1차원 정적 배열

```
자료형 변수명[배열 크기];
```

```
#include<stdio.h>
int main()
{
    int arr[4] = { 1, 2, 3, 4 };
    int arr2[] = { 1, 5 };
    int arr3[3] = { 0, };

    printf("arr의 byte 크기 : %d\n", sizeof(arr));
    printf("arr2의 byte 크기 : %d\n", sizeof(arr2));
    printf("arr3의 byte 크기 : %d\n\n", sizeof(arr3));

    for (int i = 0;i < sizeof(arr) / sizeof(int);i++) {
        printf("arr[%d] : %d\n", i, arr[i]);
    }
    printf("\n");
    for (int i = 0;i < sizeof(arr2) / sizeof(int);i++) {
        printf("arr2[%d] : %d\n", i, arr2[i]);
    }
    printf("\n");
    for (int i = 0;i < sizeof(arr3) / sizeof(int);i++) {
        printf("arr3[%d] : %d\n", i, arr3[i]);
    }
    return 0;
}
```

output:

```
arr의 byte 크기 : 16
arr2의 byte 크기 : 8
arr3의 byte 크기 : 12

arr[0] : 1
arr[1] : 2
arr[2] : 3
arr[3] : 4

arr2[0] : 1
arr2[1] : 5

arr3[0] : 0
arr3[1] : 0
arr3[2] : 0
```

[]안에 있는 값만큼 변수를 선언해준다고 생각하면 된다.
이때, 선언부에 있는 []는 크기를 의미하고, 사용할 때 쓰는 []는 인덱스 위치 즉, 배열이 몇 번째 배열인지를 의미한다.

다차원 배열

```
자료형 변수명[크기][크기][크기];

[] 개수에 따라 1차원 배열, 2차원 배열, 3차원 배열 등으로 부른다.
위와 같이 []를 3개 썼다면, 3차원 배열이다.

배열의 해석은 뒤에서 앞으로한다.
```

```c
#include<stdio.h>
int main()
{
    int arr[3][4] = { {1, 2, 3, 4},
                      {5, 6, 7, 8},
                      {9, 10, 11, 12} };

    for (int i = 0;i < 3;i++) {
        for (int j = 0;j < 4;j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

output:

```
1 2 3 4
5 6 7 8
9 10 11 12
```

위 예제처럼 arr[3][4]로 선언했다면, 우선 뒤에 있는 [4]부터 해석을 하면 된다.
arr[4]를 표현해보면,

[0]	[1]	[2]	[3]

가 된다.
[3]은 위에 그린 4개의 칸이 3번 더 그려진다고 생각하면 된다.

[0][0]	[0][1]	[0][2]	[0][3]
[1][0]	[1][1]	[1][2]	[1][3]
[2][0]	[2][1]	[2][2]	[2][3]

ASCII

```
#include<stdio.h>
int main()
{
    char a = 65;
    int b = 67;
    char c = 'A';

    printf("%d %d %d\n", a, b, c);
    printf("%c %c %c\n", a, b, c);
    return 0;
}
```

output:

65 67 65

A C A

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

ASCII코드는 7비트 부호 체계로 8비트 중 1비트는 통신 에러 검출을 위한 용도로 비워뒀기 때문이다. 7비트만 사용하므로, 0~127까지의 문자를 표현할 수 있다.

문자형으로 사용하는 방법으로 ‘’안에 문자를 적으면 컴파일하면서 자동으로 아스키코드로 변환된다. 따라서, 아스키코드표를 외우고 있을 필요는 없다.

문자열

```c
#include<stdio.h>
int main()
{
    char str[50] = "Hello, World!";
    int len = -1;

    while (str[++len] != NULL);

    printf("문자열의 길이 : %d\n", len);
    printf("문자열 str : %s\n", str);
    printf("문자열의 끝 : %d\n", str[len]);
    return 0;
}
```

output:

```
문자열의 길이 : 13
문자열 str : Hello, World!
문자열의 끝 : 0
```

문자열은 간단하게 생각해서 문자의 배열이라고 생각하면 된다.
다만 일반적인 배열과 달리 문자열은 하나의 문장을 표현하기 위해 사용하므로, 서식지정자로 %s라는 것을 사용한다.
%s로 간편하게 읽어오기 위해선, 문자열을 자동으로 탐색하는 방법이 필요하다.
그러기 위해서 문자열의 끝에 NULL이라는 것을 삽입해준다.
NULL 문자를 끝에 삽입하기 때문에, 문자열의 크기를 정할 때도 항상 NULL이 들어갈 자리도 생각해서 정해야한다.
직접 문자열의 길이를 찾아낼 때도, 인덱스를 움직이며 NULL의 위치를 찾아내는 방식을 사용한다.