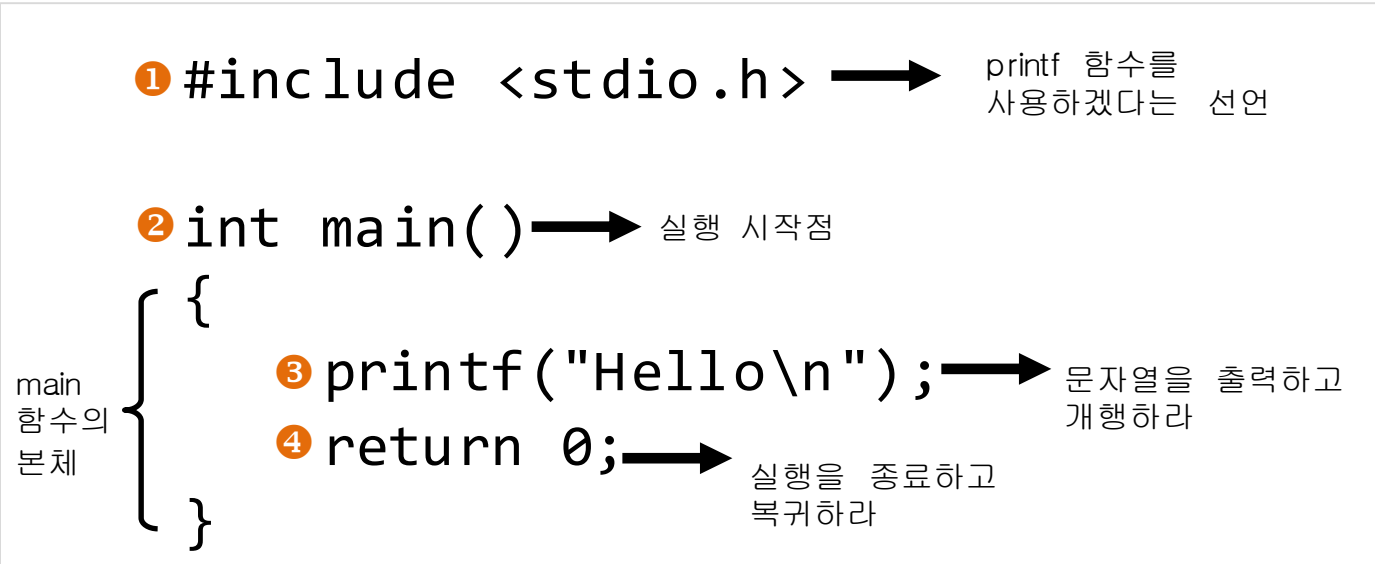


■ 소스의 구조



~~Wn~~ = \n

- 실질적인 명령은 printf 하나밖에 없다.
- 분석 후 변형해 본다. 여러 줄 출력 및 한글 출력도 가능하다.

- C 프로그램은 main 함수에서 시작하여 main함수로 종료
 - C 프로그램은 반드시 **하나의 main 함수를 포함** 해야 함
- main함수는 중괄호({) 로 시작하여 중괄호 (}) 로 끝남
- 모든 문장은 세미콜론으로 끝난다.

소스 코드

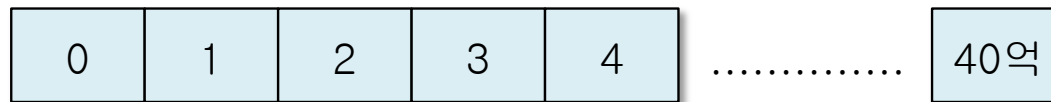
```
#include          // 필요한 헤더 파일 포함
#define           // 매크로 상수
함수 원형         // 함수 선언
전역 변수         // 동작에 필요한 변수 선언

int main()        // 프로그램의 시작점
{
    본체 코드
}
사용자 함수
사용자 함수
....
```

- C언어 프로그래밍에서 변수를 사용하려면 반드시 정의를 해야함
- 컴퓨터는 유한한 메모리 자원을 사용하고 있기 때문에 이걸 이렇고, 저건 저렇다고 일일이 정의를 해주어야 함
- 변수형을 정의할 때 “문자형”을 사용할지, “정수형” 을 사용할 지 어떻게 알 수 있지?
 - 가방의 크기를 정하는 것과 같음
 - 사용하고자 하는 값의 범위에 맞게 정의

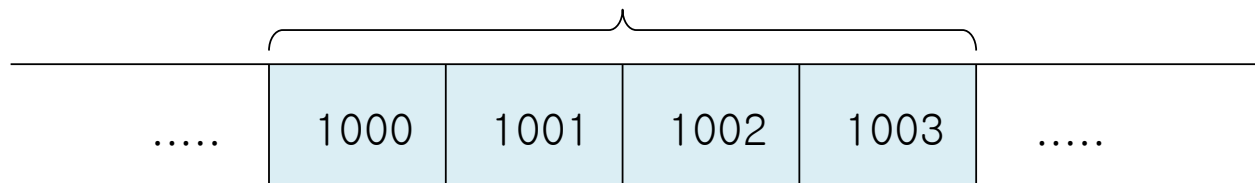
문자형	-128~+127 또는 0~255
정수형	-2,147,483,648~2,147,483,647 또는 0~4,294,967,295
실수형	1.7E-308 ~ 1.7E308

- 번지 : 메모리에 붙여 놓은 일련 번호



- 번지값에 이름을 붙여 놓은 것을 변수라고 한다.

이 메모리 영역을 price라고 선언한다.



- 번지보다는 다루기 쉽고 직관적이다.

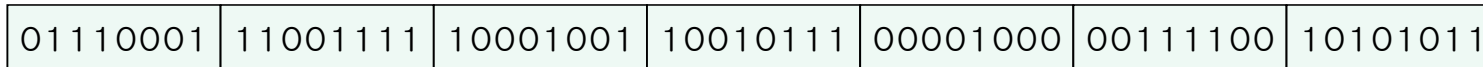
- $[0x3a28ff00] = [0xd800c0f8] + [0xdb10891a]$
- $salary = pay + bonus;$

- 키워드는 쓸 수 없다.
- 알파벳, 숫자, 밑줄로 구성된다. 공백은 쓸 수 없다.
- 첫 문자로 숫자는 쓸 수 없다.
- 대소문자를 구분한다.
 - 적합한 변수명 : price, sum, total_score, bonus4, health
 - 잘못된 변수명 : total score, price\$, 2ndlife
- 대상을 잘 표현하는 설명적인 이름을 붙인다. 여러 단어로 된 명칭은 밑줄이나 대문자를 활용한다.
 - get_total_score
 - GetTotalScore
- 좋은 이름을 붙이는 것은 굉장히 중요하고도 어려운 기술이다.

- 변수의 값을 읽으려면 위치, 길이, 형태에 대한 정보가 필요하다.

