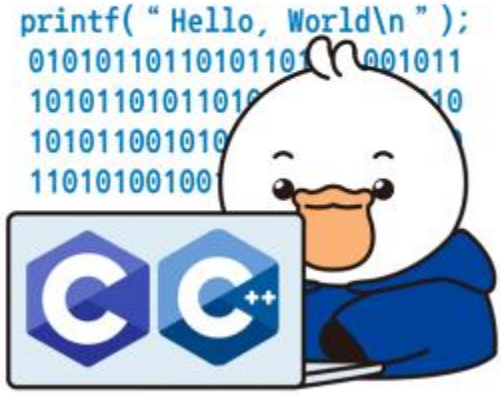
학번 : _____ 성명 : _____.

Part 3. 입출력과 연산자

목차



3.1 printf()함수

- printf() 함수란?
  - ✓ 표준 출력 stdout(standard output)을 이용하여 출력하는 함수이다.
    printf()의 f는 [ ]의 약자이다. formatted
    즉 지정 서식에 따라 출력하는 함수이다.

Printf	함수 원형	int printf(const char * restrict format, ...);	
	함수인자	포맷 문자열	%d, %c ....
		출력 목록	가변적이란 의미로 ...로 표현
	반환값	출력에 성공한 문자 수, 에러 발생시 음수 값 반환	

3.1 printf()함수

소스 3-1

```
1 #include <stdio.h>
2 int main(void)
3 {
4    int x =3;
5    int y = x +10;
6    printf("%d에 10을 더하면 %d입니다.\n", x, y);
7    return 0;
8 }
```

출력 예

```
3에 10을 더하면 13입니다.
```


3.1 printf()함수

소스 3-2

```
1 #include <stdio.h>
2 int main(void)
3 {
4    int x =10;
5    printf("%d대에는 \"공부\"를 열심히 해야 할 시기이다.\n", x);
6    return 0;
7 }
```

출력 예

```
10대에는 "공부"를 열심히 해야 할 시기이다.
```

3.1 printf()함수

- 제어문자란?

✓포맷 문자열 내부에서 □을 설정 할 수 있도록 돕는 역할

➢출력 제어를 위한 용도로 사용

\a	경고음(alert)	청각적 또는 시각적인 경고음 발생
\b	백스페이스	출력 위치를 직전 위치로 이동
\f	페이지 바꿈	페이지를 바꾸고 다음 페이지 맨 앞으로 이동
\n	**줄 바꿈**	줄 바꿈하고 다음 줄 맨 앞으로 이동
\r	캐리지 리턴	현재 줄의 맨 앞 위치로 이동
\t	**수평 탭**	다음 수평 탭 위치로 이동
\v	수직 탭	다음 수직 탭 위치로 이동
****	**백슬래시 ₩**	백슬래시 출력
\'	**작은따옴표 '**	작은따옴표 출력
\"	**큰따옴표 "**	큰따옴표 출력

<제어 문자의 종류>

3.1 printf()함수

소스 3-3

```
1 #include <stdio.h>
2 int main(void)
3 {
4    printf("Hello world\v");
5    printf("SnSlab\t");
6    printf("Hello daegu\n");
7    printf("Hello");
8    return 0;
9 }
```

출력 예

```
Hello world
          SnSlab        Hello daegu
Hello
```

3.1 printf()함수

소스 3-4

```
1 #include <stdio.h>
2 int main(void)
3 {
4    printf("Hello world\vSnSlab\tHello daegu\nHello"");
5    return 0;
6 }
```

출력 예

```
Hello world
           SnSlab   Hello daegu
Hello
```


3.1 printf()함수

<C 언어에서 사용할 수 있는 기호의 종류>

특수기호	영어	한국어	비고
:	Colon	콜론 (쌍점)	
;	Semicolon	세미콜론 (쌍반점)	
/	Slash	슬래시, 빗금	
\	Back Slash	백슬래시, 역슬래시, 역사선	₩로도 사용
@	At Sigh	엣 사인 또는 엣, 골뱅이표	
&	Ampersand	앰퍼센드	앤드(and)라고도 함
'	Aposterophe	아포스트로피, 작은따옴표	
`	Grave	그레이브, 억음 부호	
-	Hyphen	하이픈, 붙임표	
<	Left Angle Braket, Less than	레프트 앵글 브라킷, 작다, 화살괄호 열기	
>	Right Angle Bracket Bigger than	라이트 앵글 브라킷, 크다, 화살괄호 닫기	
{	Left Brace	레프트 브레이스, 중괄호 열기	
}	Right Brace	라이트 브레이스, 중괄호 닫기	

3.1 printf()함수

<C 언어에서 사용할 수 있는 기호의 종류>

특수기호	영어	한국어	비고
[	Left Braket	레프트 브라킷, 대괄호 열기	
]	Right Braket	라이트 브라킷, 대괄호 닫기	
|	Vertical Bar, Pipe	버티컬 바, 파이프, 수직선	
*	Asterisk	아스테리스크, 별표	
"	Quotation Mark	쿼테이션 마크, 쌍 따옴표, 큰 따옴표	
!	Exclamation Point	익스클레메이션 포인트, 느낌표	
#	Hash	해시, 샵, 우물정자	
.	Period or Dot	피리어드 또는 닷, 마침표, 온점	
~	Tilde	틸드, 물결표	
^	Caret	캐럿, 삿갓표, 곡절부호	
%	Percent Sign Modulo, Remainder	퍼센트 사인, 퍼센트 모듈로, 리마인더, 나머지	
$	Dollar Sign	달러 사인	
=	Equal Sign	이퀄 사인, 등호	

3.2 형식 지정자

- 형식 지정자(conversion specifier, 변환 지정자)란?
  - ✓포맷 문자열 내에서 인자값(argument)들과 1:1 대응하여 값을 출력
  - ✓f로 끝나는 대부분의 입출력 스트림(stream) 관련 함수의 [ ] 문자열에 사용

    포맷
  - ✓printf()와 scanf()에서 주로 사용
- 형식 지정자 사용방법
  - ✓fprintf() 함수 기준 설명

    파일 포인터
    - ➢fprintf() 함수 파라미터 중 [ ]를 제외한 것이 printf()

3.2 형식 지정자

소스 3-5

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int i = 3;
5      printf("%d\n", i);
6      return 0;
7  }
```

대응

출력 예

