

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	70	1101010	152	j
11	B	1011	13	(VERTICAL TAB)	59	3B	111011	73	11	107	71	1101011	153	k
12	C	1100	14	(FORM FEED)	60	3C	111100	74	12	108	72	1101100	154	l
13	D	1101	15	(CARRIAGE RETURN)	61	3D	111101	75	13	109	73	1101101	155	m
14	E	1110	16	(SHIFT OUT)	62	3E	111110	76	14	110	74	1101110	156	n
15	F	1111	17	(SHIFT IN)	63	3F	111111	77	15	111	75	1101111	157	o
16	10	10000	20	(DATA LINK ESCAPE)	64	40	1000000	100	16	112	76	1110000	160	p
17	11	10001	21	(DEVICE CONTROL 1)	65	41	1000001	101	17	113	77	1110001	161	q
18	12	10010	22	(DEVICE CONTROL 2)	66	42	1000010	102	18	114	78	1110010	162	r
19	13	10011	23	(DEVICE CONTROL 3)	67	43	1000011	103	19	115	79	1110011	163	s
20	14	10100	24	(DEVICE CONTROL 4)	68	44	1000100	104	20	116	80	1110100	164	t
21	15	10101	25	(NEGATIVE ACKNOWLEDGE)	69	45	1000101	105	21	117	81	1110101	165	u
22	16	10110	26	(SYNCHRONOUS IDLE)	70	46	1000110	106	22	118	82	1110110	166	v
23	17	10111	27	(END OF TRANS. BLOCK)	71	47	1000111	107	23	119	83	1110111	167	w
24	18	11000	30	(CANCEL)	72	48	1001000	110	24	120	84	1110000	170	x
25	19	11001	31	(END OF MEDIUM)	73	49	1001001	111	25	121	85	1110001	171	y
26	1A	11010	32	(SUBSTITUTE)	74	4A	1001010	112	26	122	86	1110100	172	z
27	1B	11011	33	(ESCAPE)	75	4B	1001011	113	27	123	87	1110101	173	{
28	1C	11100	34	(FILE SEPARATOR)	76	4C	1001100	114	28	124	88	1111000	174	
29	1D	11101	35	(GROUP SEPARATOR)	77	4D	1001101	115	29	125	89	1111001	175	}
30	1E	11110	36	(RECORD SEPARATOR)	78	4E	1001110	116	30	126	90	1111100	176	~
31	1F	11111	37	(UNIT SEPARATOR)	79	4F	1001111	117	31	127	91	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)

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

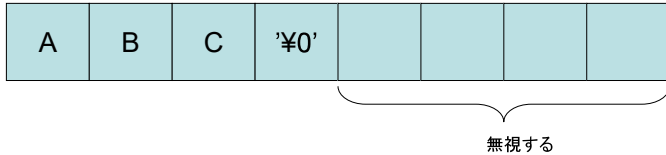
- 文字(漢字, ひらがな除く)は合計数十個
 - アルファベット(大文字・小文字), 数字, 記号
 - 8bit (256通り) で表すことが出来る
- 文字を表す型: char (character の略)
- 整数を表す型の一種(int などの仲間)
 - 通常, char 型は 8bit
 - 通常, int は 32bit (16bit の機械・マイコン系も)
- 文字コードを得る方法
 - 'a' のようにシングルクォートで挟む
char ch = 'a';

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:
```

Decimal	Hexadecimal	Binary	Ascii Char	Decimal	Hexadecimal	Binary	Ascii Char	Decimal	Hexadecimal	Binary	Ascii Char
0	0	00000000		128	80	10000000		256	100	100000000	
1	1	00000001	SOFT OF CRASHING	129	81	10000001	!	257	101	100000001	~
2	2	00000010	SOFT OF CRASHING	130	82	10000010	"	258	102	100000010	~
3	3	00000011	SOFT OF CRASHING	131	83	10000011	#	259	103	100000011	~
4	4	00000100	SOFT OF CRASHING	132	84	10000100	\$	260	104	100000100	~
5	5	00000101	SOFT OF CRASHING	133	85	10000101	%	261	105	100000101	~
6	6	00000110	SOFT OF CRASHING	134	86	10000110	&	262	106	100000110	~
7	7	00000111	SOFT OF CRASHING	135	87	10000111	'	263	107	100000111	~
8	8	00001000	SOFT OF CRASHING	136	88	10001000	(264	108	100001000	~
9	9	00001001	SOFT OF CRASHING	137	89	10001001)	265	109	100001001	~
10	A	00001010	SOFT OF CRASHING	138	8A	10001010	*	266	10A	100001010	~
11	B	00001011	SOFT OF CRASHING	139	8B	10001011	+	267	10B	100001011	~
12	C	00001100	SOFT OF CRASHING	140	8C	10001100	,	268	10C	100001100	~
13	D	00001101	SOFT OF CRASHING	141	8D	10001101	~	269	10D	100001101	~
14	E	00001110	SOFT OF CRASHING	142	8E	10001110	^	270	10E	100001110	~
15	F	00001111	SOFT OF CRASHING	143	8F	10001111	_	271	10F	100001111	~
16	0	00010000	SOFT OF CRASHING	144	90	10010000	`	272	110	100010000	~
