

# 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。

## 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	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	(NULL)	48	30	110000	60	0	96	60	1100000	140	
1	1	1	1	(START OF HEADING)	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	(START OF TEXT)	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	(END OF TEXT)	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	(END OF TRANSMISSION)	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	(ENQUIRY)	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	(ACKNOWLEDGE)	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	(BELL)	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	(BACKSPACE)	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	(HORIZONTAL TAB)	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	(LINE FEED)	58	3A	111010	72	10	106	6A	1101010	152	j
11	B	1011	13	(VERTICAL TAB)	59	3B	111011	73	11	107	6B	1101011	153	k
12	C	1100	14	(FORM FEED)	60	3C	111100	74	12	108	6C	1101100	154	l
13	D	1101	15	(CARRIAGE RETURN)	61	3D	111101	75	13	109	6D	1101101	155	m
14	E	1110	16	(SHIFT OUT)	62	3E	111110	76	14	110	6E	1101110	156	n
15	F	1111	17	(SHIFT IN)	63	3F	111111	77	15	111	6F	1101111	157	o
16	10	10000	20	(DATA LINK ESCAPE)	64	40	1000000	100	16	112	70	1110000	160	p
17	11	10001	21	(DEVICE CONTROL 1)	65	41	1000001	101	17	113	71	1110001	161	q
18	12	10010	22	(DEVICE CONTROL 2)	66	42	1000010	102	18	114	72	1110010	162	r
19	13	10011	23	(DEVICE CONTROL 3)	67	43	1000011	103	19	115	73	1110011	163	s
20	14	10100	24	(DEVICE CONTROL 4)	68	44	1000100	104	20	116	74	1110100	164	t
21	15	10101	25	(NEGATIVE ACKNOWLEDGE)	69	45	1000101	105	21	117	75	1110101	165	u
22	16	10110	26	(SYNCHRONOUS IDLE)	70	46	1000110	106	22	118	76	1110110	166	v
23	17	10111	27	(END OF TRANS. BLOCK)	71	47	1000111	107	23	119	77	1110111	167	w
24	18	11000	30	(CANCEL)	72	48	1001000	110	24	120	78	1110000	170	x
25	19	11001	31	(END OF MEDIUM)	73	49	1001001	111	25	121	79	1110001	171	y
26	1A	11010	32	(SUBSTITUTE)	74	4A	1001010	112	26	122	7A	1110100	172	z
27	1B	11011	33	(ESCAPE)	75	4B	1001011	113	27	123	7B	1110101	173	{
28	1C	11100	34	(FILE SEPARATOR)	76	4C	1001100	114	28	124	7C	1111000	174	
29	1D	11101	35	(GROUP SEPARATOR)	77	4D	1001101	115	29	125	7D	1111001	175	}
30	1E	11110	36	(RECORD SEPARATOR)	78	4E	1001110	116	30	126	7E	1111100	176	~
31	1F	11111	37	(UNIT SEPARATOR)	79	4F	1001111	117	31	127	7F	1111101	177	[DEL]
32	20	100000	40	(SPACE)	80	50	1010000	120	32					
33	21	100001	41		81	51	1010001	121	33					
34	22	100010	42		82	52	1010010	122	34					
35	23	100011	43		83	53	1010011	123	35					
36	24	100100	44		84	54	1010100	124	36					
37	25	100101	45	%	85	55	1010101	125	37					
38	26	100110	46	&	86	56	1010110	126	38					
39	27	100111	47		87	57	1010111	127	39					
40	28	101000	50	(	88	58	1011000	130	40					
41	29	101001	51	)	89	59	1011001	131	41					
42	2A	101010	52	+	90	5A	1011010	132	42					
43	2B	101011	53	*	91	5B	1011011	133	43					
44	2C	101100	54	,	92	5C	1011100	134	44					
45	2D	101101	55	-	93	5D	1011101	135	45					
46	2E	101110	56	=	94	5E	1011110	136	46					
47	2F	101111	57	/	95	5F	1011111	137	47					

ASCIIコード表  
(文字コード表)

## 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';

## 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)

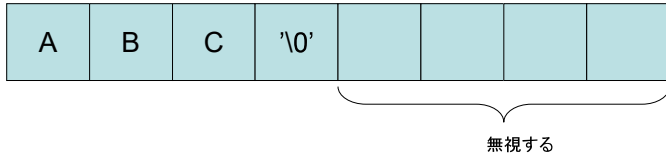
- ということは  
a = 'A'  
とするのは  
a = 65;  
とするのと**同じ**.