```
3
```


3.2 형식 지정자

- 형식 사양(Conversion Specification)
  - ✓%d와 같은 문자
  - ✓형식 지정자와 같은 의미이나 두 용어 모두 혼용하여 사용됨
  - ✓%와 형식 지정자 사이 다양한 옵션 선택적 추가 가능
  - ✓필드 길이는 값 전체 자릿수 (출력값)
  - ✓ 정밀도는 전체 자릿수 중 소숫점 이하 숫자의 수

%[플래그][필드 길이].[정밀도][길이 수정자]형식지정자

%[flags][field width].[precision][length modifier]conversion specifier

정렬 및 부호표시 지정 | 전체 출력 폭 지정 | 소수점 아래 부분 출력 폭 지정 | 인자의 크기를 지정

<형식 사양 정의>

3.2 형식 지정자

%[플래그][필드 길이].[정밀도][길이 수정자]형식지정자
%[flags][field width].[precision][length modifier]conversion specifier

<플래그의 용도와 사용법>

플래그	설명
-	대체 될 값을 출력 할 때 좌측으로 정렬 (없으면 기본값인 우측 정렬)
+	음수를 표현하는 –기호 뿐만아니라 양수에 +기호를 표시 (음수는 항상 표시)
공백문자	부호 기호(+/-)가 첫문자가 아닌 경우, 출력 문자 앞에 공백 추가 (공백문자와 + 기호를 동시에 사용하는 경우 공백문자는 무시됨)
#	대체 형식으로 변환하여 출력. - 형식 지정자가 o인 경우, 정밀도(소수점 이하 자릿수)를 높여 첫 숫자를 0으로 맞춤. 만약 값과 소숫점 이하 숫자 모두 0인 경우 0.0이 출력. - 형식 지정자가 x 또는 X인 경우, 출력값이 0이 아니라면 접두사 0x(또는 0X)를 추가 - 형식 지정자가 a 또는 A, e, E, f, F, g, G 라면, 항상 소숫점을 출력 - 형식 지정자가 g 또는 G인 경우, 숫자 뒷부분의 0을 그대로 출력 이외의 형식 지정자는 정의되지 않음
0	앞쪽 공백 출력 대신 0을 출력하여 자릿수 맞춤 - 형식 지정자가 d, i, o, u, x, X, a, A, e, E, f, F, g, 또는 G 라면, 0이 공백을 대신해 출력 (- 플래그와 동시에 사용되면 무시됨) 형식 지정자가 d, i, o, u, x, 또는 X 이고, 정밀도가 지정되어 있는 경우에는 무시됨. 이외의 형식 지정자는 정의되지 않음

<플래그 요약>

플래그	설명
-	지정된 폭에서 좌측정렬
+	부호 +, - 표시
#	우측정렬의 경우, 남은 폭 0으로 채우기
0	접두사 붙이기

3.2 형식 지정자

%[플래그][필드 길이].[정밀도][길이 수정자]형식지정자
%[flags][field width].[precision][length modifier]conversion specifier

<길이 수정자의 종류>

길이 수정자	설명
hh	형식 지정자가 d, i, o, u, x 또는 X 일 때, signed char 또는 unsigned char 인수에 적용하도록 지정 (값의 범위는 형식 지정자의 범위를 따름) 형식 지정자가 n일 때, signed char 포인터 인수에 적용
h	형식 지정자가 d, i, o, u, x 또는 X 일 때, short int 또는 unsigned short int 인수에 적용하도록 지정 (값의 범위는 형식 지정자의 범위를 따름) 형식 지정자가 n인 경우에는 short int 포인터 인수에 적용
l	형식 지정자가 d, i, o, u, x 또는 X 일 때, long int 또는 unsigned long int 인수에 적용하도록 지정 형식 지정자가 n인 경우에는 long int 포인터 인수에 적용
ll	형식 지정자가 d, i, o, u, x 또는 X 일 때, long long int 또는 unsigned long long int 인수에 적용하도록 지정 형식 지정자가 n인 경우에는 long long int 포인터 인수에 적용

3.2 형식 지정자

%[플래그][필드 길이].[정밀도][길이 수정자]형식지정자
%[flags][field width].[precision][length modifier]conversion specifier

<길이 수정자의 종류>

길이 수정자	설명
j	형식 지정자가 d, i, o, u, x 또는 X 일 때, intmax_t 또는 uintmax_t 인수에 적용하도록 지정 형식 지정자가 n인 경우에는 intmax_t 포인터 인수에 적용
z	형식 지정자가 d, i, o, u, x 또는 X 일 때, size_t 또는 상응하는 부호 있는 정수형 데이터 타입 인수에 적용하도록 지정 형식 지정자가 n인 경우에는 size_t 포인터 인수에 적용
t	형식 지정자가 d, i, o, u, x 또는 X 일 때, ptrdiff_t 또는 상응하는 부호 없는 정수형 데이터 타입 인수에 적용하도록 지정 형식 지정자가 n인 경우에는 ptrdiff_t 포인터 인수에 적용
L	형식 지정자가 a, A, e, E, f, F, g 또는 G 일 때, long double 인수에 적용하도록 지정

3.2 형식 지정자

%[플래그][필드 길이].[정밀도][길이 수정자]형식지정자
%[flags][field width].[precision][length modifier]conversion specifier

<형식 지정자의 종류>

형식 지정자	설명
d, i	int형 인자를 출력하기 위한 형식 지정자로 10진 정수로 표현한다.
o, u, x, X	unsigned int형 인자를 출력하기 위한 것으로 8진수(o), 10진수(u), 16진수 (x 또는 X)를 의미한다. 16진수에서 x는 출력 글자를 소문자로 출력하고, X는 대문자로 출력한다.
f, F	부동 소숫점을 대표하는 double형을 10진수로 표현한다. 정밀도가 지정되지 않으면 소숫점 이하 6자리를 기본으로 출력한다. 정밀도가 0이고 # 플래그가 명시되지 않은 경우에는 소숫점을 출력하지 않는다. 만약 소숫점을 출력하는 경우에는 최소 1자리의 숫자가 소숫점 이전에 출력된다. 소숫점 이하 출력의 마지막 숫자에서 반올림되어 출력된다. 무한을 뜻하는 inf 또는 infinity와 숫자가 아님을 뜻하는 nan을 f일때는 소문자로, F일때는 대문자로 출력한다.
e, E	부동소수점을 대표하는 double형을 *[-]d.dddE±dd*와 같은 형태의 지수로 표현한다. 정밀도가 지정되지 않으면 소수점 이하 6자리를 기본으로 출력한다. 정밀도가 0이고 # 플래그가 명시되지 않은 경우에는 소수점을 출력하지 않는다. 소수점 이하 출력의 마지막 숫자에서 반올림되어 출력된다. e와 E는 지수 표현을 소문자 e로 할지 대문자 E로 할지 결정한다.

3.2 형식 지정자

%[플래그][필드 길이].[정밀도][길이 수정자]형식지정자
%[flags][field width].[precision][length modifier]conversion specifier

<형식 지정자의 종류>

형식 지정자	설명
g, G	g일때는 f와 e, G일때는 F와 E 중 값과 정밀도에 따라 보기 편한 방식을 선택하여 double 형을 표현한다.
a, A	*[-]0xh.hhhh p±d*와 같은 형태의 16진수 부동소수점으로 표현한다.
c	int형을 unsinged char로 변환하여 한 글자를 표현한다.
s	char 포인터나 char 배열 인수를 문자열로 표현한다.
p	void형 포인터 인수를 받아 주솟값을 사전에 정의된 형태로 출력한다.
n	지금까지 출력 스트림으로 출력한 문자의 수를 기록하고 있는 정수에 대한 포인터이다. 실제 인수가 변환 되지는 않지만, 사용 된 것으로 간주한다. 플래그나 필드 길이, 정밀도가 주어지면 동작하지 않는다.
%	형식 사양의 시작을 알리는 문자이기 때문에, 출력을 위해서 정의되었다. %%로 사용하면 이는 %를 하나 출력한다. 인수는 소모되지 않는다.

3.2 형식 지정자

소스 3-6

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4     int i = 1004;
 5     int j = 356;
 6     int k = 4856;
 7     printf("[%d]\n", i);       // [1004]
 8     printf("[%+09d]\n", i);    // [+00001004]
 9     printf("[%+9d]\n", i);     // [    +1004]
10     printf("[%-09d]\n", i);    // [1004     ]
11     printf("[%9o]\n", j);      // [      544]
12     printf("[%-9o]\n", j);     // [544      ]
13     printf("[%+9o]\n", j);     // [      544]
14     printf("[%-9x]\n", k);     // [12f8     ]
15     printf("[%+9x]\n", k);     // [     12f8]
16     return 0;
17 }
```