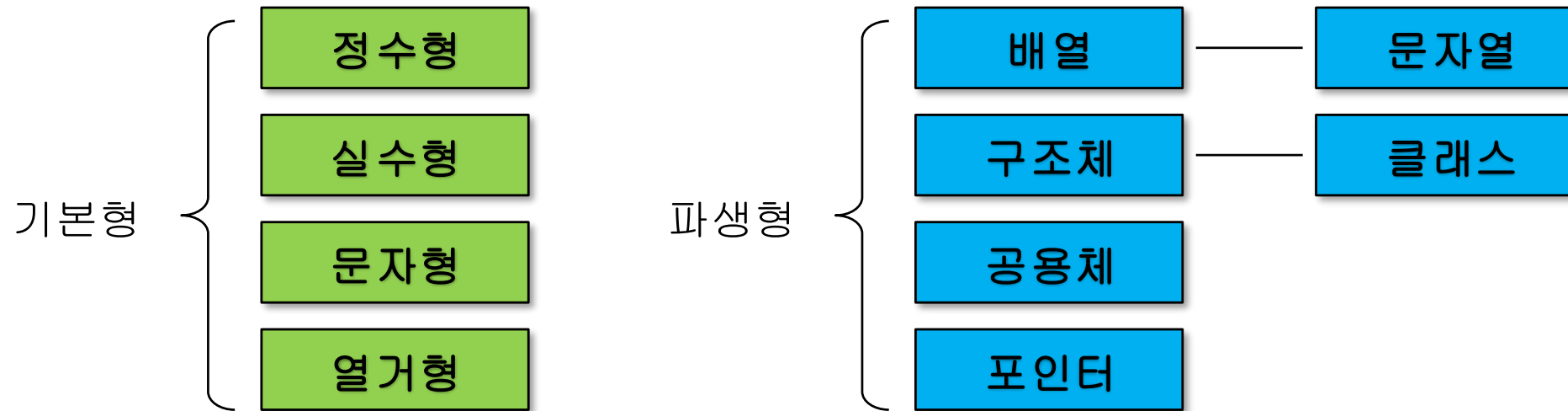
번지 = 변수의 시작 위치

길이 = 변수가 차지한 메모리의 양

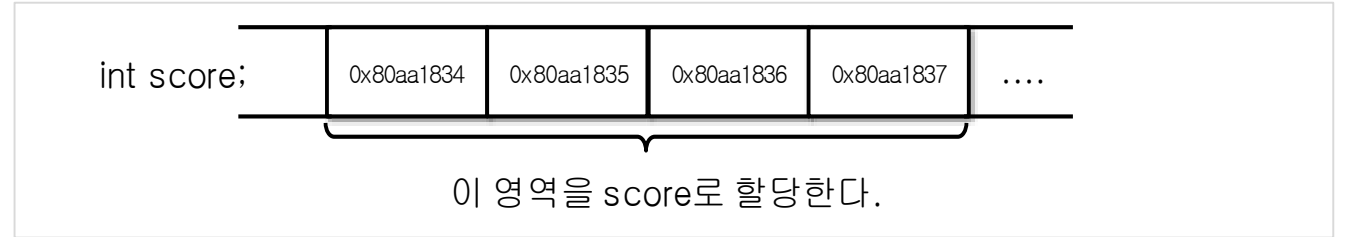


형태 = 메모리에 기억된 값을 해석하는 방식

- 타입 : 길이와 해석 방법에 대한 정보



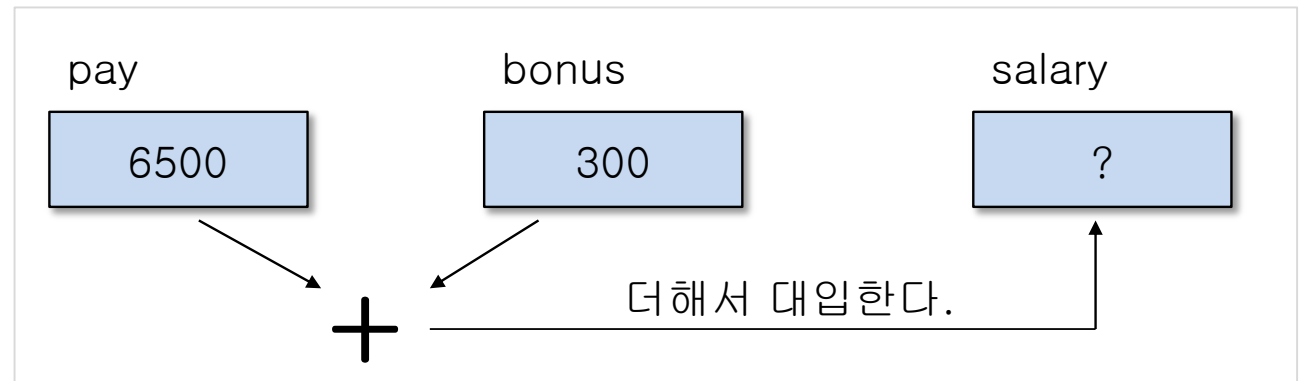
- 선언(Declaration)이란 컴파일러에게 사용할 변수를 알리는 것
- 타입 변수명[=초깃값];
- `int score;`
- `int score = 89;`



variable

```
int pay = 6500;  
int bonus = 300;  
int salary;
```

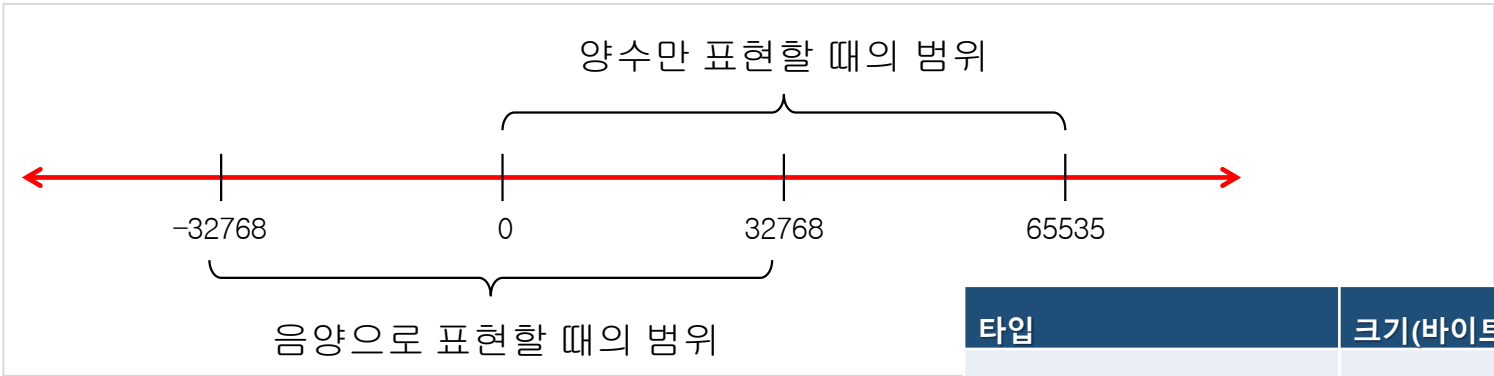
```
salary = pay + bonus;  
printf("급여 = %d원\n", salary);
```



■ 비트수에 따라 표현 가능한 수가 늘어난다.

0	00	000	100	0000	0100	1000	1100
1	01	001	101	0001	0101	1001	1101
	10	010	110	0010	0110	1010	1110
	11	011	111	0011	0111	1011	1111
1비트	2비트	3비트		4비트			

■ 부호 여부에 따라 표현 범위가 달라진다.



■ 정수의 종류

타입	크기(바이트)	부호	범위
char	1	있음	-128 ~ 127
short	2	있음	-32768 ~ 32767
int	4	있음	-2147483648 ~ 2147483647
long	4	있음	-2147483648 ~ 2147483647
unsigned short	2	없음	0 ~ 65535
unsigned	4	없음	0 ~ 4294967295
__int64(long long)	8	있음	-922경 ~ 922경

- 메모리가 무한하지 않아 표현 범위의 한계가 있다.

overflow

```
short a = 20000;  
short b = 30000;  
short c = a + b;  
printf("c = %d\n", c);
```

20000

+

30000

=

-15536

0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

→ 이 자리가 부호로 해석되어 음수가 된다.

- 오버 플로우 : 표현 가능한 최댓값을 초과하는 현상
- int는 가변 크기이고 long은 고정 크기이나 현재는 모두 32비트이다.

	int	long
16비트	16	32
32비트	32	32
64비트	32	32

■ 아라비아 숫자와 부호로 표현한다.

- 1234
- 8906299
- -1440

■ 부호 없는 타입과 long 타입은 접미를 붙인다.

- L : long형 리터럴이다. 1234L은 4바이트의 long형 리터럴이 된다.
- U : 부호없는(unsigned) 리터럴이다. 12345U는 unsigned 타입의 리터럴이다.

■ 진법을 표현할 때는 접두를 붙인다.

- 8진수 : 0으로 시작하면 8진수이다. 015, 032 식으로 쓰면 8진수이다.
- 16진수 : 0x로 시작하면 16진수이다. 0x1f34, 0xabcd 식으로 표기한다. 10 이상의 수를 표현할 때는 알파벳 문자 A ~ F를 사용하며 대소문자는 구분하지 않는다.

- 더 큰 값은 double 형을 사용한다.

float	부호	지수부 : 8	가수부 : 23
double	부호	지수부 : 11	가수부 : 52

■ 아라비아 숫자와 부호 수소점으로 표기한다.

- -2.414
- 0.25
- 89.3400
- 8.0

■ 부동 소수점은 "가수e지수" 형식으로 표기한다.

- 2.414e2 // 241.4와 같다.
- 2.3e-1 // 0.23과 같다.

■ 언더 플로우 : 표현 범위를 넘어서 미세한 값이 잘려 나가는 현상

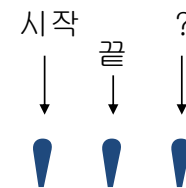
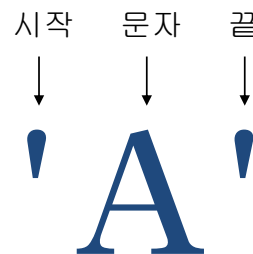
■ 실수를 쓸 일이 많지 않다.

- 문자도 숫자에 대응시켜 저장한다. 대응 방식을 문자셋이라고 하는데 아스키가 기본이다.
- 문자형 타입은 부호에 따라 두 가지 종류가 있다.

