

그 전에 잠깐!

서식문자란 %d, %f, %c, %s

입력이나 출력 형태를 지정하는 문자!

ex. printf("%d" , a); // 변수 a에 들어있는 값을 정수 형태로 출력

scanf("%f", &b); // 변수 b에 들어갈 값을 실수 형태로 입력

서식문자	실수 자료형
%f	float
	double

정정!

서식문자	실수 자료형
%f	float
%lf	double

printf로 출력할 때에는 혼용해도 문제가 발생하지 않았으나,

scanf로 입력받을 때 혼용해서 사용하면 오류가 발생!

```
#include<stdio.h>
int main(void) {
    float num1, num2;
    scanf("%f %lf", &num1, &num2);
    printf("%lf %f", num1, num2);
    return 0;
}
```

8. C언어가 제공하는 기본 자료형의 이해

- 자료형(data type) : 데이터를 표현하는 방법

■ 기본자료형의 종류와 데이터 표현범위

자료형	크기	값의 표현범위
정수형	char	$-128 \sim +127$ ($-2^7 \sim 2^7-1$)
	unsigned char	$0 \sim 255$ ($0 \sim 2^8-1$)
	short	$-32,768 \sim +32,767$ ($-2^{15} \sim 2^{15}-1$)
	unsigned short	$0 \sim +65,535$ ($0 \sim 2^{16}-1$)
	int	$-2,147,483,648 \sim +2,147,483,647$ ($-2^{31} \sim 2^{31}-1$)
	unsigned int	$0 \sim +4,294,967,296$ ($0 \sim 2^{32}-1$)
	long	$-2,147,483,648 \sim +2,147,483,647$ ($-2^{31} \sim 2^{31}-1$)
	unsigned long	$0 \sim +4,294,967,296$ ($0 \sim 2^{32}-1$)
	long long	$-9,223,372,036,854,775,808 \sim +9,223,372,036,854,775,807$ ($-2^{63} \sim 2^{63}-1$)
실수형	float	$-3.402823466 \times 10^{38} \sim 3.402823466 \times 10^{38}$
	double	$-1.7976931348623158 \times 10^{308} \sim 1.7976931348623158 \times 10^{308}$

■ 연산자 sizeof를 이용하여 자료형의 크기를 확인

```
#include <stdio.h>
int main()
{
    char ch = 9;
    int inum = 1052;
    double dnum = 3.141592;

    printf("변수 ch의 크기 : %d\n", sizeof(ch));
    printf("변수 inum의 크기 : %d\n", sizeof(inum));
    printf("변수 dnum의 크기 : %d\n", sizeof(dnum));

    printf("char의 크기 : %d\n", sizeof(char));
    printf("int의 크기 : %d\n", sizeof(int));
    printf("long의 크기 : %d\n", sizeof(long));
    printf("long long의 크기 : %d\n", sizeof(long long));
    printf("float의 크기 : %d\n", sizeof(float));
    printf("double의 크기 : %d\n", sizeof(double));

    return 0;
}
```

실행결과

```
Microsoft Visual Studio
변수 ch의 크기 : 1
변수 inum의 크기 : 4
변수 dnum의 크기 : 8
char의 크기 : 1
int의 크기 : 4
long의 크기 : 4
long long의 크기 : 8
float의 크기 : 4
double의 크기 : 8
```

8. 문자의 표현방식과 문자를 위한 자료형

■ 문자의 표현을 위한 약속! 아스키(ASCII)코드!

- 미국 표준 협회(ANSI)에 의해 정의
- 컴퓨터는 0과1 이진수로 이루어져 있기 때문에 문자를 표현하기 위한 표준 코드
- 문자와 숫자의 연결 관계를 정의(문자 A는 숫자 65, 문자 B는 숫자 66...)
- ASCII 코드의 범위(0이상 127이하)
- 문자의 표현은 따옴표(' ')를 이용해서 표현