<정수형 형식 지정자 예시>

형식 사양	정수	결과
%d	1004	1004
%+6i	1004	+1004
%-#6o	075	075
%0#6x	0x9a	0x009a

출력 예

```
[1004]
[+00001004]
[    +1004]
[1004     ]
[      544]
[544      ]
[      544]
[12f8     ]
[     12f8]
```

3.2 형식 지정자

소스 3-7

```
1  #include <stdio.h>
2  int main(void)
3  {
4     double x =3.141592;
5     printf("%f\n", x);
6     printf("%12.3f\n", x);
7     printf("%012.3f\n", x);
8     printf("%e\n", x);
9     printf("%+12.4e\n", x);
10    printf("%-12.3E", x);
11    return 0;
12 }
```

<실수형 형식 지정자 예시>

형식 사양	실수	결과
%f	2.8745	2.874500
%010.3f	2.8745	0000002.874
%e	2.8745	2.874500e+00
%-12.3E	2.8745	2.874E+00

출력 예

```
3.141592
       3.142
00000003.142
3.141592e+00
 +3.1416e+00
3.142E+00
```


3.2 형식 지정자

<문자열 형식 지정자 예시>

형식 사양	문자열	출력 결과
%s	"Hello World!!!"	Hello World!!!
%17.4s	"Hello World!!!"	Hell
%-17.14s	"Hello World!!!"	Hello World!!!
%17.12s	"Hello World!!!"	Hello World!
%19.5s	"Hello World!!!"	Hello

소스 3-8

```
1  #include <stdio.h>
2  int main()
3  {
4      printf("운송 수단은 %s, %s, %s, %s, 등이 있다.","자동차","비행기","배","기차");
5  }
```

출력예: 운송 수단은 자동차, 비행기, 배, 기차, 등이 있다.

3.3 scanf()함수

- scanf() 함수란?
  - ✓ 표준 [ ]인 키보드 이용 변수에 데이터를 저장하기 위해 사용
  - ✓ printf()와 동일하게 <stdio.h> 파일에 정의
  - ✓ fscanf() 함수에서 첫번째 인자를 [ ]으로 정의 시 scanf()와 동일

입력

stdin

scanf	함수 원형	int scanf(const char * restrict format, ...);	
	함수인자	포맷 문자열	%d, %c ....
		주소 인자	&
	반환값	첫 번째 입력 전 에러 발생시 EOF 반환,	

3.3 scanf() 함수

소스 3-9

```
1 #include <stdio.h>
2 int main(void)
3 {
4    int a;
5    scanf("%d", &a);
6    printf("입력값은 %d이다.", a);
7    return 0;
8 }
```

대응

```
#define _CRT_SECURE_NO_WARNINGS
```

일부 (Visual Studio 등) 개발 환경의 경우 소스코드 전처리 부분에 추가 시 경고 없이 프로그램 컴파일 및 실행 가능

입력 예	10
출력 예	입력 값은 10이다.

3.3 scanf() 함수

소스 3-10

```
1  #include <stdio.h>
2  int main(void) {
3      int i;
4      double d;
5      float f;
6      char ch;
7      scanf("%c", &ch);
8      scanf("%d",&i);
9      scanf("%lf", &d);
10     scanf("%f", &f);
11     printf("10진수 %d은 16진수 %x이다.\n", i, i);
12     printf("실수형 자료형 float은 %f이다.\n", f);
13     printf("실수형 자료형 double은 %f이다.\n", d);
14     printf("입력한 문자는 %c이다.\n", ch);
15     return 0;
16 }
```

<scanf의 형식 지정자 종류>

형식지정자	의미
%d	정수를 10진수로 입력
%x	정수를 16진수로 입력
%i	정수를 10진수, 8진수, 16진수로 입력
%f	float형 입력
%lf	double형 입력
%c	char형 입력
%s	문자열 입력

입력 예	C 100 3.14 6.25
출력 예	10진수 100은 16진수 64이다. 실수형 자료형 float은 6.250000이다. 실수형 자료형 double은 3.140000이다. 입력한 문자는 C 이다.

