version
javac 13.0.1
```

```
C:¥Users¥lepton>java -version
openjdk version "13.0.1" 2019-10-15
OpenJDK Runtime Environment (build 13.0.1+9)
OpenJDK 64-Bit Server VM (build 13.0.1+9, mixed mode, sharing)

C:¥Users¥lepton>
```

C 1バイト文字の文字コード

コンピュータ上では、文字に番号を付けて、文字を数値として取り扱います。これを文字コードと呼びますが、世の中にはさまざまな種類の文字コードがあることは、本文中でも解説しました。ここでは、英数字・記号と制御文字を表現できる文字コードである ASCII と、JIS（の英数字）のコードを表に示します。これらは、いずれも 7 ビットで表現できる文字コードです。ASCII はもともとアメリカ規格協会（ANSI）によって定められた文字コードで、JIS 7 ビットコードはそれとほぼ同じものです。違いは、**表 C-1** で網掛けされている部分だけです。

なお、7 ビットで表すことができる文字の数は、$2^7 = 128$ ですから 128 種類ということになります。英数字やいくつかの記号や制御文字だけならば 128 個で収まりますが、それ以外の文字も取り扱おうとすると 7 ビットでは収まりません。JIS では、JIS X 0201 という規格で、上記の英数字・記号のほかに、カタカナについても定めています。規格によれば、カタカナについても 7 ビットが必要で、JIS X 0201 の英数字・記号とカタカナを同時に扱う場合には 8 ビット（$2^8 = 256$）必要になります。

表 C-1　1 バイト文字の文字コード

文字コード		文字	
10 進	16 進	JIS	ASCII
0	00	NUL	NUL
1	01	SOH	SOH
2	02	STX	STX
3	03	ETX	ETX
4	04	EOT	EOT
5	05	ENQ	ENQ
6	06	ACK	ACK
7	07	BEL	BEL
8	08	BS	BS
9	09	HT	HT

文字コード		文字	
10 進	16 進	JIS	ASCII
10	0A	NL	NL
11	0B	VT	VT
12	0C	FF	FF
13	0D	CR	CR
14	0E	SO	SO
15	0F	SI	SI
16	10	DLE	DLE
17	11	DC1	DC1
18	12	DC2	DC2
19	13	DC3	DC3
20	14	DC4	DC4
21	15	NAK	NAK
22	16	SYN	SYN
23	17	ETB	ETB
24	18	CAN	CAN
25	19	EM	EM
26	1A	SUB	SUB
27	1B	ESC	ESC
28	1C	FS	FS
29	1D	GS	GS
30	1E	RS	RS
31	1F	US	US
32	20	SP	SP
33	21	!	!
34	22	"	"
35	23	#	#
36	24	$	$
37	25	%	%

文字コード		文字	
10 進	16 進	JIS	ASCII
38	26	&	&
39	27	'	'
40	28	(	(
41	29	)	)
42	2A	*	*
43	2B	+	+
44	2C	,	,
45	2D	-	-
46	2E	.	.
47	2F	/	/
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	:	:
59	3B	;	;
60	3C	<	<
61	3D	=	=
62	3E	>	>
63	3F	?	?
64	40	@	@
65	41	A	A

文字コード		文字	
10 進	16 進	JIS	ASCII
66	42	B	B
67	43	C	C
68	44	D	D
69	45	E	E
70	46	F	F
71	47	G	G
72	48	H	H
73	49	I	I
74	4A	J	J
75	4B	K	K
76	4C	L	L
77	4D	M	M
78	4E	N	N
79	4F	O	O
80	50	P	P
81	51	Q	Q
82	52	R	R
83	53	S	S
84	54	T	T
85	55	U	U
86	56	V	V
87	57	W	W
88	58	X	X
89	59	Y	Y
90	5A	Z	Z
91	5B	[	[
92	5C	¥	\
93	5D	]	]

文字コード		文字	
10 進	16 進	JIS	ASCII
94	5E	^	^
95	5F	_	_
96	60	`	`
97	61	a	a
98	62	b	b
99	63	c	c
100	64	d	d
101	65	e	e
102	66	f	f
103	67	g	g
104	68	h	h
105	69	i	i
106	6A	j	j
107	6B	k	k
108	6C	l	l
109	6D	m	m
110	6E	n	n
111	6F	o	o
112	70	p	p
113	71	q	q
114	72	r	r
115	73	s	s
116	74	t	t
117	75	u	u
118	76	v	v
119	77	w	w
120	78	x	x
121	79	y	y

文字コード		文字	
10 進	16 進	JIS	ASCII
122	7A	z	z
123	7B	{	{
124	7C	|	|
125	7D	}	}
126	7E	‾	~
127	7F	DEL	DEL

※表中の「SP」は空白文字、色文字は通常の文字ではなく制御文字

『プログラマーのためのコンピュータ入門』
（Lepton 著、2020 年 6 月 16 日、株式会社オーム社発行）読者のための付録