타입	크기(바이트)	부호	범위
signed char	1	있음	-128 ~ 127
unsigned char	1	없음	0 ~ 255

- 문자 리터럴은 작은따옴표로 감싸 표기한다.
- 문자형은 코드값으로 해석되는 작은 정수형이다.

- 따옴표안에 담기 어려운 문자가 있다.



- 키보드로 직접 입력이 어려운 문자는 확장열로 표기한다.

확장열	코드	설명
\n	0x0a	개행
\r	0x0d	다음줄
\'	0x27	홀따옴표
\"	0x22	겹따옴표
\\	0x5c	백슬레쉬
\a	0x07	벨 소리
\b	0x08	백 스페이스
\t	0x09	탭
\x##	0x##	16진 코드
\####	0####	8진 코드
\?	0x3f	물음표

■ 일련의 문자가 연속되는 문자의 집합. 큰따옴표로 감싼다.

- "Programming"
- "대한민국"

■ 컴파일러는 문자열 마지막에 끝 표시인 널 문자를 붙인다.

- `char str[6] = "korea";`

k	o	r	e	a	\0
---	---	---	---	---	----

■ 문자열 리터럴 안에서 확장열을 모두 사용할 수 있다.

- "Let's go" // 가능
- "say "Hello" to you" // 에러
- "say \"Hello\" to you" // 가능

- 참 또는 거짓 둘 중의 하나의 값을 기억한다.
- C는 별도의 진위형이 없어 정수를 사용한다.
- 0은 거짓이며 그 외의 값은 모두 참이다.
- 참값에 대한 일관성이 없어 사용자 정의형으로 진위형을 만든다.
 - `typedef int BOOL;`
 - `#define TRUE 1`
 - `#define FALSE 0`
- 정수에 비해 가독성에 차이가 있다.
- C++은 별도의 `bool` 타입을 제공한다.

```
BOOL bMan = TRUE;  
if (bMan == TRUE) {  
    // 남자일 때의 처리  
}
```

```
int bMan = 1;  
if (bMan == 1) {  
    // 남자일 때의 처리  
}
```

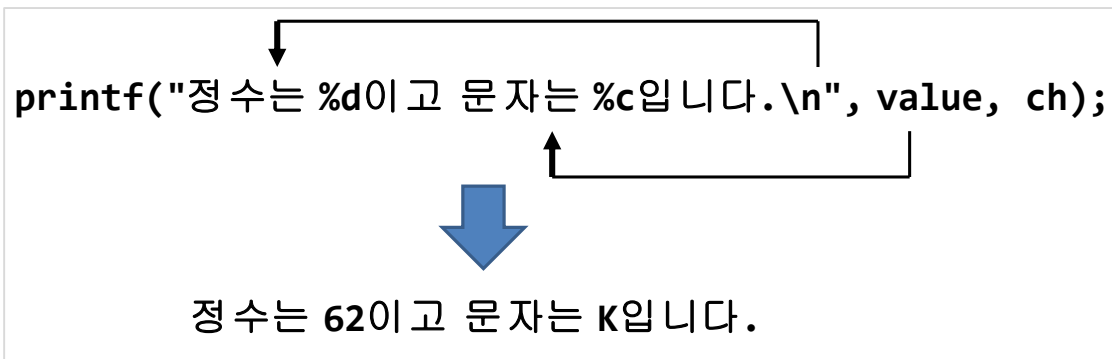

03

기본 입출력 함수

■ C 언어의 가장 기본적인 출력 함수

- `printf("서식문자열" [, 인수, 인수, ...]);`

■ 서식의 개수만큼 뒤쪽에 출력할 값이 있어야 한다.



■ %와 서식 사이에 폭, 정렬, 정밀도를 지정하는 플래그가 들어간다.

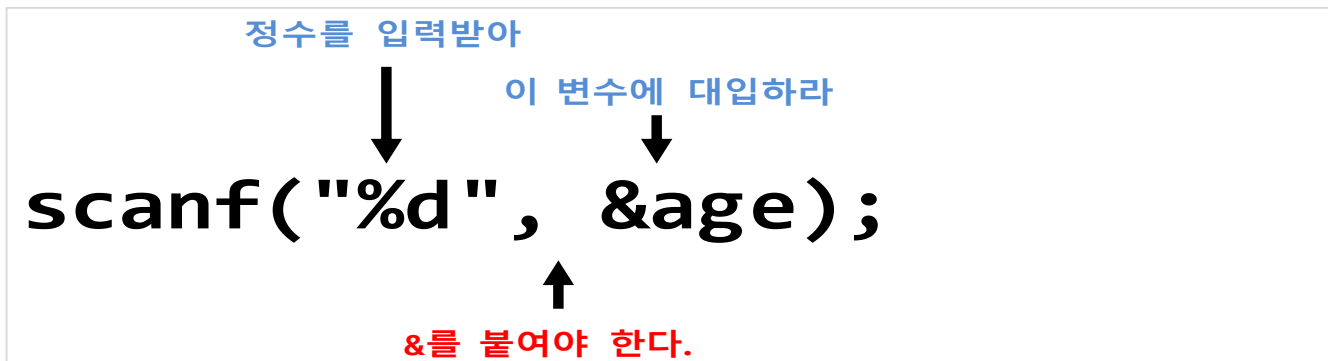
서식	의미	설명
%d 또는 %i	Decimal, Integer	10진 정수로 출력한다.
%o	Octal	8진 정수로 출력한다.
%x 또는 %X	heXadecimal	16진 정수로 출력한다. 대문자 X를 쓰면 A~F의 숫자도 대문자로 출력된다.
%u	Unsigned	부호없는 10진 정수로 출력한다.
%ld	long Decimal	long 형 정수
%lld, %I64d	long long	64비트의 정수
%c	Character	1개의 문자를 출력한다.
%s	String	문자열을 출력한다.
%f	Float	고정 소수점 형식의 실수로 출력한다.
%e 또는 %E		부동 소수점 형식의 실수로 출력한다.
%g 또는 %G		%e, %f중 더 짧은 형식으로 출력한다.
%p	Pointer	포인터의 번지값을 출력한다.
%n		출력된 문자 개수를 포인터 변수에 대입한다.
%%		%문자 자체를 출력한다.

3.2 기본 입출력 함수 – scanf()

- 사용자로부터 입력을 받는 함수

- scanf("서식문자열", &변수);

- 입력을 받아 변수에 대입해 준다. 반드시 번지를 넘겨야 한다.



- 여러 개의 값을 입력받을 수도 있다.

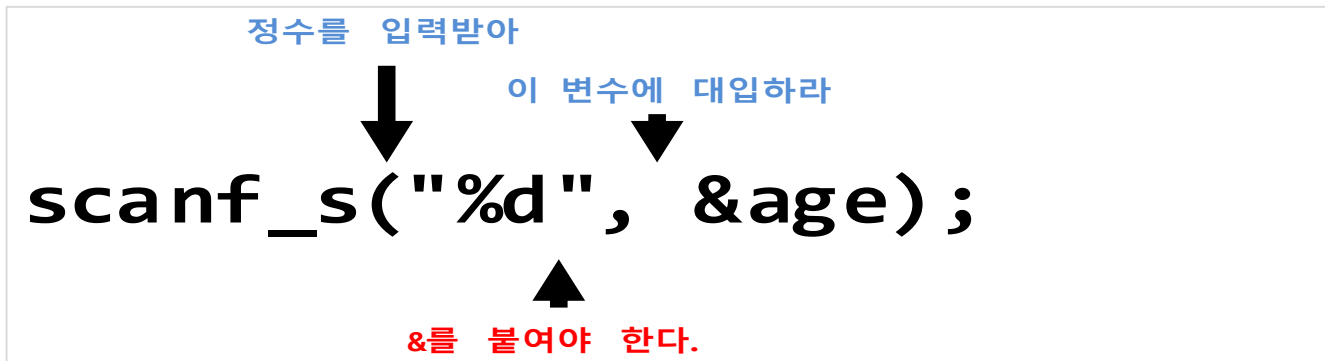
- 그래픽 환경에서는 컨트롤을 사용하므로 실습 이상의 의미는 없다.

3.2 기본 입출력 함수 – scanf_s()

■ 사용자로부터 입력을 받는 함수

- scanf_s("서식문자열", &변수);

■ 입력을 받아 변수에 대입해 준다. 반드시 번지를 넘겨야 한다.



■ 여러 개의 값을 입력 받을 수도 있다.