3.3 scanf() 함수

소스 3-11

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int i;
5      double d;
6      float f;
7      printf("세개의 수 입력 : ");
8      scanf("%d %lf %f", &i, &d, &f);
9      printf("첫 번째 입력받은 수는 %d이다.\n", i);
10     printf("두 번째 입력받은 수는 %f이다.\n", d);
11     printf("세 번째 입력받은 수는 %f이다.", f);
12     return 0;
13 }
```

입력 예	12 3.7 7.8
출력 예	첫 번째 입력받은 수는 12이다. 두 번째 입력받은 수는 3.700000이다. 세 번째 입력받은 수는 7.800000이다.

3.3 scanf() 함수

소스 3-12

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i;
5     double d;
6     float f;
7     printf("세 개의 수 입력 : ");
8     scanf("%d,%lf,%f", &i, &d, &f);
9     printf("첫 번째 입력받은 수는 %d이다.\n", i);
10    printf("두 번째 입력받은 수는 %.1f이다.\n", d);
11    printf("세 번째 입력받은 수는 %.1f이다.", f);
12    return 0;
13 }
```

입력 예	12,3.7,7.8
출력 예	첫 번째 입력받은 수는 12이다. 두 번째 입력받은 수는 3.7이다. 세 번째 입력받은 수는 7.8이다.

3.4 연산자

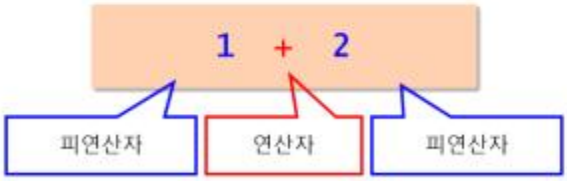
- 연산자란? 연산식
  - ✓ C 언어에서의 [ ]은 다른 말로 표현식(expressions)
  - ✓ 수학에서의 [ ]와 유사한 형태 연산자
    - ➢ 키보드 입력 가능한 연산 기호 제약으로 인해 다른 형태도 존재
  - ✓ 연산자마다 [ ] 존재 우선순위
    - ➢ [ ] 기억 및 정확한 연산식의 필요성 우선순위

연산식의 예

```
1 1 + 2; //덧셈 연산의 결과는 정수값
2 x < 30; //비교 연산의 결과는 1(참) 또는 0(거짓)
3 3.14; //실수형 상수
4 'a'; //문자 상수
5 x; //변수 x도 값이 있다면 수식
6 printf("Hello World!!!");//출력한 문자의 개수 리턴
```

3.4 연산자

- 연산식
  - ✓ 연산자(operator)와 피연산자(operand)로 구성
  - ✓ 연산자의 종류에 따라 하나 이상의 [ ]를 필요
  - ✓ 두 개의 피연산자 – [ ]항(binary) 연산자
  - ✓ 세 개의 피연산자 – [ ]항(ternary)연산자



3.4 연산자

<피연산자 개수에 따른 연산자의 종류>

피연산자 개수에 따른 연산자의 종류		연산자
단항 연산자		+, -, ++, --, !, *, &, ~, sizeof
이항 연산자	산술	+, -, *, /, %
	대입	=, +=, -=, *=, /=, %=, >>=, <<=, &=, |=, ^=
	관계	>, <, >=, <=, ==, !=
	논리	&&, ||
	비트	&, |, ^, <<, >>
삼항 연산자		? :

3.4 연산자

<역할 기준으로 분류한 연산자의 종류>

연산자의 종류	연산자
산술 연산자	+, -, *, /, %, +(단항), -(단항)
증감 연산자	++(전위), ++(후위), --(전위), --(후위)
관계 연산자	>, <, >=, <=, ==, !=
논리 연산자	&&, ||, !
비트 연산자	&, |, ^, ~, <<, >>
대입 연산자	=, +=, *=, /=, %=, &=, |=, ^=, <<=, >>=
멤버 접근 연산자	*, &, [], ., ->
그 밖의 연산자	? :, ,, sizeof, ()

3.4 연산자

- 산술연산자
  - ✓ 더하기(+)와 빼기(-), 곱하기(*), 나누기(/) 사칙연산과 나머지(%) 등
  - ✓ 아래 표에서 변수 x의 값은 10, y의 값은 4
  - ✓ %는 나머지(remainder) 연산자 혹은 모듈러(Modular) 연산자

<산술 연산자의 역할 및 사용 예시>

산술 연산자	역할	산술 연산식 예시	연산 결과
+ (단항)	양수 부호	+x	10
- (단항)	음수 부호	-x	-10
+	더하기	x + y	14
-	빼기	x - y	6
*	곱하기	x * y	40
/	나누기	x / y	2
%	나머지 연산	x % y	2

3.4 연산자

소스 3-13

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int a = 11;
5     int b = 4;
6     double x = 11;
7     double y = 4;
8     printf("정수형 자료형을 피연산자로 나누기 연산 : %d\n", a/b);
9     printf("실수형 자료형을 피연산자로 나누기 연산 : %f\n", x/y);
10    printf("정수형 자료형을 피연산자로 나머지 연산 : %d\n", a%b);
11    return 0;
12 }
```

출력 예

```
정수형 자료형을 피연산자로 나누기 연산 : 2
실수형 자료형을 피연산자로 나누기 연산 : 2.750000
정수형 자료형을 피연산자로 나머지 연산 : 3
```


3.4 연산자

소스 3-14

```
1  #include <stdio.h>
2  int main(void) {
3     int a, b;
4     scanf("%d %d", &a, &b);
5     printf("+%d = %d\n", a, +a);
6     printf("-%d = %d\n", a, -a);
7     printf("%d + %d = %d\n", a, b,a + b);
8     printf("%d - %d = %d\n", a, b, a - b);
9     printf("%d / %d = %d\n", a, b, a / b);
10    printf("%d * %d = %d\n", a, b, a * b);
11    printf("%d %% %d = %d\n", a, b, a % b);
12    return 0;
13 }
```

입력 예 1

```
4 5
```

출력 예 1

```
+4 = 4
-4 = -4
4 + 5 = 9
4 - 5 = -1
4 / 5 = 0
4 * 5 = 20
4 % 5 = 4
```

입력 예 2

```
4 0
```

출력 예 2

```
+4 = 4
-4 = -4
4 + 0 = 4
4 - 0 = 4
Floating point exception
```

3.4 연산자

- 연산자 사용시 주의점
  - ✓ 나눗셈과 나머지 연산 시 나누는 수로 [ ] 사용 불가 (0)
  - ✓ 연산 데이터 타입이 서로 다른 경우
    - ➢ 보다 큰 수 또는 [ ] 수를 저장할 수 있는 데이터 타입으로 결과가 생성 (정밀한)
      - int형 값과 float형 값의 연산 결과는 [ ] 형 (float)

```
1.25 * 3    // double + double
1.25 + 0.5F // double + double
1.25F / 2   // float / float
'a' + 10L   // long + long
```

```
short x = 100;
123U - x          //unsigned int - unsigned int로 처리
123U + 1234567LL  // long long + long long으로 처리
```


3.4 연산자

소스 3-15

