古くから使われている文字集合

関連：pp.40-41

ASCII
JIS X 0201

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000 00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL	VT	NP	CR	SO	SI
0001 10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010 20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011 30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100 40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101 50	P	Q	R	S	T	U	V	W	X	Y	Z	[				
0110 60	`	a	b	c	d	e	f	g	h	i	j	k				
0111 70	p	q	r	s	t	u	v	w	x	y	z	{				

これだけなら8bit(1Byte)で足りる。実は，**7bit**で十分（128文字以下）

問い：表を用いて **Kobe** を，16進数と2進数で表そう（ノートに）。

問い：表を用いて **Kobe** を，16進数と2進数で表そう。
答：

K	4B	0100 1011
o	6F	0110 1111
b	62	0110 0010
e	65	0110 0101

古くから使われている文字集合

関連：pp.40-41

ASCII
JIS X 0201

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000 00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL	VT	NP	CR	SO	SI
0001 10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010 20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011 30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100 40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101 50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110 60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111 70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
1000 80																
1001 90																
1010 A0		。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
1011 B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
1100 C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
1101 D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
1110 E0																
1111 F0																

これらを加えるなら，**8bit**（1Byte)必要。（256文字以下）

補足：ISO646（現ISO/IEC646）に基づく文字割当例

コード番号	0x23	0x24	0x40	0x5B	0x5C	0x5D	0x5E	0x60	0x7B	0x7C	0x7D	0x7E
ASCII	#	$	@	[	\	]	^	`	{	|	}	~
日本	#	$	@	[	¥	]	^	`	{	|	}	‾
イギリス	£	$	@	[	\	]	^	`	{	|	}	~
ドイツ	#	$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
フランス	£	$	à	°	ç	§	^	µ	é	ù	è	¨

文字のディジタル表現：基本

関連：pp.40-41

コンピュータにおける文字の処理

基本は **1対1対応**

文字を入力するとコンピュータは，
⇒ 規則に従って，文字と**ビット列**（0,1の並び）を**1対1対応**させる。
⇒ データ保存は，ビット列（0,1の並び）の状態で。
⇒ 画面や印刷物に，ビット列に対応した文字の形(ﾌｫﾝﾄ)を表示させる。

よく使われるビット列の長さは7，8，16（24,32等もあり）

1バイト文字（半角文字） 英字や数字やよく使う記号は7ビット（128種類）で十分。
カナを使うには1バイト必要（**半角カタカナ**を追加！）。

2バイト文字（全角文字） 日本語をすべて使用するには2バイト（65,536種類）で安心。
（重複するが英字・数字・記号・カタカナも含めて）

この後，日本語全ての表現方法を学習する。

文字のディジタル表現：学習内容の一部

関連：pp.40-41

- 文字集合：JIS規格等「人が理解するための一覧表」が複数種類存在する。
- 文字コード体系：コンピュータは数値しか扱えないので，「文字とビット列との対応表」も必要。

※「文字コード体系」を作るために，まず「文字集合」を定義しておく。
また「どのように文字にビット列を対応させるか」の検討も必要。

知っておくべき文字コード体系の種類

① ASCII （情報交換用米国標準コード1963） 7ビット
② JIS （ISO-2022-JP メールで使われる） 8,16ビット(1,2バイト)
③Shift_JIS （Microsoft社，Apple社が使用） 8,16ビット(1,2バイト)
④ EUC-JP （Extended Unix Code：UNIX,Linuxで使用） 8,16ビット(1,2バイト)
⑤ UTF-** （世界各国の言語に対応） 3バイト以上使用される場合有

文字のディジタル表現：用語の定義

関連：pp.40-41

よく使われる専門用語の定義

① 文字集合（character set）：
文字の一覧。**符号化文字集合**（coded character set）等ともいう。

② 文字符号化方式（character encoding scheme）：
文字集合①に含まれる各文字に1対1で対応づけた記号を，データ列（**ビット列**）に変換する方法（規則）。

③ 文字コード（character code）：
文字符号化方式②で各文字に対応したビット列。

④ 文字コード体系（character encoding system）：
文字コードの一覧。単に**文字コード**と呼ぶことも多い（紛らわしい）。

⑤ エンコーディング（符号化処理）：
文字等のデータをビット列に置き換える処理。
逆に元に戻すことを，**デコーディング**という。

文字コード系 とも呼ばれる。

補足：紛らわしい専門用語の解説

関連：pp.40-41

①文字集合（符号化文字集合）　③文字コード
②文字符号化方式　④文字コード体系，等

A:人間用の符号	B:文字例	C:コンピュータ用符号（=数値）
1-4-73	ら	1000 0010 1110 0111 (82E7)
1-4-74	り	1000 0010 1110 1000 (82E8)
1-4-75	る	1000 0010 1110 1001 (82E9)