문자	10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진
NUL	0	0x00	SP	32	0x20	@	64	0x40	.	96	0x60
SOH	1	0x01	!	33	0x21	A	65	0x41	a	97	0x61
STX	2	0x02	"	34	0x22	B	66	0x42	b	98	0x62
ETX	3	0x03	#	35	0x23	C	67	0x43	c	99	0x63
EOT	4	0x04	\$	36	0x24	D	68	0x44	d	100	0x64
ENQ	5	0x05	%	37	0x25	E	69	0x45	e	101	0x65
ACK	6	0x06	&	38	0x26	F	70	0x46	f	102	0x66
BEL	7	0x07	'	39	0x27	G	71	0x47	g	103	0x67
BS	8	0x08	(40	0x28	H	72	0x48	h	104	0x68
HT	9	0x09)	41	0x29	I	73	0x49	i	105	0x69
LF	10	0x0A	*	42	0x2A	J	74	0x4A	j	106	0x6A
VT	11	0x0B	+	43	0x2B	K	75	0x4B	k	107	0x6B
FF	12	0x0C	,	44	0x2C	L	76	0x4C	l	108	0x6C
CR	13	0x0D	-	45	0x2D	M	77	0x4D	m	109	0x6D
SO	14	0x0E	.	46	0x2E	N	78	0x4E	n	110	0x6E
SI	15	0x0F	/	47	0x2F	O	79	0x4F	o	111	0x6F
DLE	16	0x10	0	48	0x30	P	80	0x50	p	112	0x70
DC1	17	0x11	1	49	0x31	Q	81	0x51	q	113	0x71
DC2	18	0x12	2	50	0x32	R	82	0x52	r	114	0x72
DC3	19	0x13	3	51	0x33	S	83	0x53	s	115	0x73
DC4	20	0x14	4	52	0x34	T	84	0x54	t	116	0x74
NAK	21	0x15	5	53	0x35	U	85	0x55	u	117	0x75
SYN	22	0x16	6	54	0x36	V	86	0x56	v	118	0x76
ETB	23	0x17	7	55	0x37	W	87	0x57	w	119	0x77
CAN	24	0x18	8	56	0x38	X	88	0x58	x	120	0x78
EM	25	0x19	9	57	0x39	Y	89	0x59	y	121	0x79
SUB	26	0x1A	:	58	0x3A	Z	90	0x5A	z	122	0x7A
ESC	27	0x1B	;	59	0x3B	[91	0x5B	{	123	0x7B
FS	28	0x1C	<	60	0x3C	\	92	0x5C		124	0x7C
GS	29	0x1D	=	61	0x3D]	93	0x5D	}	125	0x7D
RS	30	0x1E	>	62	0x3E	^	94	0x5E	~	126	0x7E
US	31	0x1F	?	63	0x3F	_	95	0x5F	DEL	127	0x7F

```
#include <stdio.h>
int main()
{
    char ch1 = 'A', ch2 = 65;
    int ch3 = 'Z', ch4 = 90;

    printf("%c %d\n", ch1, ch1);
    printf("%c %d\n", ch2, ch2);
    printf("%c %d\n", ch3, ch3);
    printf("%c %d\n", ch4, ch4);
    return 0;
}
```

실행결과

- 서식문자 %c : 문자의 형태로 데이터를 출력하라
- 정수는 출력의 방법에 따라서 문자의 형태로도, 숫자의 형태로도 출력이 가능하다!

문제 04-1 [scanf 함수와 아스키코드]

1. 두 점의 x , y 좌표를 입력받아서, 두 점이 이루는 직사각형의 넓이는 계산하여 출력하는 프로그램을 작성해보자. 단 좌표 (x_1, y_1) , (x_2, y_2) 에서 $x_1 < x_2$ 이고 $y_1 < y_2$ 이다!

예시)

```
Microsoft Visual Studio 디버그 콘솔
x1, y1 좌표 (예시 : 1 2) : 2 4
x2, y2 좌표 (예시 : 4 5) : 4 8
두 점이 이루는 직사각형의 넓이는 8 입니다.
```

2. 사용자로부터 두 개의 실수를 입력받아서 double형 변수에 저장하자. 그리고 두 수의 사칙연산 결과를 출력해보자.

예시) 두 개의 실수 입력 (0 제외): 1.23 4.56

```
1.230000 + 4.560000 = 5.790000
1.230000 - 4.560000 = -3.330000
1.230000 * 4.560000 = 5.608800
1.230000 / 4.560000 = 0.269737
```

3. 프로그램 사용자로부터 아스키 코드 값을 정수의 형태로 입력받은 후에 해당 정수의 아스키 코드 문자를 출력하는 프로그램을 작성해보자.

예시)

```
Microsoft Visual Studio 디버그 콘솔
아스키코드 값 정수로 입력 : 70
70에 해당하는 아스키코드 문자는 F입니다.
```

4. 프로그램 사용자로부터 알파벳 문자 하나를 입력받는다. 그리고 이에 해당하는 아스키 코드 값을 출력하는 프로그램을 작성해보자.