```
1 #include <stdio.h>
2 int main(void)
3 {
4    short a =10;
5    printf("+a의 크기는 %d이다.\n", sizeof(+a));
6    printf("-a의 크기는 %d이다.\n", sizeof(-a));
7    return 0;
8 }
```

출력 예

```
+a의 크기는 4이다.
-a의 크기는 4이다.
```

3.4 연산자

소스 3-16

```
1  #include <stdio.h>
2  #define PI 3.141592
3  int main(void)
4  {
5     int r;
6     double area, perimeter;
7     scanf("%d", &r);
8     area = PI * r * r;
9     perimeter = PI * r *2;
10    printf("원의 면적은 %.3f이다.\n", area);
11    printf("원의 면적은 %.3f이다.\n", perimeter);
12    return 0;
13 }
```

묵시적 형변환, 매크로 상수 PI의 값이 double형으로 인식

입력 예	12
출력 예	원의 면적은 452.389이다. 원의 둘레는 75.398이다.

3.4 연산자

- 증감 연산자

  ✓ 피연산자 ☐ 증가와 감소하는 두 가지 존재, 모두 단항 연산자.

  ✓ 연산자의 ☐ 에 따라 연산결과의 차이 발생

  ➢ 전위: 피연산자의 앞에 사용 (전치, prefix)

  ➢ 후위: 피연산자의 뒤에 사용 (후치, postfix)

<증감 연산자의 특징과 의미>

증감 연산자	의미
++x	피연산자의 값을 1 증가시킨 후 연산 진행
x++	연산을 먼저 수행 후, 피연산자의 값 1 증가
--x	피연산자의 값을 1 감소시킨 후 연산 진행
x--	연산을 먼저 수행 후, 피연산자의 값 1 감소

3.4 연산자

소스 3-17

```c
#include <stdio.h>
int main(void) {
   int i = 5;
   double d = 1.5;
   printf("%d\n", i++);
   printf("%d\n", i);
   printf("%d\n", --i);
   printf("%d\n", i);
   printf("%f\n", d--);
   printf("%f\n", d);
   printf("%f\n", ++d);
   printf("%f\n", d);
   return 0;
}
```

출력 예

```
5
6
5
5
1.500000
0.500000
1.500000
1.500000
```


3.4 연산자

- 대입 연산자 (Assignment Operator)
  - ✓ 변수에 값을 할당 또는 대입하기 위해 사용하는 이항 연산자
  - ✓ =는 수학에서 "같다"라는 의미로 사용, C 언어에서는 [ ] 연산자 대입
  - ✓ = 연산자의 좌측 피연산자를 l-value, 우측 위치 피연산자를 r-value
  - ✓ [ ]측에서 [ ]측으로 연산을 수행 우측 → 좌측
  - ✓ 반드시 단 하나의 변수만 [ ]변에 기입 좌변

3.4 연산자

<대입 연산자의 사용 예시>

증감 연산자	의미

바른 예	`int exch_rate = 1024, dollar = 100;`
	`int won = 0;`
	`won = dollar * exch_rate;`

그른 예	`10 = y;` //10변수가 아니므로 컴파일 에러 `x + 1 = x;` //x + 1은 변수가 아니므로 컴파일 에러 `printf("xyz") = 10;` //printf("xyz")는 변수가 아니므로 컴파일 에러

<축약 대입 연산자의 종류와 의미>

축약 대입 연산자의 예	동일한 결과의 연산식	축약 대입 연산자의 예	동일한 결과의 연산식
a+=b	a=a+b	a&=b	a=a&b
a-=b	a=a-b	a|=b	a=a|b
a*=b	a=a*b	a^=b	a=a^b
a/=b	a=a/b	a<<=b	a=a<<b
a%=b	a=a%b	a>>=b	a=a>>b

3.4 연산자

소스 3-18

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int images = 523;
 5    int pages = 0;
 6    int imagePerPage = 4;
 7    pages = images / imagePerPage;
 8    images %= imagePerPage;
 9    printf("%d페이지 출력 후 %d개의 이미지가 남았네요.\n", pages, images);
10    return 0;
11 }
```

출력 예

```
130페이지 출력 후 3개의 이미지가 남았네요.
```

3.4 연산자

- 관계(비교) 연산자
  - ✓ 두 개의 피연산자 값 [ ]를 통해 참(true), 거짓(false) 판단
  - ✓ C 언어에서 참(true)은 1, 거짓(false)은 0으로 표현
  - ✓ 값으로 참/거짓을 판단 시 0에 해당하는 모든 값은 거짓, 이외 모든 값은 참

<관계 연산자의 종류와 의미 >

관계 연산자 예	의미	결과
a==b	a와 b가 같은가?	1(참) 또는 0(거짓)
a!=b	a와 b가 다른가?	1(참) 또는 0(거짓)
a>b	a가 b보다 큰가?	1(참) 또는 0(거짓)
a<b	a가 b보다 작은가?	1(참) 또는 0(거짓)
a>=b	a가 b보다 크거나 같은가?	1(참) 또는 0(거짓)
a<=b	a가 b보다 작거나 같은가?	1(참) 또는 0(거짓)

3.4 연산자

소스 3-19

```
1  #include <stdio.h>
2  int main(void) {
3     int in1, in2;
4     scanf("%d %d", &in1, &in2);
5     printf("%d == %d : %d\n", in1, in2, in1==in2);
6     printf("%d != %d : %d\n", in1, in2, in1!=in2);
7     printf("%d > %d : %d\n", in1, in2, in1>in2);
8     printf("%d < %d : %d\n", in1, in2, in1<in2);
9     printf("%d >= %d : %d\n", in1, in2, in1>=in2);
10    printf("%d <= %d : %d\n", in1, in2, in1<=in2);
11    return 0;
12 }
```

입력 예	3 5
출력 예	3 == 5 : 0 3 != 5 : 1 3 > 5 : 0 3 < 5 : 1 3 >= 5 : 0 3 <= 5 : 1

3.4 연산자

- 논리 연산자
  - ✓ 참과 거짓 이용한 논리 연산 수행
    - ➢ &&(AND)와 ||(OR), !(NOT) 존재
    - ➢ □항 연산자: &&와 ||
    - ➢ □항 연산자: !