17	1	00010001	SOFT OF CRASHING	145	91	10010001	a	273	111	100010001	~
18	2	00010010	SOFT OF CRASHING	146	92	10010010	b	274	112	100010010	~
19	3	00010011	SOFT OF CRASHING	147	93	10010011	c	275	113	100010011	~
20	4	00010100	SOFT OF CRASHING	148	94	10010100	d	276	114	100010100	~
21	5	00010101	SOFT OF CRASHING	149	95	10010101	e	277	115	100010101	~
22	6	00010110	SOFT OF CRASHING	150	96	10010110	f	278	116	100010110	~
23	7	00010111	SOFT OF CRASHING	151	97	10010111	g	279	117	100010111	~
24	8	00011000	SOFT OF CRASHING	152	98	10011000	h	280	118	100011000	~
25	9	00011001	SOFT OF CRASHING	153	99	10011001	i	281	119	100011001	~
26	A	00011010	SOFT OF CRASHING	154	9A	10011010	j	282	11A	100011010	~
27	B	00011011	SOFT OF CRASHING	155	9B	10011011	k	283	11B	100011011	~
28	C	00011100	SOFT OF CRASHING	156	9C	10011100	l	284	11C	100011100	~
29	D	00011101	SOFT OF CRASHING	157	9D	10011101	m	285	11D	100011101	~
30	E	00011110	SOFT OF CRASHING	158	9E	10011110	n	286	11E	100011110	~
31	F	00011111	SOFT OF CRASHING	159	9F	10011111	o	287	11F	100011111	~
32	0	00100000	SOFT OF CRASHING	160	A0	10100000	p	288	120	101000000	~
33	1	00100001	SOFT OF CRASHING	161	A1	10100001	q	289	121	101000001	~
34	2	00100010	SOFT OF CRASHING	162	A2	10100010	r	290	122	101000010	~
35	3	00100011	SOFT OF CRASHING	163	A3	10100011	s	291	123	101000011	~
36	4	00100100	SOFT OF CRASHING	164	A4	10100100	t	292	124	101000100	~
37	5	00100101	SOFT OF CRASHING	165	A5	10100101	u	293	125	101000101	~
38	6	00100110	SOFT OF CRASHING	166	A6	10100110	v	294	126	101000110	~
39	7	00100111	SOFT OF CRASHING	167	A7	10100111	w	295	127	101000111	~
40	8	00101000	SOFT OF CRASHING	168	A8	10101000	x	296	128	101010000	~
41	9	00101001	SOFT OF CRASHING	169	A9	10101001	y	297	129	101010001	~
42	A	00101010	SOFT OF CRASHING	170	AA	10101010	z	298	12A	101010010	~
43	B	00101011	SOFT OF CRASHING	171	AB	10101011	[299	12B	101010011	~
44	C	00101100	SOFT OF CRASHING	172	AC	10101100	\	300	12C	101010100	~
45	D	00101101	SOFT OF CRASHING	173	AD	10101101]	301	12D	101010101	~
46	E	00101110	SOFT OF CRASHING	174	AE	10101110	^	302	12E	101010110	~
47	F	00101111	SOFT OF CRASHING	175	AF	10101111	_	303	12F	101010111	~
48	0	00110000	SOFT OF CRASHING	176	B0	10110000	`	304	130	101100000	~
49	1	00110001	SOFT OF CRASHING	177	B1	10110001	a	305	131	101100001	~
50	2	00110010	SOFT OF CRASHING	178	B2	10110010	b	306	132	101100010	~
51	3	00110011	SOFT OF CRASHING	179	B3	10110011	c	307	133	101100011	~
52	4	00110100	SOFT OF CRASHING	180	B4	10110100	d	308	134	101100100	~
53	5	00110101	SOFT OF CRASHING	181	B5	10110101	e	309	135	101100101	~
54	6	00110110	SOFT OF CRASHING	182	B6	10110110	f	310	136	101100110	~
55	7	00110111	SOFT OF CRASHING	183	B7	10110111	g	311	137	101100111	~
56	8	00111000	SOFT OF CRASHING	184	B8	10111000	h	312	138	101110000	~
57	9	00111001	SOFT OF CRASHING	185	B9	10111001	i	313	139	101110001	~
58	A	00111010	SOFT OF CRASHING	186	BA	10111010	j	314	13A	101110010	~
59	B	00111011	SOFT OF CRASHING	187	BB	10111011	k	315	13B	101110011	~
60	C	00111100	SOFT OF CRASHING	188	BC	10111100	l	316	13C	101110100	~
61	D	00111101	SOFT OF CRASHING	189	BD	10111101	m	317	13D	101110101	~
62	E	00111110	SOFT OF CRASHING	190	BE	10111110	n	318	13E	101110110	~
63	F	00111111	SOFT OF CRASHING	191	BF	10111111	o	319	13F	101110111	~
64	0	01000000	SOFT OF CRASHING	192	C0	01000000	p	320	140	010000000	~
65	1	01000001	SOFT OF CRASHING	193	C1	01000001	q	321	141	010000001	~