■ 그래픽 환경에서는 컨트롤을 사용하므로 실습 이상의 의미는 없다.

scanf_s()

정수를 입력하세요: 100
입력한 숫자는 100입니다.

```
#include <stdio.h>

int main()
{
    int num;
    printf("정수를 입력하세요: ");
    scanf_s("%d",&num);
    printf("입력한 숫자는 %d입니다.\n",num);
    return 0;
}
```

scanf_s()

처음 정수를 입력하세요: 100
다음 정수를 입력하세요: 45
100 - 45 = 55

```
#include <stdio.h>

int main()
{
    int num1, num2;

    printf("처음 정수를 입력하세요: ");
    scanf_s("%d",&num1);
    printf("다음 정수를 입력하세요: ");
    scanf_s("%d",&num2);

    printf("%d - %d = %d\n",num1, num2, num1-num2);
    return 0;
}
```

scanf_s()

정수를 입력하세요: 2
다음 정수를 입력하세요: 3
2 x 3 = 6

```
#include <stdio.h>

int main()
{
    int num1, num2;

    printf("정수를 입력하세요: ");
    scanf_s("%d",&num1);
    printf("다음 정수를 입력하세요: ");
    scanf_s("%d",&num2);

    printf("%d x %d = %d\n",num1, num2, num1*num2);
    return 0;
}
```

scanf_s()

정수를 입력하세요: 2
다음 정수를 입력하세요: 3
2 / 3 = 0

```
#include <stdio.h>

int main()
{
    int num1, num2;

    printf("정수를 입력하세요: ");
    scanf_s("%d",&num1);
    printf("다음 정수를 입력하세요: ");
    scanf_s("%d",&num2);

    printf("%d / %d = %d\n",num1, num2, num1/num2);
    return 0;
}
```



```
#include <stdio.h>

int main()
{
    char name[255]="";
    printf("당신의 이름은?: ");
    /*sizeof 연산자는 변수나 값, 자료형의 크기를 알려줌 */
    scanf_s("%s",name, sizeof(name));
    printf("당신의 이름은 %s 입니다.",name);
    return 0;
}
```

```
#include <stdio.h>

int main()
{
    char name[255]="";
    printf("당신의 이름은?: ");
    scanf("%s",name);
    printf("당신의 이름은 %s 입니다.",name);
    return 0;
}
```

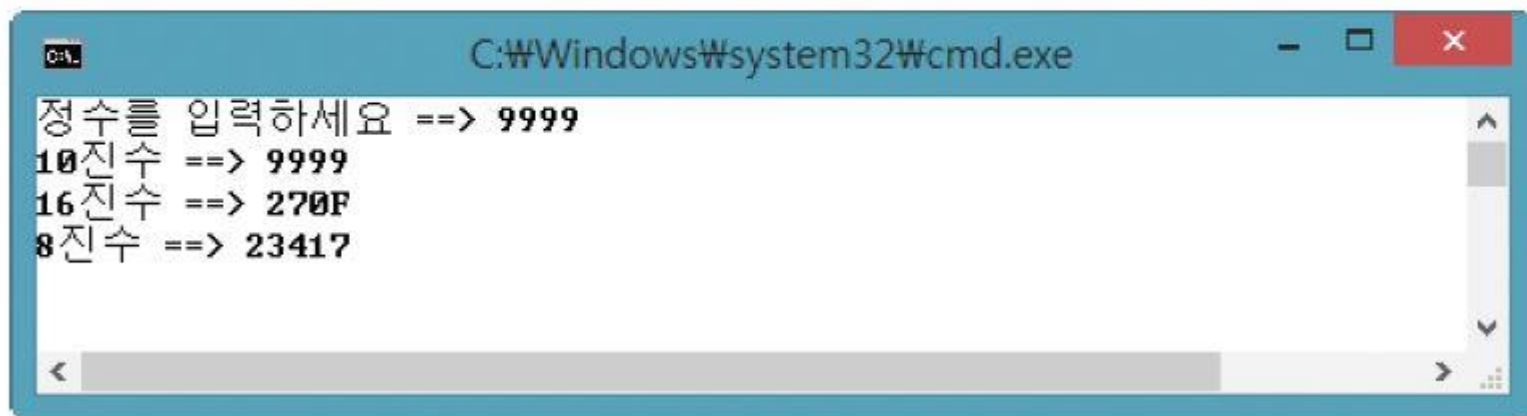
scanf_s()

당신의 이름은?: 홍길동
당신의 이름은 홍길동 입니다.

- `puts("문자열")` : 문자열 출력 후 개행한다.
- `putch(c)` : 문자 하나만 출력한다.
- `gets(변수)` : 문자열을 입력받는다. 공백까지 받을 수 있다.
- `getch()` : 문자 하나를 입력받는다. 입력할 때까지 대기한다.
- `kbhit()` : 키가 눌러졌는지 조사한다.
- `exit(0)` : 프로그램을 종료한다.

예제 설명 정수를 하나 입력받아 10진수, 16진수, 8진수로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

정수를 입력하세요 ==> 9999
10진수 ==> 9999
16진수 ==> 270F
8진수 ==> 23417
```

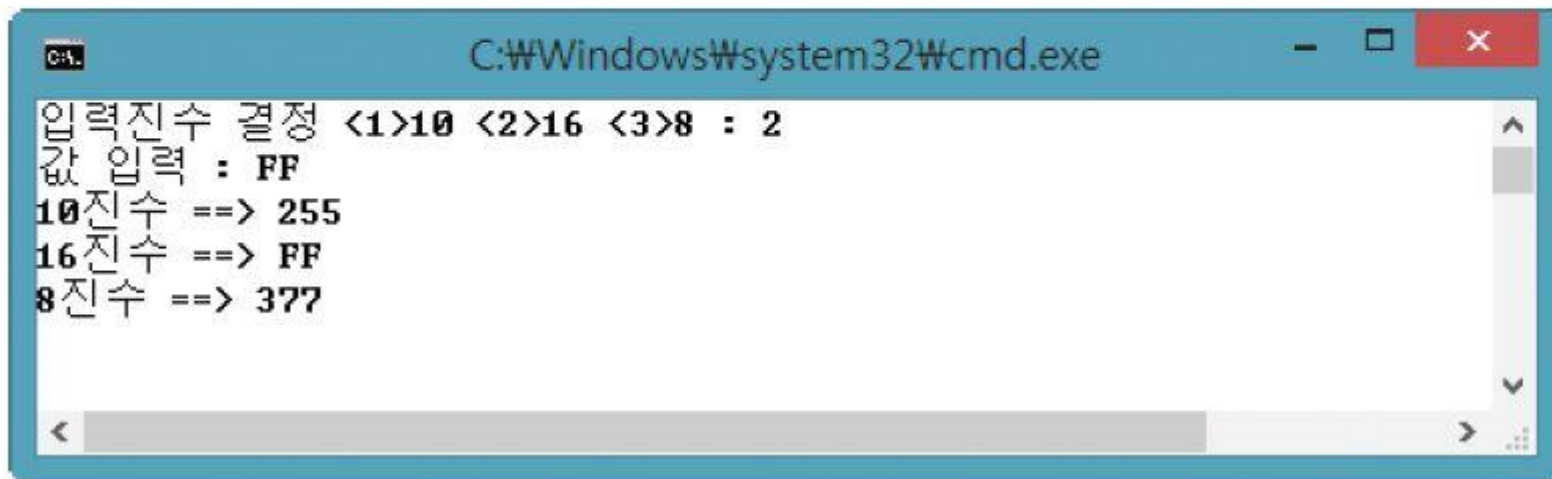
[예제모음 04] 정수형을 출력하는 프로그램

```
01  #include <stdio.h>
02
03  int main( )
04  {
05      int data;          ---정수형 변수를 선언한다.
06
07      printf("정수를 입력하세요 ==> ");
08      scanf_s("%d", &data); ---키보드로 정수를 입력받는다.
09
10      printf("10진수 ==> %d \n", data); ---10진수(%d), 16진수(%X), 8진수(%o)를 출력한다.
11      printf("16진수 ==> %X \n", data);
12      printf("8진수 ==> %o \n", data);
13  }
```

[예제모음 05] 입력하는 정수의 진수 결정

예제 설명 10진수, 16진수, 8진수 중 어떤 진수의 값을 입력받을지 결정하고, 입력받은 수를 10진수, 16진수, 8진수로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
입력진수 결정 <1>10 <2>16 <3>8 : 2
값 입력 : FF
10진수 ==> 255
16진수 ==> FF
8진수 ==> 377
```