- JIS規格等，「A:とB:の対応表」が複数種類存在する。
  「①文字集合」とは，これら「A:とB:の対応表」のこと。
- コンピュータは数値(ビット列)しか扱えないから「B:とC:の対応表」も必要。
  「④文字コード体系」とは，この「B:とC:の対応表」のこと。
- ④を作る際の（すなわち数値化するための）規則が②であり，②の規則に沿ってできた文字ごとの数値(ビット列)が③である。

文字コード体系の例

関連：pp.40-41

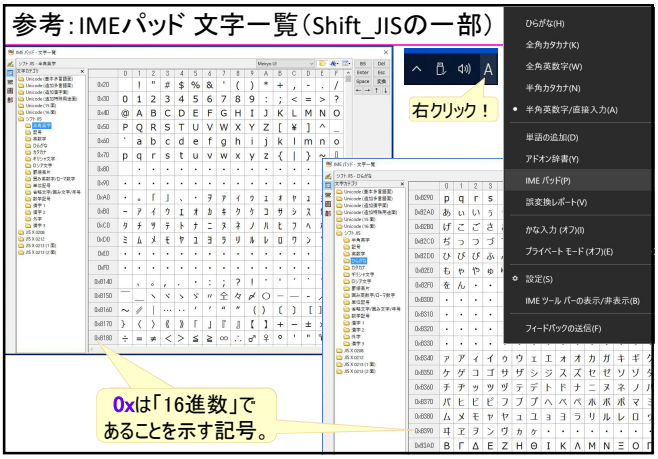
（符号化）文字集合 (Coded) character set	文字符号化方式 character encoding scheme	文字コード体系（文字コード） character encoding system, character code
ASCII	なし	**ASCII** 7bit
JIS X 0201 & JIS X 0208 等	ISO-2022-JP	**JIS(漢字)コード** 1,2Byte(7bit表現)
	Shift_JIS	**Shift_JISコード** 1,2Byte
ISO-2022-JP：半角のカタカナ含まず	EUC	**EUC-JPコード** 1,2Byte
Unicode	UTF-8, UTF-8N	**UTF-8** ASCII部分：1Byte 他：2-6Byte **Web**で使用
ここでは，N,BE,LE等を覚えなくてよい。UCSという表現も見かけるが,さらに複雑になるので気にしなくてよい。	UTF-16BE, UTF-16LE	**UTF-16** 2Byte or 4Byte
	UTF-32BE, UTF-32LE	**UTF-32** 4Byte固定長

文字集合の例(変更の頻度は著しい) 覚えなくてよいが理解を！

関連：pp.40-41

名称・規格番号	通称・俗称	文字総数	備考
ASCII		128	1963年制定
JIS X 0201	ANKコード	159	1969年制定
JIS X 0208	JIS78(旧JIS)	6802	1978年
	JIS83(新JIS)	6877	1983年
	JIS90	6879	1990年
	97JIS	6879	1997年 第1水準・第2水準
JIS X 0212	補助漢字	6067	1990年
JIS X 0213	JIS2000	11223	2000年 第3水準・第4水準
	JIS2004	11233	2004年
Unicode		136690	1991年制定，頻繁に改良

参考：IMEパッド 文字一覧（Shift_JISの一部）



参考：文字のデザイン（表示・印刷）

関連：pp.184-185

グリフ　各文字の形（デザイン）　フォント　グリフの一覧

セリフ　サンセリフ　文字の装飾による分類
https://www.indetail.co.jp/blog/170127/

等幅フォント　文字幅が均一化（統一）されている。
ノンプロポーショナルフォントとも呼ばれる。

プロポーショナルフォント　文字によって文字幅が異なる

ビットマップフォント　点の配置関係でグリフを決定する
https://seiai.ed.jp/sys/text/cs/chp02/c02a140.html

アウトラインフォント　輪郭線を数式にしてその都度計算してグリフを決定する

pt（ポイント）　文字サイズの単位　1pt ≒ 約0.35mm
https://www.well-corp.jp/solution/%E3%83%9D%E3%82%A4%E3%83%B3%E3%83%88/

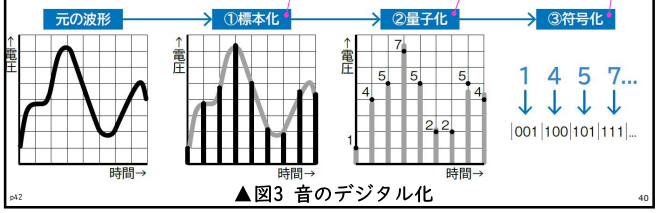
2 音のデジタル化

縦横共に一定間隔。サンプル以外は全て破棄。

貴重なサンプルを改竄してしまえ！
⇒整数にする。

整数⇒2進数

音は以下の手順でデジタル化される。
■標本化(サンプリング) 一定の時間間隔で波の高さを取り出す
■量子化 取り出した波の高さに近い段階値を読み取る
■符号化 量子化された数値を2進法で表す
あらかじめ標本化の間隔・量子化の段階を決めておく



