情報処理演習
(8)配列その2
文字列

知能システム学　准教授
万 偉偉（ワン ウェイウェイ）

課題提出の締め切りについて

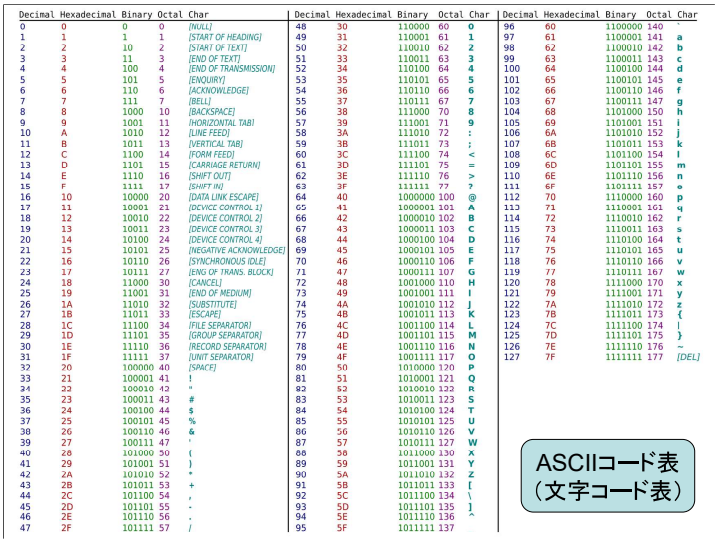
- 毎回授業の始まりまで　厳守
  - 1：30pm
  - システムが自動的に計測しますので、1分過ぎても遅延として取り扱われます．注意してください．

課題の問題

- マクロの定義
  - 大文字にすること
  - 関数内ではなく、#includeの前にすること
- 関数の宣言
  - #defineの後、main関数の前、必ず関数を宣言すること
- 多数の条件
  - 二つ以上の条件がある場合&&や||で接続する

文字列とは

- 文字が並んだもの
  - "ABC"　→ 'A', 'B', 'C' の3つの文字が順に並ぶ
  - C言語では「文字の配列」として表現される
- 文字とは
  - それぞれの文字に番号が振られている
  - 例えば 'A' は 65(0x41), 'B' は 66
  - '0' は 0 ではない
    （文字 '0' は 0x30 = 48 で表される）



Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[END OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

ASCIIコード表
（文字コード表）

確かめてみる

シングルクォート

```
#include <stdio.h>

int main(void) {
   int a;

   a = 'A';
   printf("A is %d¥n", a);

   return 0;
}
```

- 整数の変数 a に'A' を代入している
- 結果はA is 65 (%xなら16進数)
- ということは
  a = 'A'
  とするのは
  a = 65;
  とするのと同じ.

文字を表す型

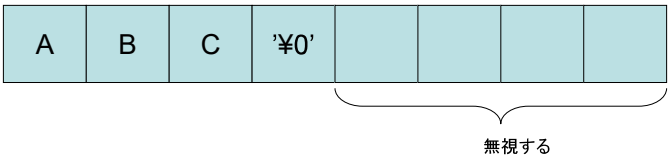
- 文字（漢字，ひらがな除く）は合計数十個
  - アルファベット（大文字・小文字），数字，記号
  - 8bit (256通り) で表すことが出来る
- 文字を表す型: char（character の略）
- 整数を表す型の一種（int などの仲間）
  - 通常，char 型は 8bit
  - 通常，int は 32bit (16bit の機械・マイコン系も)
- 文字コードを得る方法
  - 'a' のようにシングルクォートで挟む
    char ch = 'a';

文字列

- 文字の配列として表現する
  - char str[10];
    str[0] が1文字目，str[1] が2文字目，・・・
  - 文字列は "ABC" のようにダブルクォートで表現
- 最後に終端記号（ターミネータ，'¥0'）を置く
  - char str[3] = "AB"; の場合
    str[0] の中身は 'A'　str[1] は 'B'　str[2] は '¥0'
  - '¥0' = 0 であって，'0' ではない（'0' は 48）

終端記号（ターミネータ）

char str[8] = ”ABC”;



- ターミネータが来るまでが文字列とする
  - ターミネータ以降は無視する
- 利点
  - 文字列の長さを別の変数に管理する必要がない

文字列の初期化

- 配列を直接，文字列で初期化できる
  - char s[] = "ABC";
    char s[] = { 'A', 'B', 'C', '¥0' }; と同じ
    要素数は3ではなく4になることに注意
- 要素数を省略することが出来る