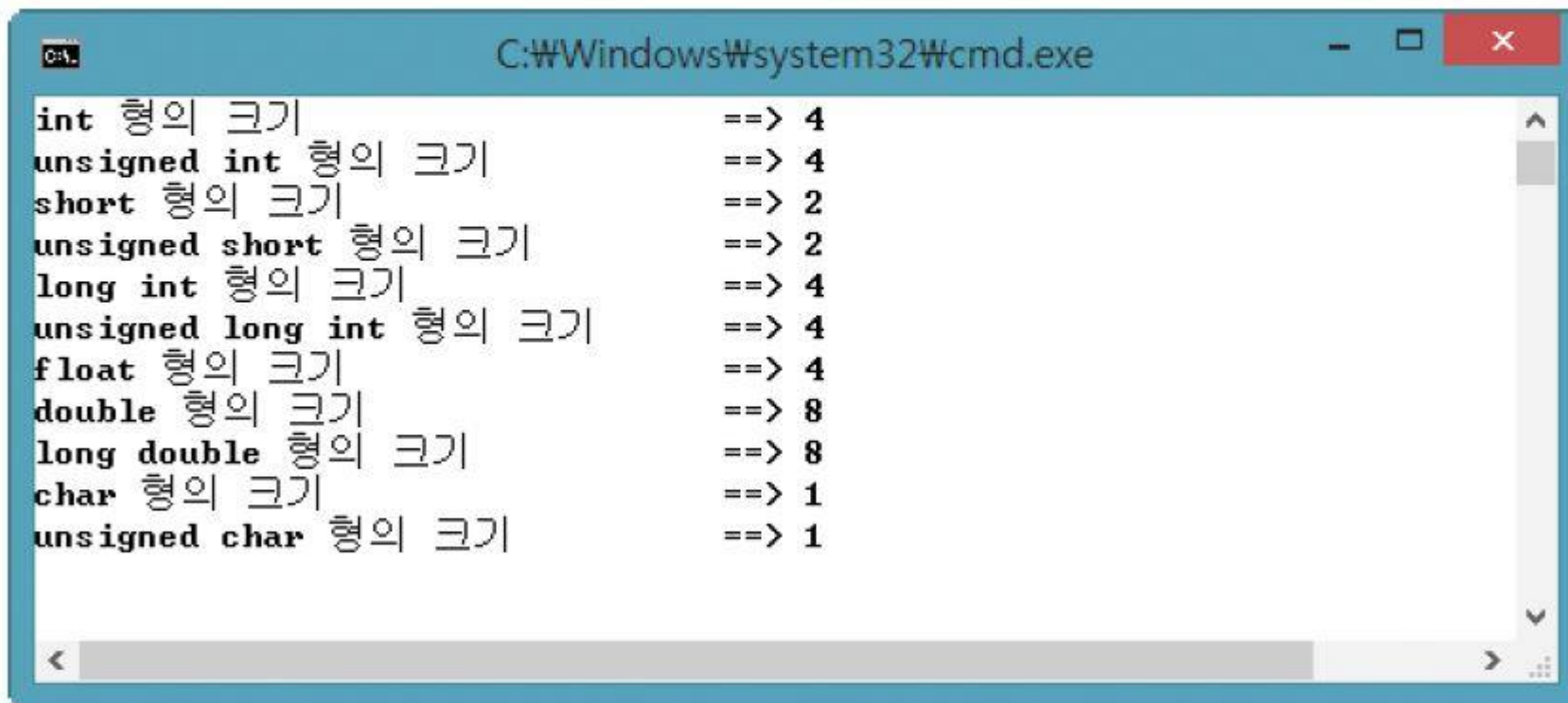
[예제모음 05] 입력하는 정수의 진수 결정

```
01  #include <stdio.h>
02
03  int main( )
04  {
05      int type, data;
06
07      printf("입력진수 결정 <1>10 <2>16 <3>8 : ");
08      scanf("%d", &type); ----키보드로 1~3 중 하나를 입력받는다.
09
10      printf("값 입력 : ");
11
12      if(type == 1) ----입력값이 1이면 10진수를 입력받는다.
13      { scanf("%d", &data); }
14
15      if(type == 2) ----입력값이 2이면 16진수를 입력받는다.
16      { scanf("%x", &data); }
17
18      if(type == 3) ----입력값이 3이면 8진수를 입력받는다.
19      { scanf("%o", &data); }
20
21      printf("10진수 ==> %d \n", data); ----입력받은 data 값을 10진수, 16진수, 8진수로 변환하여 출력한다.
22      printf("16진수 ==> %X \n", data);
23      printf("8진수 ==> %o \n", data);
24  }
```

[예제모음 06] 데이터형의 크기 확인

예제 설명 sizeof() 함수를 사용해서 각 데이터형의 크기를 확인하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

int 형의 크기 ==> 4
unsigned int 형의 크기 ==> 4
short 형의 크기 ==> 2
unsigned short 형의 크기 ==> 2
long int 형의 크기 ==> 4
unsigned long int 형의 크기 ==> 4
float 형의 크기 ==> 4
double 형의 크기 ==> 8
long double 형의 크기 ==> 8
char 형의 크기 ==> 1
unsigned char 형의 크기 ==> 1
```

[예제모음 06] 데이터형의 크기 확인

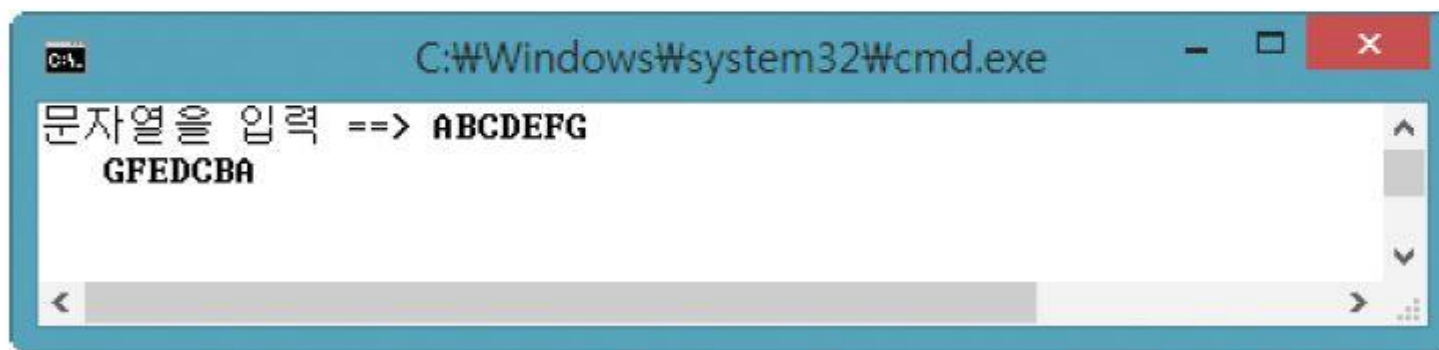
```
01 #include <stdio.h>
02
03 int main( )
04 {
05     printf("int 형의 크기\t\t\t ==> %d\n", sizeof(int));
06     printf("unsigned int 형의 크기\t\t ==> %d\n", sizeof(unsigned int));
07     printf("short 형의 크기\t\t\t ==> %d\n", sizeof(short));
08     printf("unsigned short 형의 크기\t ==> %d\n", sizeof(unsigned short));
09     printf("long int 형의 크기\t\t\t ==> %d\n", sizeof(long int));
10     printf("unsigned long int 형의 크기\t ==> %d\n", sizeof(unsigned long int));
11     printf("float 형의 크기\t\t\t ==> %d\n", sizeof(float));
12     printf("double 형의 크기\t\t\t ==> %d\n", sizeof(double));
13     printf("long double 형의 크기\t\t ==> %d\n", sizeof(long double));
14     printf("char 형의 크기\t\t\t ==> %d\n", sizeof(char));
15     printf("unsigned char 형의 크기\t\t ==> %d\n", sizeof(unsigned char));
16 }
```

--sizeof() 함수로 각 데이터형의 크기(바이트 수)를 출력한다. 이때 컴파일러에 따라서 long double 형은 16바이트 크기일 수도 있다.

[예제모음 07] 입력된 문자열을 반대 순서로 출력

예제 설명 열 글자 미만의 문자열을 입력받고, 입력받은 문자열을 반대 순서로 출력하는 프로그램이다(아직 배우지 않은 내용이 나오지만 나중에 위해 미리 살펴보자).

실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력 ==> ABCDEFG
GFEDCBA
```

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt is "문자열을 입력 ==>". The user has entered "ABCDEFGH", and the program has output "GFEDCBA". The window has a blue title bar and standard Windows window controls (minimize, maximize, close).

