printf

```
#include<stdio.h>
int main()
{
	printf("Hello World!\t");
	printf("Hello World!");
	printf("\n");

	printf("Hello World!\n");
	printf("Hello World!");
	return 0;
}
```

output:

```
Hello World!	Hello World!
Hello World!
Hello World!
```

#include<>는 <>안에 있는 파일을 포함하는 코드로 위 코드에서는 stdio.h 파일을 포함한다는 의미를 가진다. stdio.h는 standard input output의 축약어이다. 즉, 기본적인 입출력 함수들을 담고 있는 파일이라고 생각하면 된다.

int main(){}은 C언어 프로그램을 실행시켰을 때, 가장 처음 접근하는 함수로 메인 함수라고 부른다. {} 안에 실행시키고 싶은 명령어들을 작성하면 된다. 단, 마지막에 무조건 return 0; 를 붙여줘야만 한다.

printf()는 CMD(콘솔창)에 글씨를 출력하는 함수로 “”안에 작성한 글자들이 CMD창에 출력된다. \t, \n등 \가 붙는 것을 이스케이프 문자라고 부른다.

※ printf는 print + formatted의 축약어이다.

이스케이프 문자	표현
\a	벨(경고)
\b	백스페이스
\f	폼 피드
\n	줄 바꿈
\r	캐리지 리턴(커서를 처음으로 옮김)
\t	가로 탭
\v	세로 탭
\’	작은 따옴표
\“	큰 따옴표
\\	백슬래시
\?	리터럴 물음표
\o	8진수 (ex. \o31)
\x	16진수 (ex. \xAA)

이때 ;(세미콜론)은 코드 한 줄이 끝났음을 뜻하므로, 항상 코드의 끝에 붙여줘야한다.

변수

```c
#include<stdio.h>
int main()
{
    int a = 3;
    char b = 'A';
    double c = 3.14;

    printf("정수형 변수 a 출력 : %d\n", a);
    printf("문자형 변수 b 출력 : %c\n", b);
    printf("실수형 변수 c 출력 : %lf", c);
    return 0;
}
```

output:

```
정수형 변수 a 출력 : 3
문자형 변수 b 출력 : A
실수형 변수 c 출력 : 3.140000
```

변수는

자료형 변수명;

으로 선언한다.

자료형	크기	서식지정자
char	1byte	%c
short	2byte	%d
int	4byte	%d, %i(잘 쓰이지 않음)
long	4byte	%d
float	4byte	%f
double	8byte	%lf
long long	8byte	%lld
unsigned int	4byte	%u
unsigned long long	8byte	%llu

printf 또는 scanf 함수를 사용할 때, 서식지정자를 사용하여 변수를 사용할 수 있다.

scanf

```
#include<stdio.h>
int main()
{
    int a;
    char b;
    double c;

    printf("정수형 변수 a 입력 : ");
    scanf("%d", &a);
    printf("문자형 변수 b 입력 : ");
    scanf("%c", &b);
    printf("실수형 변수 c 입력 : ");
    scanf("%lf", &c);

    printf("\n");
    printf("정수형 변수 a 출력 : %d\n", a);
    printf("문자형 변수 b 출력 : %c\n", b);
    printf("실수형 변수 c 출력 : %lf", c);
    return 0;
}
```

위와 같이 쓰면 제대로 입력이 되지 않는다.
그 이유는 scanf의 작동 방식 때문이다.
scanf는 버퍼에 입력한 문자를 담고 서식지정자에 따라 맞춰서 가져가지만, %c와 같은 경우는 \n과 공백 모두 문자로 취급하기 때문에 뒤에 붙인 \n 또는 공백을 가져가서 오류가 발생하는 것이다. 따라서 위 코드를 수정하면

```c
#include<stdio.h>
int main()
{
    int a;
    char b;
    double c;

    printf("문자형 변수 b 입력 : ");
    scanf("%c", &b);
    printf("정수형 변수 a 입력 : ");
    scanf("%d", &a);
    printf("실수형 변수 c 입력 : ");
    scanf("%lf", &c);

    printf("\n");
    printf("정수형 변수 a 출력 : %d\n", a);
    printf("문자형 변수 b 출력 : %c\n", b);
    printf("실수형 변수 c 출력 : %lf", c);
    return 0;
}
```

