

# C 프로그래밍 1

## Lecture Note #06

---

백윤철  
ybaek@smu.ac.kr

# 내용

---

- printf()
- escape sequences
- conversion character
- scanf()
- conversion character

# printf 함수와 특수문자(Escape Sequence)

```
int main(void) {  
    printf("I like programming\n");  
    printf("I love puppy!\n");  
    printf("I am so happy\n");  
    return 0;  
}
```

실행결과

```
I like programming  
I love puppy!  
I am so happy
```

- ▣ printf 함수는 첫 번째 인자로 전달된 문자열을 출력

# printf함수와 특수문자(Escape Sequence)

```
printf("앞집 강아지가 말했다. "멍~!  
멍~!" 정말 귀엽다");
```

잘못된 호출



"앞집 강아지가 말했다. " → 문자열  
멍~! 멍~! → 모르는 것  
" 정말 귀엽다" → 문자열

컴파일러의 오해



```
printf("앞집 강아지가 말했다. \"멍~!  
멍~!\" 정말 귀엽다");
```

제대로 호출

- 문자열 안에 큰 따옴표를 넣고 싶으면 \" 형태로 표시해줄 것

# Escape Sequence

Escape Sequence	의미하는 바
\a	경고음
\b	백스페이스(backspace)
\f	폼 피드(form feed)
\n	개행(new line)
\r	캐리지 리턴(carriage return)
\t	수평 탭
\v	수직 탭
\'	작은 따옴표 출력
\"	큰 따옴표 출력
\?	물음표 출력
\\	역슬래시 출력

\f와 \v는  
프린터 출력을  
위해 정의  
된 것이므로,  
모니터 출력  
에 사용하면  
이상한 문자  
출력

# printf함수의 서식지정과 서식문자들

```
int main(void) {  
    int myAge = 12;  
    printf("제 나이는 10진수로 %d살, 16진수로 %X 살입  
니다.\n", myAge, myAge);  
    return 0;  
}
```

실행결과

제 나이는 10진수로 12살, 16진수로 C살입니다.

- ▣ 서식 문자를 이용해서 출력할 문자열의 형태를 조합 가능
- ▣ 즉 출력의 서식을 지정할 수 있음

# printf함수의 서식지정과 서식문자들

```
int main(void) {  
    int num1 = 7, num2 = 13;  
    printf("%o %#o\n", num1, num1);  
    printf("%x %#x\n", num2, num2);  
    return 0;  
}
```

실행결과

7 07

d 0xd

- ▣ #을 삽입하면 8진수 앞에 0, 16진수 앞에 0x가 삽입된다

서식문자	출력 대상(자료형)	출력 형태
%d	char, short, int	부호 있는 10진수 정수
%ld	long	부호 있는 10진수 정수
%lld	long long	부호 있는 10진수 정수
%u	unsigned int	부호 없는 10진수 정수
%o	unsigned int	부호 없는 8진수 정수
%x, %X	unsigned int	부호 없는 16진수 정수
%f	float, double	10진수 방식의 부동소수점 실수
%Lf	long double	10진수 방식의 부동소수점 실수
%e, %E	float, double	e또는 E 방식의 부동소수점 실수
%g, %G	float, double	값에 따라 %f와 %e 사이에서 선택
%c	char, short, int	값에 대응하는 문자
%s	char*	문자열
%p	void*	포인터의 주소 값



## 실수 출력을 위한 서식문자들: %f, %e

```
int main(void) {  
    printf("%f\n", 0.1234);  
    printf("%e\n", 0.1234); // e 표기법 기반의 출력  
    printf("%f\n", 0.12345678);  
    printf("%e\n", 0.12345678); // e 표기법 출력  
}
```

실행결과

0.123400

1.234000e-001

0.123457

1.234568e-001

**컴퓨터는 지수를 표현할 수 없으므로 e 표기법으로 지수를 대신 표현한다.**

## 실수 출력을 위한 서식문자들: %f, %e

$0.\underbrace{0000000000000000}_{\text{zeros}}1$

 $1.0 \times 10^{-20}$ 

지수 표기법

1.0 x e-20

## e 표기법

지수 표기법

$$1.2 \times 10^{+12}, 1.15 \times 10^{-12}, 1.7 \times 10^{-15}$$

1.2e+12, 1.15e-12, 1.7e-15

## e 표기법

# %g의 실수 출력

	실행결과
<code>int main(void) {</code>	
<code>double d1 = 1.23e-3; /* 0.00123 */</code>	0.00123
<code>double d2 = 1.23e-4; /* 0.000123 */</code>	0.000123
<code>double d3 = 1.23e-5; /* 0.0000123 */</code>	1.23e-005
<code>double d4 = 1.23e-6; /* 0.00000123 */</code>	1.23e-006
<code>printf("%g\n", d1) /* %f 스타일 출력 */</code>	
<code>printf("%g\n", d2) /* %f 스타일 출력 */</code>	
<code>printf("%g\n", d3) /* %e 스타일 출력 */</code>	
<code>printf("%g\n", d4) /* %e 스타일 출력 */</code>	
<code>}</code>	