[예제모음 07] 입력된 문자열을 반대 순서로 출력

```
01 #include <stdio.h>
```

```
02
```

```
03 int main( )
```

```
04 {
```

```
05     char str[10]= "";
```

```
06     int i;
```

```
07
```

```
08     printf("문자열을 입력 ==> ");
```

```
09     scanf_s("%s", str, sizeof(str));
```

```
10
```

```
11     for(i=sizeof(str)-1; i>=0; i--)
```

```
12     {
```

```
13         printf ("%c", str[i]);
```

```
14     }
```

```
15     printf("\n");
```

```
16 }
```

---문자열을 입력받을 str 배열을 준비한다.

---첨자를 준비한다.

---문자열을 입력받는다.

---str 배열에 들어 있는 문자열을 맨 뒤의 str[9]부터 str[0]까지 출력한다. 즉 입력한 순서의 반대로 출력되는 것이다.

```
#include <stdio.h>

void main() {
    // 변수 선언
    int a=20, b=10, c;

    // 변수 a와 b의 합을 변수 c에 기억 후 결과 출력
    c=a+b;
    printf("%d+%d=%d\n", a,b,c);

    // 변수 a와 b의 차를 변수 c에 기억 후 결과 출력
    c=a-b;
    printf("%d-%d=%d\n", a,b,c);

    //printf() 함수를 이용하여 변수 a와 b의 곱셈 결과 출력
    printf("%d*%d=%d\n", a,b,a*b);

    //printf() 함수를 이용하여 변수 a와 b의 나눗셈 결과 출력
    printf("%d/%d=%d\n", a,b,a/b);
}
```

결과

```
20+10=30
20-10=10
20*10=200
20/10=2
```

■ Program 3-2

```
#include <stdio.h>

int main() {
    int a=12345;
    float pi = 3.141592;
    //자릿수를 지정하여 데이터 출력

    printf("%10d\n", a);
    printf("%3d\n",a);
    printf("%10.2f\n", pi);
    printf("%3.2f\n", pi);

    return 0;
}
```

결과

```
        12345
12345
        3.14
3.14
```

■ Program 2-3

```
#include <stdio.h>

int main() {
    // 문자열 변수 선언 후 초기화
    char a[10] = "ABCDE";

    // 자릿수를 지정한 숫자 출력
    printf("숫자의 오른쪽 정렬:%10d\n", 12345);

    // 음수(-)를 사용한 숫자 출력
    printf("숫자의 왼쪽 정렬:%-10d\n", 12345);

    // 자릿수를 지정한 문자 출력
    printf("문자의 오른쪽 정렬:%10s\n", a);

    // 음수(-)를 사용한 문자 출력
    printf("문자의 왼쪽 정렬:%-10s\n", a);

    return 0;
}
```

결과

```
숫자의 오른쪽 정렬:      12345
숫자의 왼쪽 정렬:12345
문자의 오른쪽 정렬:      ABCDE
문자의 왼쪽 정렬:ABCDE
```

■ Program 2-4

```
#include <stdio.h>

void main() {
    // 변수의 선언
    char a = 'a';
    char b = 65;

    // 변수의 출력
    printf("char a = %c\n", a);
    printf("ASCII Code 65 = %c\n", b);

    getchar();
}
```

결과

```
char a = a
ASCII Code 65 = A
```

■ Program 2-5

```
#include <stdio.h>
void main() {
    //문자 배열을 선언한 후 값을 초기화 한다.
    char name[4] = "Lee"; //3글자를 기억시킬 것이므로 크기를 '4'로 지정한다.

    //문자 배열에 기억된 값을 출력한다.
    printf("name = %s\n", name);

    getchar();
}
```

결과

```
name = Lee
```

■ Program 2-6

```
#include <stdio.h>

//typedef을 이용하여 unsigned int 형을 uint이라는 이름으로 재정의
typedef unsigned int uint;
//typedef을 이용하여 int 형을 integer이라는 이름으로 재정의
typedef int integer;

int main() {

    //uint형으로 number1 선언
    uint number1;

    //uint형으로 number2 선언
    uint number2;

    number1 = 100;
    number2 = 200;

    printf("number1 : %d, number2 : %d\n", number1, number2);
    return 0;
}
```

결과

```
number1 : 100, number2 : 200
```


■ Program 2-7

```
#include <stdio.h>

void main() {

    const float PI = 3.14;
    int radius;

    scanf_s("%d",&radius);
    printf("%d * 3.14 = %f\n", radius, radius*PI);
    getchar();
}
```

결과

```
3
3 * 3.14 = 9.420000
```

■ Program 2-8

```
//CHARGE 상수 선언
#define CHARGE 2000

#include <stdio.h>

void main() {
    // 사용 시간과 사용 요금을 정수형 변수로 선언한다.
    int time, rate;
    printf("사용시간을 입력:");

    //사용 시간을 입력 받는다.
    scanf_s("%d",&time);

    //사용요금은 사용시간x상수 CHARGE로 계산한다.
    rate = time * CHARGE;

    //사용요금을 출력한다.
    printf("사용요금은 %d원 입니다.\n", rate);

    getchar();
}
```

결과

사용시간을 입력:2
사용요금은 4000원 입니다.

■ Program 2-9

```
//CHARGE 상수 선언
#define CHARGE 3000

#include <stdio.h>

void main() {
    // 사용 시간과 사용 요금을 정수형 변수로 선언한다.
    int time, rate;
    printf("사용시간을 입력:");

    //사용 시간을 입력 받는다.
    scanf("%d",&time);

    //사용요금은 사용시간x상수 CHARGE로 계산한다.
    rate = time * CHARGE;

    //사용요금을 출력한다.
    printf("사용요금은 %d원 입니다.\n", rate);

    getchar();
}
```

결과

사용시간을 입력:2
사용요금은 6000원 입니다.

■ Program 2-10

```
#include <stdio.h>

void main() {
    // 사용 시간과 사용 요금을 정수형 변수로 선언한다.
    int time, rate;

    //시간당 사용 요금은 상수로서 2000이 된다.
    const int charge = 2000;

    //사용 시간을 입력받는다.
    printf("사용시간을 입력:");
    scanf("%d",&time);

    //사용요금은 사용시간x상수 CHARGE로 계산한다.
    rate = time * charge;

    //사용요금을 출력한다.
    printf("사용요금은 %d원 입니다.\n", rate);

    getchar();
}
```

결과

사용시간을 입력:6
사용요금은 12000원 입니다.

04

연산자

```
int my_age;  
my_age = 20;  
my_age = 24 + 1;
```

■ 주소가 있으면 L-Value, 없으면 R-Value(상수, 연산식)

```
int my_age, your_age, sum;  
your_age = 19;  
my_age = your_age;  
my_age = my_age + 1;  
sum = my_age + your_age;
```

```
int my_age;  
20 = my_age;           // Error  
my_age + 1 = 20;       // Error
```

■ 변수

- 대입문의 좌변에 오면 L-Value(변수 그 자체)
- 대입문의 우변에 오면 R-Value(변수의 값)
- 대입문의 좌변에는 L-Value만 가능

```
int count, sum = 20;
```

■ 변수 선언과 동시에 초기화(Initialization) 가능

■ 초기화에 사용된 등호('=')는 대입 연산자가 아니다.

- 초기화: 변수에 메모리를 할당하면서 동시에 우변 값을 넣음.
- 대입: 프로그램 실행 도중 변수를 찾아가서 우변 값을 넣음.

```
#include <stdio.h>
int main( ){
    int my_age, your_age;
    int her_age = 20;

    my_age = her_age + 1;
    printf("My age is %d.\n", my_age);

    my_age = your_age + 1;
    printf("My age is %d.\n", my_age);

    return 0;
}
```

```
My age is 21.
My age is 1.
```

- 연산자 : 자료를 가공하여 정보를 만드는 수단
- 피연산자의 개수에 따라
단항, 이항, 삼항 연산자로 분류한다.