▲図3 音のデジタル化

画像のディジタル表現の基本

関連：pp.44-45

●色数は多いほど，現実に近づけることが可能
●ペン先は細いほど，微細な表現が可能

大事なキーワード
◆ **階調**・・・・・・・色を何段階で表せるか？
◆ **解像度**・・・・・どれほど細かく描けるか？

思い出せ，音のディジタル表現！　それが次の発想につながる

◆**音のディジタル表現**：**波の高さ**を段階値（整数）にする ⇒ 桁を揃えた2進数にする ⇒ **一列**に並べる ・・・・ 並んだ**0**と**1**の個数がデータ量。

◆**画像のディジタル表現**：点の**色を何段階か**（整数）にする ⇒ 桁を揃えた2進数にする ⇒ それらを全て（**点の個数**）並べる！・・・・というワンパターンの発想。

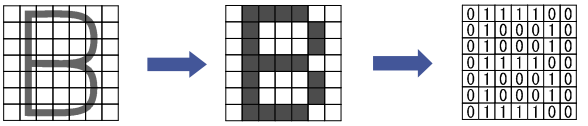
画像のディジタル表現の手順

関連：pp.44-45

標本化　画素に分割し，濃淡（標本値）を取り出す。
量子化　標本値を，段階（階調）の整数値に近似する。
符号化　0と1に置き換える。

例
（白と黒で）

・標本化　格子状に分割。
・量子化　黒か白か（２段階），判定する。
・符号化　例えば，**黒**を**1**に，**白**を**0**にする。

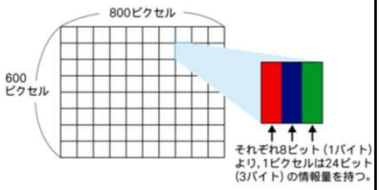


0111100 0100010 0100010 0111100 0100010 0100010 0111100

画像と文字のデータ量比較

関連：pp.44-45

例：800×600ピクセルのフルカラー画像のデータ量について考えてみよう。



（計算を簡単にするために，接頭辞は1024ではなく1000を使って解いてみよう）

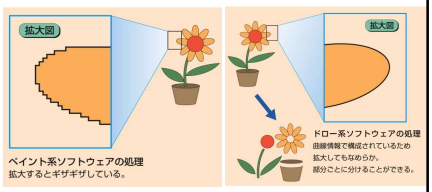
① 容量1.44MBのフロッピーディスク（FD）に，800×600ピクセルの（上例の）画像は，約何枚くらい保存できると考えられるか。
② FDに，1ページあたり40字×18行＝720字の日本語（2バイト文字）が印刷された文庫本の文字情報は，約何ページ保存できると考えられるか。
③ コンピュータにとって，どちらの処理がしんどそう？？

画像処理（ペイント系・ドロー系）

関連：pp.44-45

図形・画像の扱い方：**2種類**

フォントの種類，覚えている？



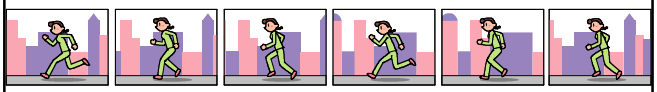
	ペイント系（ピクセルグラフィクス）	ドロー系（ベクターグラフィクス）
処理方法	図形を点の集合で表現（**ラスター表現，ラスター形式**）	図形を形状データ(座標・直線・曲線等)で表現（**ベクター表現，ベクター形式**）
拡大	拡大するとギザギザ	拡大してもなめらか
用途	写真等，グラデーションや微妙なニュアンスの表現に適する。	イラスト・設計図等に適し，変形が容易。
特徴等		図形を部品に分割できる

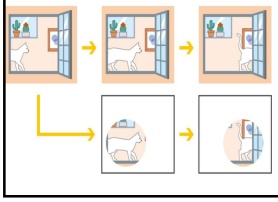
動画のしくみ

関連：pp.44-45

少しずつ変化した静止画を次々と見せる。

フレームレート：1秒間に表示させる静止画（フレームという）の枚数
（単位**fps**：frames per second）





●動画の原理：**残像効果**を利用
（パラパラまんがと同じ）
●テレビ：毎秒30枚（30fps 又は60fps）
●映画：毎秒24枚（24fps）
●データ量を減らす工夫が必要
キーフレーム⇒差分⇒差分⇒・・・キーフレーム⇒差分⇒・・・

データの容量と圧縮・解凍

関連：pp.42-45

アナログに近い状態でディジタル化したい ⇒ データ量が非常に増加！

データの**圧縮**：ファイルのデータ量（ファイルサイズ）を小さくすること。

可逆圧縮	非可逆圧縮
●元データに戻せる	●元データに（完全には）戻せない
●例：ランレングス圧縮 run length encoding	●例：音声… 聞こえない周波数の音をカット 大きな音に隠れた小さな音をカット
	●例：画像…色の違いを間引く
	●例：動画…変化した部分（**差分**）だけを記録

データの**解凍**（**展開・伸張**）：使用するためにもとの状態に戻すこと。