- ▣ %g는 실수 형태에 따라서 %f와 %e 사이에서 적절한 형태의 출력을 진행 (%G는 E를 대문자로 출력)

## %s의 문자열 출력

```
int main(void) {  
    printf("%s, %s, %s\n", "AAA", "BBB", "CCC");  
    return 0;  
}
```

실행결과

AAA, BBB, CCC

- %s의 문자열 출력과 관련해서는 배열과 포인터 공부 후에 완벽히 이해하자
- 일단은 %s의 사용법을 예제 기반으로 이해

# 필드 폭을 지정하여 정돈된 출력 보이기

**%8d**

필드 폭을 8칸 확보하고, 오른쪽 정렬해서 출력을 진행한다.

**%-8d**

필드 폭을 8칸 확보하고, 왼쪽 정렬해서 출력을 진행한다.

# 필드 폭을 지정하여 정돈된 출력 보이기

```
int main(void) {  
    printf("%-8s %14s %5s\n", "이 름", "전공학과", "학년");  
    printf("%-8s %14s %5s\n", "김동수", "전자공학", "3");  
    printf("%-8s %14s %5s\n", "이을수", "컴퓨터공학", "3");  
    printf("%-8s %14s %5s\n", "한선영", "미술교육학", "4");  
}
```

실행결과

이 름	전공학과	학년
김동수	전자공학	3
이을수	컴퓨터공학	2
한선영	미술교육학	4

# 필드 폭을 지정하여 정돈된 출력 보이기

---

- ▣ 서식문자 사이에 들어가는 숫자는 필드의 폭을 의미
- ▣ 기본 오른쪽 정렬, 따라서 - 는 왼쪽 정렬을 의미하는 용도로 사용됨

# 정수 기반의 입력형태 정의

입력의 형식 어떻게 받아들일 거니?

입력의 장소 어디에 저장할까?

**%d** 10진수 정수의 형태로  
데이터를 입력 받는다.

**%o** 8진수 양의 정수의 형태로  
데이터를 입력 받는다.

**%x** 16진수 양의 정수의 형태로  
데이터를 입력 받는다.

데이터를 입력 받는  
`scanf` 함수에게 전달해야  
할 두 가지 정보

서식문자의 의미는 출력을  
입력으로만 변경하면  
`printf` 함수와 유사하다.



# 정수 기반의 입력형태 정의

```
int main(void) {  
    int num1, num2, num3;  
    printf("세 개의 정수 입력: ");  
    scanf("%d %o %x", &num1, &num2, &num3);  
    printf("입력된 정수 10진수 출력: ");  
    printf("%d %d %d\n", num1, num2, num3);  
    return 0;  
}
```

실행결과

세 개의 정수 입력: 12 12 12

입력된 정수 10진수 출력: 12 10 18

# 실수 기반의 입력형태 정의

## float형 데이터의 삽입을 위한 서식문자

printf 함수에서는 서식문자 %f, %e 그리고 %g의 의미가 각각 달랐다.

그러나 scanf 함수에서는 'float형 데이터를 입력 받겠다'는 동일한 의미를 담고 있다.

## double형 long double형 데이터의 삽입을 위한 서식문자

%lf	double	%f에 l이 추가된 형태
-----	--------	---------------

%Lf	long double	%f에 L이 추가된 형태
-----	-------------	---------------

float, double, long double의 데이터 출력	%f, %f, %Lf
------------------------------------	-------------

float, double, long double의 데이터 입력	%f, %lf, %Lf
------------------------------------	--------------

# 실수 기반의 입력형태 정의

```
int main(void) {  
    float num1;  
    double num2;  
    long double num3;  
    printf("실수 입력1(e 표기법): ");  
    scanf("%f", &num1);  
    printf("입력된 실수 %f\n", num1);  
    printf("실수 입력 2(e 표기법): ");  
    scanf("%lf", &num2);  
    printf("입력된 실수 %f\n", num2);  
}
```

# 실수 기반의 입력형태 정의

```
printf("실수 입력 3(e 표기법): ");  
scanf("%Lf", &num3);  
printf("입력된 실수 %Lf\n", num3);  
return 0;  
}
```

실행결과

실수 입력1(e 표기법으로): 1.1e-3

입력된 실수 0.001100

실수 입력2(e 표기법으로): 0.1e+2

입력된 실수 10.000000

실수 입력3(e 표기법으로): 0.17e-4

입력된 실수 0.000017

- 실수의 입력과정에서 e표기법 사용 가능

# 정리

---

- printf()
- escape sequences
- conversion character
- scanf()
- conversion character