연산자 분류	예
산술 연산자	+ - * / %
부호 연산자	+ -
대입 연산자	= 복합 대입 연산자
증감 연산자	++ --
포인터 연산자	* & []
구조체 연산자	. ->
관계 연산자	== != <= < >= >
논리 연산자	&& !
비트 연산자	& ~ >> <<
삼항 조건 연산자	? :
쉼표 연산자	,
sizeof 연산자	sizeof
캐스트 연산자	(type)
괄호 연산자	()
C++ 연산자	new delete :: .* ->*

■ 연산자와 피연산자

연산자
 $a + b$
피 연산자 1 피 연산자 2

- 산술 연산: +, -, /, *, %
- DIV(/): 정수연산이면 몫(cf. 부동소수 연산)
- MOD(%): $10 \% 3 = 1$. 자동차 미터기. 시계 바늘

```
a - b
= a + (256 - b)
= a + (255 - b) + 1
```

- 8 비트 연산이면 $(256 + n) \% 256 = n$.
- $(255 - b)$ 는 b 에 대한 1의 보수. 1을 더하면 2의 보수

```
#include <stdio.h>
int main() {
    int a = 10, b = 3;    double p = 10.0, q = 3.0;

    printf("10 + 3 = %d.\n", a + b);
    printf("10 - 3 = %d.\n", a - b);
    printf("10 * 3 = %d.\n", a * b);
    printf("10 / 3 = %d.\n", a / b);
    printf("10 %% 3 = %d.\n\n", a % b);

    printf("10.0 + 3.0 = %f.\n", p + q);
    printf("10.0 - 3.0 = %f.\n", p - q);
    printf("10.0 * 3.0 = %f.\n", p * q);
    printf("10.0 / 3.0 = %f.\n", p / q);
    return 0;
}
```

```
10 + 3 = 13.
10 - 3 = 7.
10 * 3 = 30.
10 / 3 = 3.
10 % 3 = 1.
```

```
10.0 + 3.0 = 13.000000.
10.0 - 3.0 = 7.000000.
10.0 * 3.0 = 30.000000.
10.0 / 3.0 = 3.333333.
```

- 일상 생활에서 자주 사용하는 가감승제 연산을 수행한다.
- 나눗셈은 피연산자의 타입에 따라 결과가 달라진다.
- 정수끼리 나누면 결과도 정수, 실수끼리 나누면 결과도 실수이다.
 - $3.0 / 2.0$ // 결과는 1.5
 - $3 / 2$ // 결과는 1
- 소수점 이하까지 정밀한 값을 구하려면 실수로 나누어야 한다.

■ 코드 해설

- 정수 오버플로우: **순환 값**으로 대치($32767 + 1 = -32768$)
- 부동 소수 오버플로우: INF를 리턴

```
#include <stdio.h>
int main( ) {
    short a = 32768;
    short b = a / 2;
    float c = 1E45;
    float d = c / 2.0;

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

a: -32768, b: -16384, c: 1.#INF00, d: 1.#INF00.

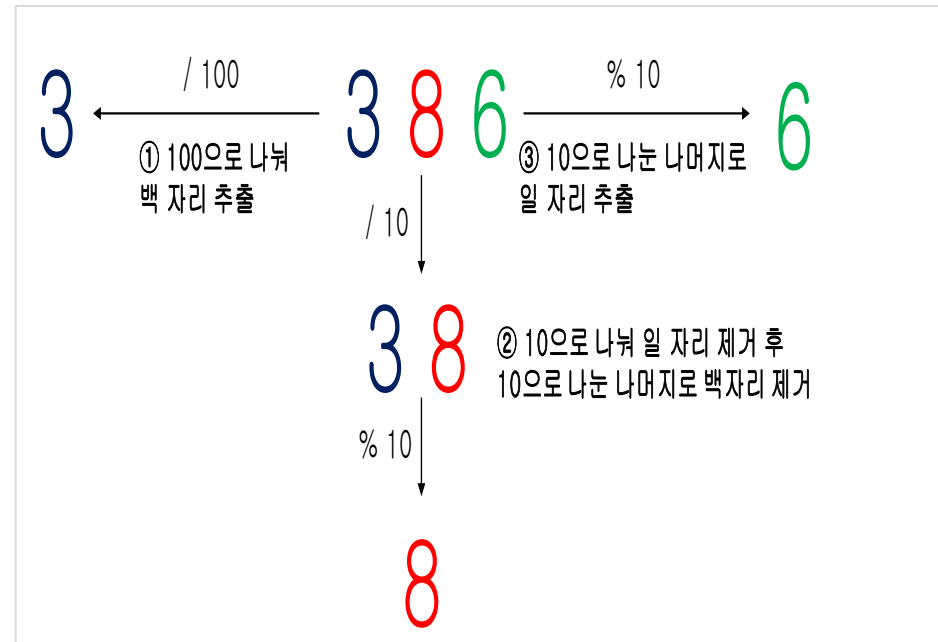
- 정수를 나눈 후 몫은 버리고 나머지만 취한다. 배수 여부를 판별할 때 주로 사용한다. 2로 나눈 나머지가 0이면 짝수이다.
- 각 자리수를 추출하는 예제 - 나누기와 나머지 연산자를 적절히 활용한다.

$$\begin{array}{r} 2 \\ 4 \overline{) 9} \\ \underline{8} \\ 1 \end{array}$$

몫은 버리고
나머지만 리턴한다.

modular

```
int value = 386;  
  
int h = value / 100;  
int d = value / 10 % 10;  
int n = value % 10;  
  
printf("%d백%d십%d\n", h, d, n);
```



- 피연산자의 부호를 지정한다. 뺄셈 연산자와 달리 단항 연산자이다.
- 피연산자 개수에 따라 연산자인지 상수인지 잘 판단해야 한다.
 - $a = b * -c - d;$
 - $a = -3;$
- 덧셈 부호 연산자는 대칭성을 위해 존재할 뿐 실용성은 없다.

```
int a = -1  
int b = 2  
int c = -5  
int d = 6
```

```
int a = -1  
int b = +2  
int c = -5  
int d = +6
```

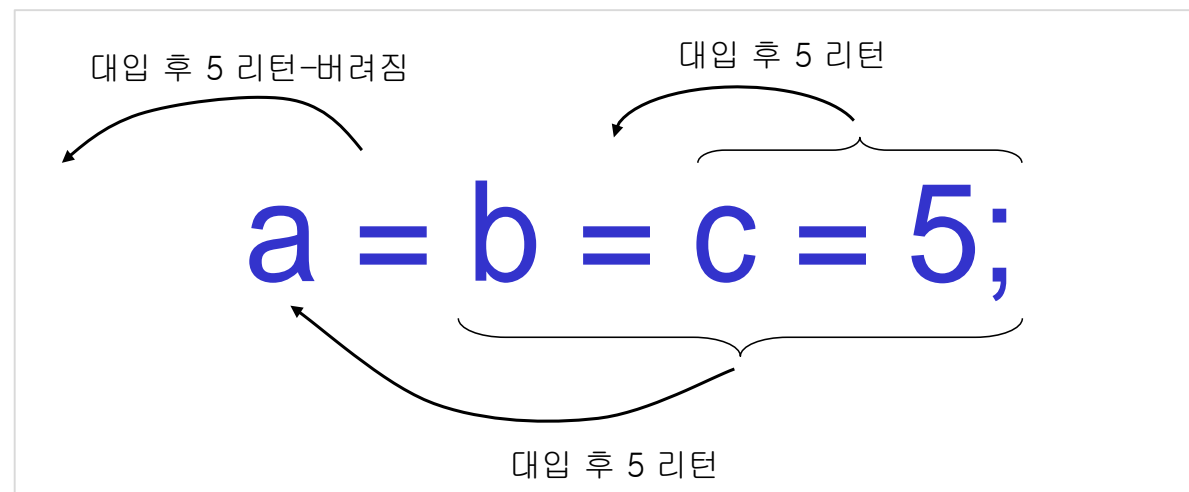
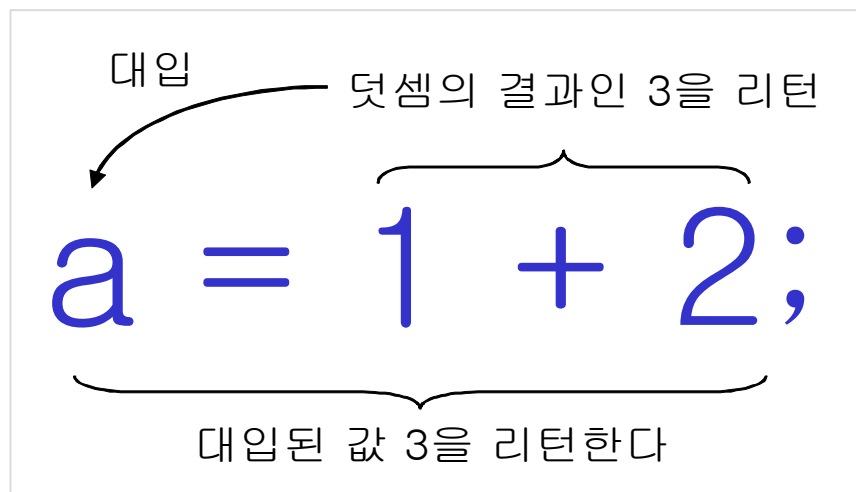
- 변수에 값을 저장하며 = 기호를 사용한다. 우변의 상수 또는 수식 변수식을 좌변에 대입한다.

- $a = 5;$
- $a = 5 * 6;$
- $b = c + d * e;$

- 대입 연산자의 왼쪽에는 메모리를 점유하고 값을 바꿀 수 있는 좌변값만 올 수 있다. 상수나 수식은 좌변값이 아니다.

- $5 = a + 1$
- $b + c = 10;$

- 연산자는 연산 결과를 리턴하며 연쇄적 연산이 가능하다.



- 좌변의 변수가 우변에 올 때는 복합 대입 연산자로 바꿀 수 있다.

- $a = a + 1;$
- $a += 1;$

- 자기 자신에 대해 연산을 수행한다.



- 변수명이 길어질 때는 편의성 차이가 있다.

- $\text{TotalScoreOfAllSubject} = \text{TotalScoreOfAllSubject} + 8;$
- $\text{TotalScoreOfAllSubject} += 8;$

- 10개의 복합 대입 연산자가 제공된다.

- $+= \quad -= \quad *= \quad /= \quad \%= \quad <<= \quad >>= \quad \&= \quad |= \quad ^=$

$a += b;$ means $a = a + b;$

$a -= b;$ means $a = a - b;$

$a *= b;$ means $a = a * b;$

$a /= b;$ means $a = a / b;$

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

    a = 8;
    a += 2;
    printf("a = %d.\n", a);

    b = 2;
    a /= b;
    printf("a = %d.\n", a);
    return 0;
}
```

a = 10.
a = 5.