<논리 연산자의 종류 및 설명>

논리 연산자	설명
a && b	a와 b가 둘 다 참일때만 참(1), 그 외엔 모두 거짓(0)
a || b	a와 b가 둘 다 거짓이면 거짓(0), 그 외엔 모두 참(1)
!a	a가 거짓이면 참(1), a가 참이면 거짓(0)

<논리 연산자의 예시>

A	B	A&&B	A||B	!A	!B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

3.4 연산자

소스 3-19

```
1 #include <stdio.h>
2 int main(void)
3 {
4    char ch ='t';
5    int i =10;
6    printf("%c는 c보다 크고 w보다 작다. : %d\n", ch, (ch>'c') && (ch<'w'));
7    printf("%d은 5보다 작거나 100보다 크다. : %d\n", i, (i<5) || (i>100));
8    return 0;
9 }
```

출력 예

```
t는 c보다 크고 w보다 작다. : 1
10은 5보다 작거나 100보다 크다. : 0
```

3.4 연산자

소스 3-20

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int a =1;
5     int b =0;
6     printf("%d\n", --a && ++b);
7     printf("%d\n", --a || ++b);
8     printf("a = %d, b = %d\n", a, b);
9     return 0;
10 }
```

출력 예

```
0
1
a = -1, b = 0
```


3.4 연산자

- 비트 연산자
  - ✓ int형 정수를 2진수 비트 단위로 연산을 수행하는 연산자
  - ✓ 비트 이동(shift) 연산자 : 비트 단위 [ ] 이동
  - ✓ 비트 논리 연산자 : 비트 단위 논리 연산 수행
  - ✓ 단항 연산자 ~를 제외하고 모두 [ ]항 연산자

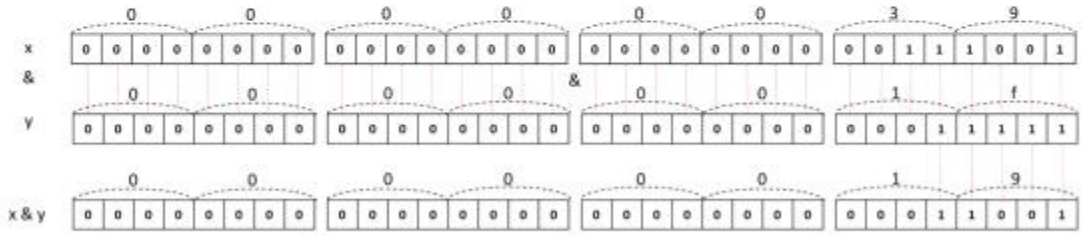
<비트 연산자의 구분 및 설명>

구분	비트 연산자	설명
비트 이동 연산자	x << y	x의 각 비트를 y만큼 왼쪽으로 이동
	x >> y	x의 각 비트를 y만큼 오른쪽으로 이동
비트 논리 연산자	x & y	x와 y의 각 비트 단위로 논리 AND 연산
	x | y	x와 y의 각 비트 단위로 논리 OR 연산
	x ^ y	x와 y의 각 비트 단위로 논리 XOR 연산
	~x	x의 각 비트 단위로 논리 NOT연산

3.4 연산자

- 비트 and 연산자 사용 방법 및 연산 수행 과정

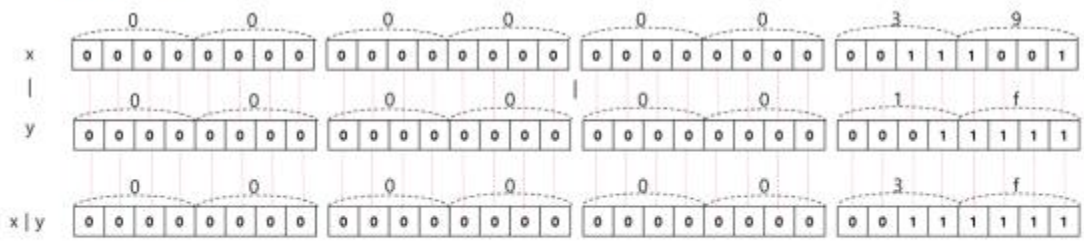
```
int x = 0x39;
int y = 0x1f;
int z = x & y;
```



3.4 연산자

- 비트 or 연산자 사용 방법 및 연산 수행 과정

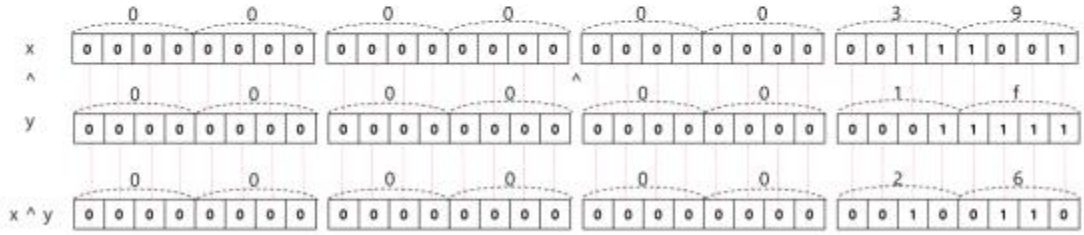
```
int x = 0x39;
int y = 0x1f;
int z = x | y;
```



3.4 연산자

- 비트 xor 연산자 사용 방법 및 연산 수행 과정

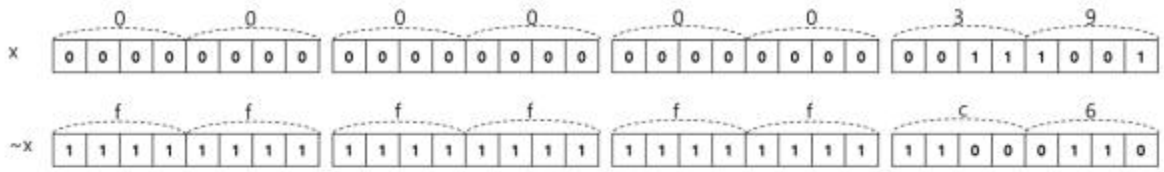
```
int x = 0x39;
int y = 0x1f;
int z = x ^ y;
```



3.4 연산자

- 비트 not 연산자 사용 방법 및 연산 수행 과정

```
int x = 0x39;
int z = ~x;
```



3.4 연산자

소스 3-21

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int x = 0x45;
 5    int y = 0xb4;
 6    printf("%0#8x & %0#8x = %0#8x\n", x, y, x & y);
 7    printf("%0#8x | %0#8x = %0#8x\n", x, y, x | y);
 8    printf("%0#8x ^ %0#8x = %0#8x\n", x, y, x ^ y);
 9    printf("~%0#8x = %0#8x", x, ~x);
10    return 0;
11 }
```