예시)

```
Microsoft Visual Studio 디버그 콘솔
알파벳 입력 (예시: p) : i
i 에 해당하는 아스키 코드 값은 105 입니다.
```

9. 상수에 대한 이해

- 상수 : 숫자의 구체적인 값이 확정되어 앞으로 변할 가능성이 없는 수

상수종류		설명
리터럴 상수	문자상수	영문자나 특수문자를 기술한 상수 예) 'A'
	문자열 상수	문자 상수들의 연속된 집합체를 표현한 상수. 다른 상수들과 달리 여러 개가 한 덩어리를 이룬 배열이며 '주소'로 식별될 수 있음 예) "Hello, World!"
	정수 상수	가장 흔하게 생각할 수 있는 숫자 상수 예) 3, 4L
	실수 상수	소수점을 포함한 실수를 표현한 상수 예) 3.4F, 123.45
심볼릭 상수	형한정	const 형한정어를 이용해 선언 및 정의되는 상수. 쉽게 생각하면 값이 변할 수 없는 변수이며, 이름을 가진 상수 예) const int nMax = 100;
	전처리기	#define 전처리기로 선언 및 정의되는 상수. 이름을 가진다는 점에서 const를 이용하는 경우와 같지만 변수가 아니라 명백한 상수이다.

- 리터럴 상수 크기 확인

```
#include <stdio.h>
int main()
{
    printf("literal int size : %d \n", sizeof(7));
    printf("literal double size : %d \n", sizeof(3.14));
    printf("literal char size : %d \n", sizeof('A'));

    return 0;
}
```

실행결과

Microsoft Visual Studio 디버그 콘솔

```
literal int size : 4
literal double size : 8
literal char size : 4
```

■ 접미사를 이용한 다양한 상수의 표현

- 정수형 상수의 표현을 위한 접미사

접미사	자료형	사용의 예
U	unsigned int	unsigned int n = 1024U;
L	long	long n = 2468L;
UL	unsigned long	unsigned long n = 3456UL;
LL	long long	long long n = 9876LL;
ULL	unsigned long long	unsigned long long n = 123456789ULL;

- 실수형 상수의 표현을 위한 접미사

접미사	자료형	사용의 예
f, F	float	float f = 3.14f;
생략	double	double d = 3.14;
l, L	long double	long double ld = 3.14l;

■ 이름을 지니는 심볼릭(Symbolic)상수 : const 상수

- 심볼릭 상수 : 변수와 마찬가지로 이름을 지니는 상수, const 키워드 사용
- 변수 선언시 const 선언 추가, 선언과 동시에 초기화 필수!

```
const int MAX = 100;    // MAX는 상수!
// MAX = 200;         // 상수로 정의했으므로 값의 변경 불가!
const double PI = 3.1415; // PI는 상수!
// PI = 3.14;         // 상수로 정의했으므로 값의 변경 불가!
```

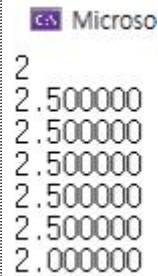
■ 자료형 불일치와 형 변환 연산자

```
#include <stdio.h>
int main()
{
    int x = 5;

    printf("%d\n", 5 / 2);           // int / int -> int
    printf("%f\n", 5.0 / 2);        // double / int -> double
    printf("%f\n", 5 / 2.0);        // int / double -> double
    printf("%f\n", (double)5 / 2);  // double / int -> double
    printf("%f\n", (double)x / 2);  // double / int -> double
    printf("%f\n", x / (double)2);  // int / double -> double
    printf("%f\n", (double)(x / 2)); // double

    return 0;
}
```

실행결과



```
2
2.500000
2.500000
2.500000
2.500000
2.500000
2.000000
```

■ 형식 문자와 이스케이프 시퀀스

- 형식 문자

형식문자	자료형	출력 형식
%c	int(char)	character. ASCII 문자로 출력
%d	int	Decimal. 부호가 있는 10진수로 출력
%o	int	Octal. 8진수로 출력
%u	unsigned int	Unsigned. 부호가 없는 10진수로 출력
%x, %X		Hexa. 16진수로 출력
%e, %E	float, double	Exponent. 지수형 소수로 출력
%f, %lf	float, double	Float. 10진형 소수로 출력
%g	double	지수형 소수(%e)나 10진형 소수(%f)로 출력. 단, 출력되는 문자열이 짧은 형태로 출력한다.
%p	Pointer	16진수 주소로 출력
%s	String	인수가 가리키는 메모리의 내용을 문자열로 출력