input:

```
A 3 3.14
```

output:

```
정수형 변수 a 출력 : 3
문자형 변수 b 출력 : A
실수형 변수 c 출력 : 3.140000
```

다음과 같이 문자형을 먼저 입력받거나

```c
#include<stdio.h>
int main()
{
    int a;
    char b;
    double c;

    printf("정수형 변수 a 입력 : ");
    scanf("%d", &a);
    scanf("%c");
    printf("문자형 변수 b 입력 : ");
    scanf("%c", &b);
    printf("실수형 변수 c 입력 : ");
    scanf("%lf", &c);

    printf("\n");
    printf("정수형 변수 a 출력 : %d\n", a);
    printf("문자형 변수 b 출력 : %c\n", b);
    printf("실수형 변수 c 출력 : %lf", c);
    return 0;
}
```

input:

```
3 A 3.14
```

output:

```
정수형 변수 a 출력 : 3
문자형 변수 b 출력 : A
실수형 변수 c 출력 : 3.140000
```

다음과 같이 scanf를 한 번 더 써서 버퍼를 비우는 방식이 존재한다.

scanf는 printf와 달리 서식지정자를 이용하여 변수와 연결할 때, &를 붙여줘야한다.
&를 붙이는 이유는 뒤에 포인터를 공부할 때, 추가로 설명한다.

※ scanf는 scan + formatted의 축약어이다.

C언어에서 정확한 자료형의 기준

자료형	크기
char	최소 1byte인 정수형
short	최소 2byte인 정수형
int	short보다 크고 최소 2byte인 정수형
long	최소 4byte 이상인 정수형

거의 대부분 int는 4byte를 의미하지만 달라질 수도 있다.

```
#include<stdio.h>
int main()
{
    int a;
    char b;
    double c;
    printf("int형의 크기 %d\n", sizeof(a));
    printf("char형의 크기 %d\n", sizeof(b));
    printf("double형의 크기 %d\n", sizeof(c));
    return 0;
}
```

output:

```
int형의 크기 4
char형의 크기 1
double형의 크기 8
```

자료형의 범위 알아내기

1byte == 8bit를 의미한다.
bit란, 0 또는 1을 나타내는 하나의 칸이다.
따라서 1byte는 0 또는 1을 나타내는 8개의 칸을 의미하며, 1byte로 표현할 수 있는 수는 0000 0000, 0000 0001, 0000 0010, 0000 0011 등등으로 총 2^8개의 수를 표현할 수 있다.

int형은 4byte를 가지며 4byte는 32bit이다.
따라서, int형이 표현할 수 있는 수는 총 2^{32}개이며, 정수형은 음수와 양수로 나누어져 있기 때문에, int형의 범위는 -2^{31}~$2^{31}-1$가 된다. 만약 unsigned를 붙여 음수를 지우면 0~2^{32}-1 이 범위가 된다.

컴퓨터가 2진수와 16진수를 주로 사용하는 이유

컴퓨터는 0과 1만을 사용하기 때문에 2의 배수 단위를 자주 사용한다.
근데 왜 8진수는 빼고 2진수와 16진수만을 사용할까? 주소값도 16진수로 표현하고…

일단 1바이트로 표현할 수 있는 수는 0~255이다.
16진수로 나타내면 0x00 ~ 0xFF
2진수로 나타내면 0 ~ 1111 1111 이다.

그렇기 때문에 쓴다고 생각하면 된다
2바이트라면 16진수로 0x00 ~ 0xFFFF
3바이트라면 16진수로 0x00 ~ 0xFFFFFF
4바이트라면 16진수로 0x00 ~ 0xFFFFFFFF
처럼 조금 더 간단하게 표현할 수 있어진다.