출력 예

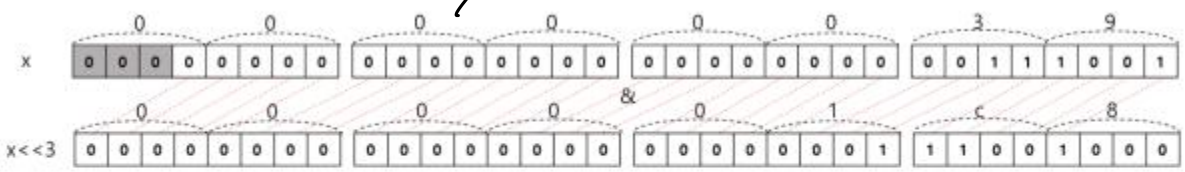
```
0x000045 & 0x0000b4 = 0x000004
0x000045 | 0x0000b4 = 0x0000f5
0x000045 ^ 0x0000b4 = 0x0000f1
~0x000045 = 0xffffffba
```


3.4 연산자

- 비트 이동 연산자 사용 방법 및 연산 수행 과정

✓ << 연산자의 동작

```
int x = 0x39;
int y = x << 3;
```

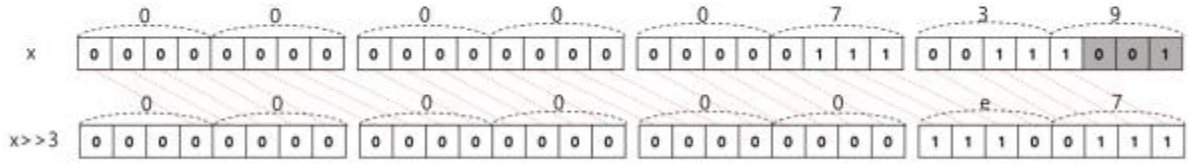


3.4 연산자

- 비트 이동 연산자 사용 방법 및 연산 수행 과정

✓ >> 연산자의 동작

```
int x = 0x739;
int y = x >> 3;
```



3.4 연산자

소스 3-22

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int x =0x0739;
 5    int y = x << 3;
 6    int z = x >> 3;
 7    printf("%0#8x >> 3 = %0#8x\n", x, z);
 8    printf("%0#8x << 3 = %0#8x\n", x, y);
 9    return 0;
10 }
```

출력 예

```
0x000739 >> 3 = 0x0000e7
0x000739 << 3 = 0x0039c8
```

3.4 연산자

- 조건 연산자
  - ✓ C언어 유일 [ ] 연산자
  - ✓ 조건 연산자는 ?:를 사용
    - ➢ 사용 예시 - a ? b : c
      - 일반적으로 a 위치에 조건식 사용
      - a의 값이 참 - b가 연산식 대푯값
      - a의 값이 거짓 - c가 대푯값

3.4 연산자

소스 3-23

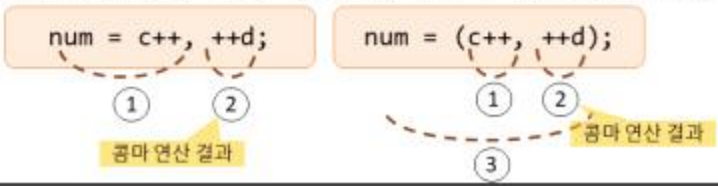
```
1  #include <stdio.h>
2  int main(void) {
3     int x =-5, y =8, z =10;
4     int max;
5     x = x > 0 ? x : -x;
6     max = x > y ? x : y;
7     printf("x의 절댓값 : %d\n", x);
8     printf("%d와 %d 중 큰 수는 : %d\n", x, y, max);
9     printf("%d, %d, %d은 삼각형 %s.\n", x, y, z, z<x+y?"이다":"이 아니다");
10    return 0;
11 }
```

출력 예

```
x의 절댓값 : 5
5와 8 중 큰 수는 : 8
5, 8, 10은 삼각형 이다.
```

3.4 연산자

- 콤마 연산자
  - ✓ 여러개의 연산식을 [ ]적으로 수행하고자 할 때 사용 순차적
  - ✓ , (콤마) 이용 연산식 여러개 작성
    - ➢ 각각의 연산식 [ ]측에서 [ ]측으로 개별적 수행 좌측 · 우측
      마지막의 연산 결과가 대푯값
  - ✓ 타 연산자에 비해 상대적으로 우선순위가 가장 [ ] 낮다.



3.4 연산자

소스 3-24

```
1 #include <stdio.h>
2 int main(void)
3 {
4    int a =1, b =2;
5    a = (++a, ++b);
6    printf("%d", a);
7    return 0;
8 }
```

출력 예 3

소스 3-25

```
1 #include <stdio.h>
2 int main(void)
3 {
4    int a =1, b =2;
5    a = ++a, ++b;
6    printf("%d", a);
7    return 0;
8 }
```

출력 예 2

3.4 연산자

- 형변환 연산자
  - ✓ 형변환(cast)이란 데이터 [ ]을 변경하는 것 타입
    - ➢ 두 가지 방식 존재
      - 묵시적(implicit) 형변환 암고 자동
        - 데이터 손실 가능성이 [ ] 연산에 필요한 경우 [ ] 수행
      - 명시적(explicit) 형변환
        - 프로그래머가 [ ]을 감수하고도 데이터 형을 변환하고자 하는 경우 수행 ← 데이터 손실

3.4 연산자

소스 3-26

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int i =321;
 5    double d =3.141592;
 6    char ch;
 7    ch = (char)i;
 8    i = (int)d;
 9    printf("%c\n", ch);
10    printf("%d\n", i);
11    return 0;
12 }
```

출력 예

```
A
3
```

3.4 연산자

- sizeof 연산자
  - ✓ 데이터 타입 크기를 확인하는 전위 □항 연산자 단항
  - ✓ 변수나 상수, 자료형 □에 사용 와도
  - ✓ 각 데이터 타입 크기를 □ 단위로 알려준다. 바이트(byte)
  - ✓ 예시)
    - ➢ 변수 i의 크기 확인 – sizeof i
    - ➢ double형 크기 확인 – sizeof (double)
  - ✓ 데이터 타입 이름으로 크기 확인 시 반드시 ()괄호 사용 필요

3.4 연산자

소스 3-27

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int i =123;
5      double d =3.141592;
6      char ch ='a';
7      printf("char형 변수 ch : %lubytes\n", sizeof ch);
8      printf("int형 변수 i : %lubytes\n", sizeof i);
9      printf("double형 변수 d : %lubytes\n", sizeof d);
10     printf("short : %lubytes\n", sizeof(short));
11     printf("long long : %lubytes\n", sizeof(long long));
12     printf("float : %lubytes\n", sizeof(float));
13     printf("long double : %lubytes\n", sizeof(long double));
14     return 0;
15 }
```

출력 예