- 이스케이프 시퀀스

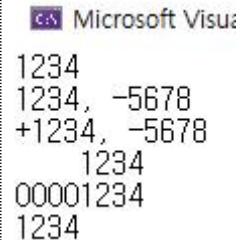
문자	의미	문자	의미
\a	경고음 울림	\\	Backslash
\b	Backspace	\?	물음표
\f	인쇄시 종이 한 장 넘김(Form feed)	\'	작은 따옴표, 문자 상수
\n	New Line	\"	큰 따옴표, 문자열 상수
\r	Carriage return	\v	Vertical tab (프린터 출력을 위한...)
\t	Tab		
\ooo	\뒤에 8진수 숫자를 지정하여 ASCII 코드의 문자 표현 예) \141은 a를 표현		
\xhh	\뒤에 16진수 숫자를 지정하여 ASCII 코드의 문자 표현 예) \x61은 a를 표현		

● 정수 출력형식

```
#include <stdio.h>
int main()
{
    printf("%d\n", 1234);
    printf("%d, %d\n", 1234, -5678);
    // 부호를 붙여서 출력한다.
    // 단순히 더하기 기호(+)를 출력하는 것이 아니다!
    printf("%+d, %+d\n", 1234, -5678);

    // 정수를 8자리로 맞추고 오른쪽 정렬해 출력한다.
    printf("%8d\n", 1234);
    // 정수를 8자리로 맞추고 왼쪽 빈공간을 모두 0으로 채워서 출력한다.
    printf("%08d\n", 1234);
    // 정수를 8자리로 맞추고 왼쪽 정렬해 출력한다.
    // 따라서 공백이나 0이 채워지지 않는다.
    printf("%-08d\n", 1234);
    return 0;
}
```

실행결과



```
Microsoft Visual Studio
1234
1234, -5678
+1234, -5678
      1234
00001234
1234
```

● 실수 출력형식

```
#include <stdio.h>
int main()
{
    double dData = 123.4567890;

    printf("%f, %f\n", dData, -dData);

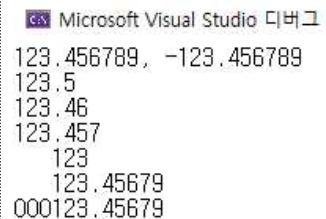
    // 소수점 2번째 자리에서 반올림하고 1자리까지 출력
    printf("%.1f\n", dData);
    // 소수점 3번째 자리에서 반올림하고 2자리까지 출력
    printf("%.2f\n", dData);
    // 소수점 4번째 자리에서 반올림하고 3자리까지 출력
    printf("%.3f\n", dData);

    printf("%6d\n", 123);

    // 소수점을 포함해 12자리로 출력. 단, 소수점 이하 6번째 자리에서
    // 반올림하고 5자리까지 출력한다.
    printf("%12.5f\n", dData);
    printf("%012.5f\n", dData);

    return 0;
}
```

실행결과



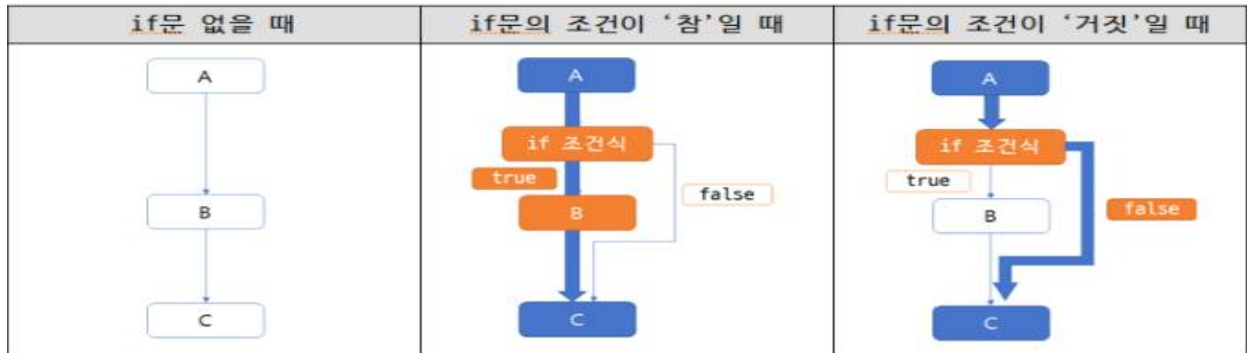
```
Microsoft Visual Studio 디버그
123.456789, -123.456789
123.5
123.46
123.457
      123
      123.45679
000123.45679
```

10. 기본 제어문

■ if문

- if 문은 다른 말로 분기문이라고 하며, 조건식을 근거로 구문들의 절차상 흐름을 변경

■ 기본구조



```

#define _CRT_SECURE_NO_WARNINGS // visual studio 에서만 필요!
#include <stdio.h>
int main()
{
    int age = 0;
    printf("나이를 입력하세요. : ");
    scanf("%d", &age);
    // if문 시작
    if (age >= 20)
        // 조건식을 만족한 경우에만 실행하는 구문
        printf("당신의 나이는 %d세 입니다.\n", age);
    // if문 끝
    printf("프로그램 끝!!!\n");
    return 0;
}
    
```

실행결과

·25를 입력했을 경우 결과

·17를 입력했을 경우 결과

■ 제어문과 스코프(Scope) *스코프란 번역하면 '범위', 즉 접근할 수 있는 범위를 의미함.