## 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:
```

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	00	00000000	000	'\0'	65	41	01000001	101	'A'	66	42	01000010	102	'B'
1	01	00000001	001	'\a'	67	43	01000011	103	'C'	68	44	01000100	104	'D'
2	02	00000010	010	'\n'	69	45	01000101	105	'E'	70	46	01000110	106	'F'
3	03	00000011	011	'\t'	71	47	01000111	107	'G'	72	48	01001000	110	'H'
4	04	00000100	100	'\b'	73	49	01001001	109	'I'	74	4A	01001010	112	'J'
5	05	00000101	101	'\f'	75	4B	01001011	111	'K'	76	4C	01001100	114	'L'
6	06	00000110	110	'\r'	77	4D	01001101	113	'M'	78	4E	01001110	116	'N'
7	07	00000111	111	'\e'	79	4F	01001111	115	'O'	80	50	01010000	120	'0'
8	08	00001000	1000	'\x08'	81	51	01010001	121	'1'	82	52	01010010	122	'2'
9	09	00001001	1001	'\x09'	83	53	01010011	123	'3'	84	54	01010100	124	'4'
10	0A	00001010	1010	'\x0A'	85	55	01010101	125	'5'	86	56	01010110	126	'6'
11	0B	00001011	1011	'\x0B'	87	57	01010111	127	'7'	88	58	01011000	130	'8'
12	0C	00001100	1100	'\x0C'	89	59	01011001	129	'9'	90	5A	01011010	132	'.'
13	0D	00001101	1101	'\x0D'	91	5B	01011011	131	','	92	5C	01011100	134	'<'
14	0E	00001110	1110	'\x0E'	93	5D	01011101	133	'>'	94	5E	01011110	136	'@'
15	0F	00001111	1111	'\x0F'	95	5F	01011111	135	'\a'	96	60	01100000	140	'@'
16	10	00010000	10000	'\x10'	97	61	01100001	137	'\n'	98	62	01100010	142	'@'
17	11	00010001	10001	'\x11'	99	63	01100011	139	'\r'	100	64	01100100	144	'@'
18	12	00010010	10010	'\x12'	101	65	01100101	141	'\f'	102	66	01100110	146	'@'
19	13	00010011	10011	'\x13'	103	67	01100111	143	'\x08'	104	68	01101000	150	'@'
20	14	00010100	10100	'\x14'	105	69	01101001	145	'\x0A'	106	6A	01101010	152	'@'
21	15	00010101	10101	'\x15'	107	6B	01101011	147	'\x0C'	108	6C	01101100	158	'@'
22	16	00010110	10110	'\x16'	109	6D	01101101	149	'\x0E'	110	6E	01101110	160	'@'
23	17	00010111	10111	'\x17'	111	6F	01101111	151	'\x0F'	112	70	01110000	166	'@'
24	18	00011000	11000	'\x18'	113	71	01110001	153	'\x10'	114	72	01110010	170	'@'
25	19	00011001	11001	'\x19'	115	73	01110011	155	'\x12'	116	74	01110100	176	'@'
26	1A	00011010	11010	'\x1A'	117	75	01110101	157	'\x14'	118	76	01110110	182	'@'
27	1B	00011011	11011	'\x1B'	119	77	01110111	159	'\x16'	120	78	01111000	188	'@'
28	1C	00011100	11100	'\x1C'	121	79	01111001	161	'\x18'	122	7A	01111010	194	'@'
29	1D	00011101	11101	'\x1D'	123	7B	01111011	163	'\x1A'	124	7C	01111100	200	'@'
30	1E	00011110	11110	'\x1E'	125	7D	01111101	165	'\x1C'	126	7E	01111110	206	'@'
31	1F	00011111	11111	'\x1F'	127	7F	01111111	167	'\x1E'	128	80	01111111	212	'@'
32	20	00100000	100000	'\x20'	129	81	10000000	201	'\x10'	130	82	10000001	214	'@'
33	21	00100001	100001	'\x21'	131	83	10000010	203	'\x12'	132	84	10000011	216	'@'
34	22	00100010	100010	'\x22'	133	85	10000100	205	'\x14'	134	86	10000101	218	'@'
35	23	00100011	100011	'\x23'	135	87	10000110	207	'\x16'	136	88	10000111	220	'@'
36	24	00100100	100100	'\x24'	137	89	10001000	209	'\x18'	138	8A	10001001	222	'@'
37	25	00100101	100101	'\x25'	139	8B	10001010	211	'\x1A'	140	8C	10001011	224	'@'
38	26	00100110	100110	'\x26'	141	8D	10001100	213	'\x1C'	142	8E	10001101	226	'@'
39	27	00100111	100111	'\x27'	143	8F	10001110	215	'\x1E'	144	90	10001111	228	'@'
40	28	00101000	101000	'\x28'	145	91	10010000	217	'\x20'	146	92	10010001	230	'@'
41	29	00101001	101001	'\x29'	147	93	10010010	219	'\x22'	148	94	10010011	232	'@'
42	2A	00101010	101010	'\x2A'	149	95	10010100	221	'\x24'	150	96	10010101	234	'@'
43	2B	00101011	101011	'\x2B'	151	97	10010110	223	'\x26'	152	98	10010111	236	'@'
44	2C	00101100	101100	'\x2C'	153	99	10011000	225	'\x28'	154	9A	10011001	238	'@'
45	2D	00101101	101101	'\x2D'	155	9B	10011010	227	'\x2A'	156	9C	10011011	240	'@'
46	2E	00101110	101110	'\x2E'	157	9D	10011100	229	'\x2C'	158	9E	10011101	242	'@'
47	2F	00101111	101111	'\x2F'	159	9F	10011110	231	'\x2E'	160	9F	10011111	244	'@'

## 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; 鑄造する, 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
  - strncpy, strncat など