文字列の表示

- printf では %s で表示することが出来る

```
char str[] = "ABC";
printf ("string is %s¥n", str);
```

- ひと文字は %c で表示することが出来る

```
printf ("character of 48 is %c¥n", 48);
printf ("character of str[0] is %c¥n", str[0]);
```

制御文字

- 改行記号 '¥n'
  - '¥n' (=10) は画面に出力されると改行する
- 終端文字（ターミネータ）'¥0'
  - 文字列の終端を表す
- その他多くの制御文字がある
- 制御文字は単一の文字としても，
  文字列中でも利用できる
  - '¥n'
  - "ABC¥n"

プログラム例

```
#include <stdio.h>
int main( void )
{
char moji[] = "Moji1";
printf( "String = %s¥n",
moji );
return 0;
}
```

```
#include <stdio.h>
int main( void )
{
char moji[6];
moji[0] = 'M';
moji[1] = 'o';
moji[2] = 'j';
moji[3] = 'i';
moji[4] = '1';
moji[5] = '¥0';
printf( "String = %s¥n", moji );
return 0;
}
```

左のプログラムと右のプログラムは等価

getchar( )

- キーボードから一文字だけ読み込む関数
- 呼び出すごとに次の文字を返してくれる
- 終了時は EOF (End Of File, -1）を返す
  - getchar () は int 型を返す.
- EOF はファイル（ストリーム）の終端で発生する
  - キーボード入力時は ctrl を押しながら D を押す

getchar( )

- getchar () が実行されると，キーボードからの入力を待つ．それまでプログラムは次の命令を実行するのを待つ．
- キーボードからの入力はリターンキーを押すまでは，プログラムに伝わらない．
- 入力された文字は，入力ストリームに保存される．リターン（改行）も保存される．
- getchar () は，入力ストリーム中の一つの文字を読み込む．
- 連続してgetchar () を呼び出すと，最初に呼び出したgetchar () だけが，キーボード入力待ちを行い，次に呼び出したgetchar () は，入力ストリームから文字を読み込みに行く．この動作は入力ストリームが空（EOF）になるまで行われる．入力ストリームが空（EOF）になれば，再度キーボード入力待ちを行う．

getchar()プログラム例

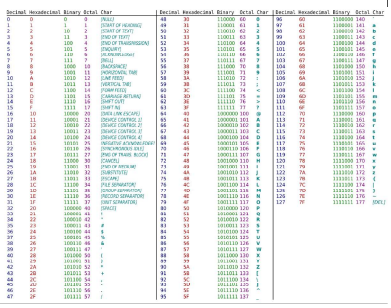
```
#include <stdio.h>

int main(void)
{
  int c1,c2,d;

    c1 = getchar();
    c2 = getchar();
    printf("¥n");
    if(c1==EOF)
      printf("c1 EOF¥n");
    if(c2==EOF)
      printf("c2 EOF¥n");
    printf("Input char 1: %x¥n", c1);
    printf("Input char 2: %x¥n", c2);
    return 0;
}
```

実行例1：

```
H↵
Input char 1:
Input char 2:
```

実行例2：

```
HB↵
Input char 1:
Input char 2:
```



CRとLF

- パソコンによって，改行のコードは違う
  - Windows : CR LF (0d 0a)
  - Unix : LF 0a
  - Mac : CR 0d
- CRとLFとは？
  - CR: Carriage Return（行頭に戻る ）
  - LF: Line Feed (行を送る)

元はラインプリンタのヘッドの制御から来ている
ラインプリンタでは，ヘッドを行頭に戻してから，行を送っていた

cast

- 型を明示的に変換する
  - (double), (int) のように型名を括弧で囲む

```
int a; double b; char c;      /* 変数の宣言*/
b = (double) a;               /* 実数型に変換*/
a = (int) b;                  /* 整数型に変換*/
c = (char) a;                 /* 文字型に変換*/
```

（キャスト；鋳造する，型にはめるの意味）

注：実際には，キャスト演算子はなくても，自動で型変換してくれるが，ある方が，プログラムのミスを防ぐことができる

文字列操作関数

- 文字列を取り扱う関数が用意されている
  - str・・・という名称の関数群
  - strcat : 2つの文字列を接続する
  - strcmp : 文字列を比較する
    - 同じなら 0，違う場合は辞書順に -1 または 1
  - strcpy : 文字列をコピーする
  - strlen : 文字列の長さを返す
- 文字列長に制限のあるバージョンもある
  - strncmp, strncat など