```
char형 변수 ch : 1bytes
int형 변수 i : 4bytes
double형 변수 d : 8bytes
short : 2bytes
long long : 8bytes
float : 4bytes
long double : 16bytes
```

3.4 연산자

- 그 외의 연산자
  - ✓ 아래 연산자는 모두 □항 연산자
  - ✓ [], ., ->□후위 단항 연산자
  - ✓ *와 &는 □위 단항 연산자

<그 외의 연산자의 종류 및 설명>

연산자	연산자 이름	활용
[]	배열 첨자 연산자	배열에서 배열의 원소에 접근할 때 사용
.	직접 멤버 (접근) 연산자	구조체의 멤버 변수에 접근할 때 사용
->	간접 멤버 (접근) 연산자	구조체 포인터를 이용하여 멤버 변수에 접근할 때 사용
*	간접 연산자	주소를 이용하여 변수에 접근할 때 사용
&	주소 연산자	변수의 주소를 확인할 때 사용

3.4 연산자

- 연산자의 우선순위와 결합법칙
  - ✓ 정확한 연산식을 위해 연산자 [ ]와 [ ]의 정확한 이해의 필요
    - ➢ 연산자 우선순위 잘못 이해 시 의도치 않은 엉뚱한 결과 도출
  - ✓ 15단계의 우선순위 존재
    - ➢ 가장 높은 1에서 가장 낮은 15까지

3.4 연산자

<우선순위 및 결합법칙의 정리>

우선순위	연산자 유형	연산자	결합법칙
1	괄호, 후위 단항 연산자, 참조 연산자	() [] . -> ++ --	→
2	전위 단항 연산자	- ! ~ ++ -- (type) * & sizeof	←
3	곱하기, 나누기, 나머지 산술 연산자	* / %	→
4	더하기, 빼기 산술 연산자	+ -	
5	비트 이동 연산자	>> <<	
6	비교 관계 연산자	< <= > >=	
8	and 비트 논리 연산자	&	→
9	xor 비트 논리 연산자	^	
10	or 비트 논리 연산자	|	
11	and 논리 연산자	&&	
12	or 논리 연산자	||	
13	조건 연산자	? :	←
14	대입 연산자	= += -= /= *= %= &= ^= |= >>= <<=	←
15	콤마 연산사	,	→

3.4 연산자

소스 3-28

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int a, b, c, d, e;
5     a = b = c = d = e = 10;
6     printf("변수 a의 값은 %d\n", a);
7     return 0;
8  }
```

출력 예

```
변수 a의 값은 10
```

3.4 연산자

소스 3-29

```
1 #include <stdio.h>
2 int main(void)
3 {
4     double celsius, fahrenheit;
5     scanf("%lf", &celsius);
6     fahrenheit = celsius * (9.0/5.0) +32;
7     printf("섭씨 %.1f도는 화씨 %.1f도 입니다.\n", celsius, fahrenheit);
8     return 0;
9 }
```

입력 예 1	10.5
출력 예 1	섭씨 10.5도는 화씨 50.9도 입니다.
입력 예 2	31.5
출력 예 3	섭씨 31.5도는 화씨 88.7도 입니다.

3.4 연산자

소스 3-30

```
1  #include <stdio.h>
2  int main(void)
3  {
4     double wonDollarExchangeRate, dollar;
5     int won;
6     scanf("%lf %d", &wonDollarExchangeRate, &won);
7     dollar = won / wonDollarExchangeRate;
8     printf("%d원은 %.2f달러 입니다.\n", won, dollar);
9     return 0;
10 }
```

입력 예	1166.36 1000000
출력 예	1000000원은 857.37달러 입니다.

3.5 Q&A

Q 12. prinf()를 부를 때, 항상 (void)로 캐스팅하고 있는데, 왜 그럴까?"

A . 어떤 컴파일러는 함수가 어떤 값을 반환 하는데, 값을 사용하지 않으면, 경고를 출력한다 Printf()는 반환값이 존재하지만 대개 쓰지 않는다. 따라서 (void)로 캐스팅하여 반환 값 무시를 컴파일러에게 알려 경고를 출력하지 않게한다.

Q 13. 왜 scanf("%d",i)는 동작하지 않나?

A . Scanf() 전달 인자는 항상 포인터이다. 각각 값 변환시 scanf()는 전달된 포인터가 가리키는 위치에 값을 저장한다. 그렇기 때문에, scanf("%d",&i)로 써야한다..

3.5 Q&A

Q 14. 다음 코드는 왜 동작하지 않나?

```
double d;
scanf("%f", &d);
```

A . scanf()는 double 타입 사용시 %lf, float타입 사용시 %f를 쓴다. scanf()는 float를 가리키는 포인터를 받게된다. 따라서 double을 가리키는 포인터는 잘못된 것, %lf 또는 변수를 float 선언 바람

Q 15. 컴파일러로 다음과 같은 코드를 실행하면 :

```
int i =7;
printf("%d\n", i ++* i ++);
```

'49'를 출력한다. 평가 순서에 상관없이, '56'을 출력해야 하지 않을까?

A . 증가(++), 감소(--) 연산자가 뒤에 쓰일 때에는 먼저 기존의 값을 계산한 다음, 증가/ 감소하게 된다. 따라서 위 코드에서는 컴파일러가 기존의 값으로 곱한 다음, 증가시키기 때문에 저런 결과가 나오는 것

3.5 Q&A

Q 16. 괄호를 써서 평가 순서를 내가 원하는 대로 바꿀 수 있을까? 만약에 괄호를 쓰지 않더라고 알아서 되지 않나?

A. 연산자 우선 순위와 괄호는 수식 평가의 일부분만을 변경할 수 있다. 다음과 같은 수식에서

```
f() + g() * h()
```

세 개의 함수 중 어떤 함수가 먼저 호출될지는 알 수 없다. 즉, 우선 순위는, 평가에서 일부분 영향을 미치나 피연산자 평가 순서에 영향을 주지는 못한다. 부분식 평가 순서가 중요할 때는, 임시 변수를 선언 후 각각 다른 문자으로 나눠쓰는 것이 좋다.

3.6 실습 및 과제

- 실습 및 과제 진행
  - DCU Code : http://code.cu.ac.kr
  - 3 주차 실습
    - 코딩 3-1, 코딩 3-2, 코딩 3-3, 코딩 3-4 (p132-144)
  - 3 주차 과제
    - 3장 프로그래밍 연습 1-6번 (p153-154)

3.7 참고 문헌

- printf() 함수
  - https://ko.wikipedia.org/wiki/Printf
  - http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- 형식지정자
  - http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- scanf() 함수
  - http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- 연산자
  - http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- Q&A
  - http://cinsk.github.io/ko/cfaqs/index.html

E N D