- 만일 if문에서 조건식을 만족할 때 수행할 구문이 여러 개면, 반드시 블록 스코프(괄호)로 묶어야 함.

```

#define _CRT_SECURE_NO_WARNINGS // visual studio 만 필요!
#include <stdio.h>
int main()
{
    int height = 0;
    printf("키를 입력하세요. : ");
    scanf("%d", &height);
    // if문 시작
    if (height >= 100)
    {
        // 조건식을 만족했을 때 실행할 구문들
        // 여러 구문을 실행해야 한다면 반드시 블록 스코프로 묶는다.
        printf("당신의 키는 %d cm 입니다.\n", height);
        height -= 110;
    } // <- if문 끝
    printf("당신의 적정 몸무게는 %d kg 입니다.\n", height);
    return 0;
}
    
```

실행결과

·170를 입력했을 경우 결과

·50을 입력했을 경우 결과

● if문 작성 연습(1)

```
#define _CRT_SECURE_NO_WARNINGS // visual studio 만 필요!
#include <stdio.h>
int main()
{
    int num;
    printf("정수 입력 : ");
    scanf("%d", &num);

    // num이 0보다 작으면 "입력 값(%d)은 0보다 작다.\n" 출력

    // num이 0보다 크면 "입력 값(%d)은 0보다 크다.\n" 출력

    // num이 0이면 "입력 값은 0이다.\n" 출력

    return 0;
}
```

실행결과

Microsoft Visual Studio 디버그 콘솔
정수 입력 : 10
입력 값(10)은 0보다 크다.

Microsoft Visual Studio 디버그 콘솔
정수 입력 : -6
입력 값(-6)은 0보다 작다.

Microsoft Visual Studio 디버그 콘솔
정수 입력 : 0
입력 값은 0이다.

● if문 작성 연습(2)

```
#define _CRT_SECURE_NO_WARNINGS // visual studio 만 필요!
#include <stdio.h>
int main()
{
    int opt;
    double num1, num2;
    double result;

    printf("1. 덧셈\t2. 뺄셈\t3. 곱셈\t4. 나눗셈\n"); // 연산 메뉴 show
    printf("선택 : "); // 메뉴 입력 안내
    scanf("%d", &opt); // 1,2,3,4 중 메뉴 입력
    printf("두 개의 실수 입력 : "); // 실수 입력 안내
    scanf("%lf %lf", &num1, &num2); // 실수 2개 입력

    // opt 가 1이면 result = num1+num2; 실행

    // opt 가 2이면 result = num1-num2; 실행

    // opt 가 3이면 result = num1*num2; 실행

    // opt 가 4이면 result = num1/num2; 실행

    printf("결과 : %f\n", result);
    return 0;
}
```

실행결과

Microsoft Visual Studio 디버그 콘솔
1. 덧셈 2. 뺄셈 3. 곱셈 4. 나눗셈
선택 : 1
두 개의 실수 입력 : 1.23 4.567
결과 : 5.797000

Microsoft Visual Studio 디버그 콘솔
1. 덧셈 2. 뺄셈 3. 곱셈 4. 나눗셈
선택 : 2
두 개의 실수 입력 : 45.29 96.33
결과 : -51.040000

Microsoft Visual Studio 디버그 콘솔
1. 덧셈 2. 뺄셈 3. 곱셈 4. 나눗셈
선택 : 3
두 개의 실수 입력 : 12.34 56.78
결과 : 700.665200

Microsoft Visual Studio 디버그 콘솔
1. 덧셈 2. 뺄셈 3. 곱셈 4. 나눗셈
선택 : 4
두 개의 실수 입력 : 1.111 3.141592
결과 : 0.353642

● if문 작성 연습(3) []

입력

두 정수 a, b가 입력된다.

출력

a가 b보다 큰 경우 출력: 입력받은 수 중 큰 수는 a이고 작은 수는 b입니다.

b가 a보다 큰 경우 출력: 입력받은 수 중 큰 수는 b이고 작은 수는 a입니다.

a와 b가 같은 경우: 입력받은 수 a, b는 동일합니다.