66	2	01000010	SOFT OF CRASHING	194	C2	01000010	r	322	142	010000010	~
67	3	01000011	SOFT OF CRASHING	195	C3	01000011	s	323	143	010000011	~
68	4	01000100	SOFT OF CRASHING	196	C4	01000100	t	324	144	010000100	~
69	5	01000101	SOFT OF CRASHING	197	C5	01000101	u	325	145	010000101	~
70	6	01000110	SOFT OF CRASHING	198	C6	01000110	v	326	146	010000110	~
71	7	01000111	SOFT OF CRASHING	199	C7	01000111	w	327	147	010000111	~
72	8	01001000	SOFT OF CRASHING	200	C8	01001000	x	328	148	010010000	~
73	9	01001001	SOFT OF CRASHING	201	C9	01001001	y	329	149	010010001	~
74	A	01001010	SOFT OF CRASHING	202	CA	01001010	z	330	14A	010010010	~
75	B	01001011	SOFT OF CRASHING	203	CB	01001011	[331	14B	010010011	~
76	C	01001100	SOFT OF CRASHING	204	CC	01001100	\	332	14C	010010100	~
77	D	01001101	SOFT OF CRASHING	205	CD	01001101]	333	14D	010010101	~
78	E	01001110	SOFT OF CRASHING	206	CE	01001110	^	334	14E	010010110	~
79	F	01001111	SOFT OF CRASHING	207	CF	01001111	_	335	14F	010010111	~
80	0	01010000	SOFT OF CRASHING	208	D0	01010000	`	336	150	010100000	~
81	1	01010001	SOFT OF CRASHING	209	D1	01010001	a	337	151	010100001	~
82	2	01010010	SOFT OF CRASHING	210	D2	01010010	b	338	152	010100010	~
83	3	01010011	SOFT OF CRASHING	211	D3	01010011	c	339	153	010100011	~
84	4	01010100	SOFT OF CRASHING	212	D4	01010100	d	340	154	010100100	~
85	5	01010101	SOFT OF CRASHING	213	D5	01010101	e	341	155	010100101	~
86	6	01010110	SOFT OF CRASHING	214	D6	01010110	f	342	156	010100110	~
87	7	01010111	SOFT OF CRASHING	215	D7	01010111	g	343	157	010100111	~
88	8	01011000	SOFT OF CRASHING	216	D8	01011000	h	344	158	010110000	~
89	9	01011001	SOFT OF CRASHING	217	D9	01011001	i	345	159	010110001	~
90	A	01011010	SOFT OF CRASHING	218	DA	01011010	j	346	15A	010110010	~
91	B	01011011	SOFT OF CRASHING	219	DB	01011011	k	347	15B	010110011	~
92	C	01011100	SOFT OF CRASHING	220	DC	01011100	l	348	15C	010110100	~
93	D	01011101	SOFT OF CRASHING	221	DD	01011101	m	349	15D	010110101	~
94	E	01011110	SOFT OF CRASHING	222	DE	01011110	n	350	15E	010110110	~
95	F	01011111	SOFT OF CRASHING	223	DF	01011111	o	351	15F	010110111	~
96	0	01100000	SOFT OF CRASHING	224	E0	01100000	p	352	160	011000000	~
97	1	01100001	SOFT OF CRASHING	225	E1	01100001	q	353	161	011000001	~
98	2	01100010	SOFT OF CRASHING	226	E2	01100010	r	354	162	011000010	~
99	3	01100011	SOFT OF CRASHING	227	E3	01100011	s	355	163	011000011	~
100	4	01100100	SOFT OF CRASHING	228	E4	01100100	t	356	164	011000100	~
101	5	01100101	SOFT OF CRASHING	229	E5	01100101	u	357	165	011000101	~
102	6	01100110	SOFT OF CRASHING	230	E6	01100110	v	358	166	011000110	~
103	7	01100111	SOFT OF CRASHING	231	E7	01100111	w	359	167	011000111	~
104	8	01101000	SOFT OF CRASHING	232	E8	01101000	x	360	168	011010000	~
105	9	01101001	SOFT OF CRASHING	233	E9	01101001	y	361	169	011010001	~
106	A	01101010	SOFT OF CRASHING	234	EA	01101010	z	362	16A	011010010	~
107	B	01101011	SOFT OF CRASHING	235	EB	01101011	[363	16B	011010011	~
108	C	01101100	SOFT OF CRASHING	236	EC	01101100	\	364	16C	011010100	~
109	D	01101101	SOFT OF CRASHING	237	ED	01101101]	365	16D	011010101	~
110	E	01101110	SOFT OF CRASHING	238	EE	01101110	^	366	16E	011010110	~
111	F	01101111	SOFT OF CRASHING	239	EF	01101111	_	367	16F	011010111	~
112	0	01110000	SOFT OF CRASHING	240	F0	01110000	`	368			

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