a	NOT a
0	1
1	0

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

■ Boolean Algebra: 논리 연산 표현을 위한 대수학

논리 연산자	의미
&&	logical AND
	logical OR
!	logical NOT

- C 언어는 세 가지 논리 연산을 제공.
- 논리 연산은 0(false) 아닌 모든 값은 1(true)로 간주.

```
#include <stdio.h>
int main( ){
    int a = 4, b = 3, c = 2, d = 1;

    printf("%d \n", (a > b) && (c > d));
    printf("%d \n", (a < b) || (c > d));
    printf("%d \n", (a < b) && (c > d));
    printf("%d \n", (a > b) || (c > d));
    printf("%d \n", (a < b || c > d) && (a == b && c >= d));

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

1
1
0
1
0
0
0
0
0

관계 연산자	의미
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
==	equal to
!=	not equal

- 서로 같은지 비교하려면 등호를 두 번 써야 한다.
- 관계 연산의 결과는 **1(true)** 또는 **0(false)**다.

```
#include <stdio.h>

int main() {
    int a = 10, b = 5;
    double p = 10.0, q = 5.02, r = 10.0000000000000001;

    printf("%d \n", a > b);
    printf("%d \n", a >= b);
    printf("%d \n", a == b);
    printf("%d \n", a != b);

    printf("%d \n", p > q);
    printf("%d \n", p == r);
    return 0;
}
```

```
1
1
0
1
1
1
```

■ 변수의 값을 1씩 증가, 감소시킨다. 1 증가시키는 세가지 방법

- ① `a = a + 1` // 평이한 대입문
- ② `a += 1` // 복합 대입문
- ③ `a++` // 증가 연산자

■ 증감 연산자는 두 가지 형식으로 사용된다.

- 전위형(Prefix) : 증감 연산자가 피연산자 앞에 위치한다. `++a`, `--a`
- 후위형(Postfix) : 증감 연산자가 피연산자 뒤에 위치한다. `a++`, `a--`

■ 증가전의 값을 리턴하는지 증가 후의 값을 리턴하는지가 다르다.

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

    a = 0;
    a++;
    printf("a = %d.\n", a);
    b = a++;
    printf("a = %d. b = %d.\n", a, b);

    a = 0;
    ++a;
    printf("a = %d.\n", a);
    b = ++a;
    printf("a = %d. b = %d.\n", a, b);

    a = b = 0;
    printf("a = %d. b = %d.\n", (1 + a++) + 2, ++b);
    printf("a = %d. b = %d.\n", a, b);
    return 0;
}
```

```
a = 1.
a = 2. b = 1.
a = 1.
a = 2. b = 2.
a = 3. b = 1.
a = 1. b = 1.
```

증가된 값 증가하기 전의 값

`b = ++a;` `b = a++;`

비트 연산자	의미
&	bitwise AND
 	bitwise OR
^	bitwise XOR
~	bitwise NOT
<<	bitwise LEFT SHIFT
>>	bitwise RIGHT SHIFT

bitwise AND	bitwise OR	bitwise XOR	bitwise NOT
0110 1001 (69 _{hex})	0110 1001	0110 1001	
0101 0101 (55 _{hex})	0101 0101	0101 0101	0110 1001
0100 0001 (41 _{hex})	0111 1101	0011 1100	1001 0110

```
#include <stdio.h>

int main() {
    int a = 105, b = 85;
    printf("%d\n", a & b);
    printf("%X\n", a | b);
    printf("%X\n", a ^ b);
    printf("%X\n", ~a);
    return 0;
}
```

```
65
7D
3C
FFFFFF96
```

```
#include <stdio.h>

int main() {
    int a = 105, b = 32;
    printf("105 modulo 32 is %d.\n", a % b);
    printf("105 bitwise AND 31 is %d.\n", a & (b - 1));
    return 0;
}
```

```
105 modulo 32 is 9.
105 bitwise AND 31 is 9.
```

```
#include <stdio.h>

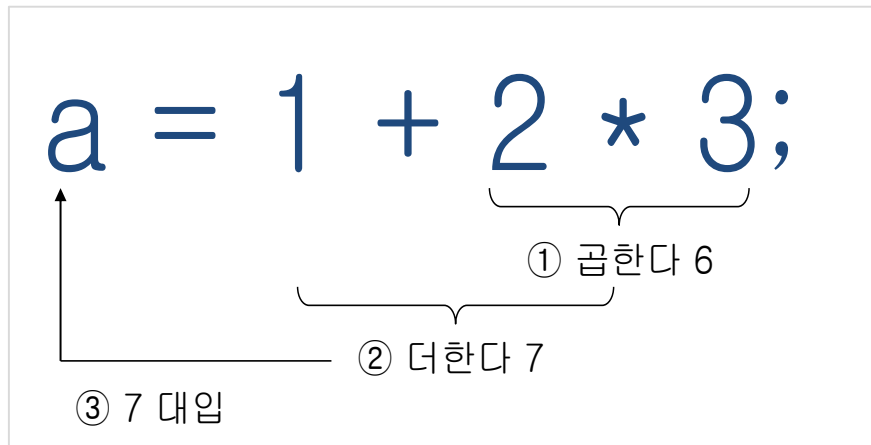
int main() {
    unsigned int a = 25;
    unsigned left, right;

    left = a << 3;
    right = a >> 3;
    printf("left shift: %d, right shift: %d.\n", left, right);
    return 0;
}
```

```
left shift: 200, right shift: 3.
```

4.4 연산자 우선순위

- 한 수식에 여러 개의 연산자가 있을 때 어떤 연산자를 먼저 처리할 것인가의 순서를 우선순위라고 한다.
- C의 모든 연산자는 우선 순위가 정해져 있다.
- 우선순위가 헛갈리면 명시적으로 괄호를 감싸는 것이 좋다.
- 결합 방향은 같은 순위의 연산자가 있을 때 연산할 방향을 지정한다. 대부분 왼쪽 우선이나 단항, 대입, 삼항 연산자만 오른쪽 우선이다.



순위	연산자	결합순서
1	<code>() [] -> .</code>	왼쪽 우선
2	<code>! ~ ++ -- + -(부호) *(포인터) & sizeof 캐스트</code>	오른쪽 우선
3	<code>*(곱셈) / %</code>	왼쪽 우선
4	<code>+ -(덧셈, 뺄셈)</code>	왼쪽 우선
5	<code><< >></code>	왼쪽 우선
6	<code>< <= > >=</code>	왼쪽 우선
7	<code>== !=</code>	왼쪽 우선
8	<code>&</code>	왼쪽 우선
9	<code>^</code>	왼쪽 우선
10	<code> </code>	왼쪽 우선
11	<code>&&</code>	왼쪽 우선
12	<code> </code>	왼쪽 우선
13	<code>? :</code>	오른쪽 우선
14	<code>= 복합대입</code>	오른쪽 우선
15	<code>,</code>	왼쪽 우선

- 연산자의 양쪽은 같은 타입끼리 연산하는 것이 원칙이다. 그러나 타입이 약간 달라도 자동 변환하여 연산 가능하다.
- 더 큰 타입으로 바꿀 때 상승 변환이 발생하며 작은 타입의 변수에 대입할 때 하강 변환이 발생한다.

convert

```
int a = 3;  
double d = 2.41;  
  
int b = a + d;  
printf("b = %d\\n", b);
```

$b = a + d$

상승 변환 : a가 잠시 실수형이 된다.

하강 변환 : b에 대입되기 위해 정수형이 된다.

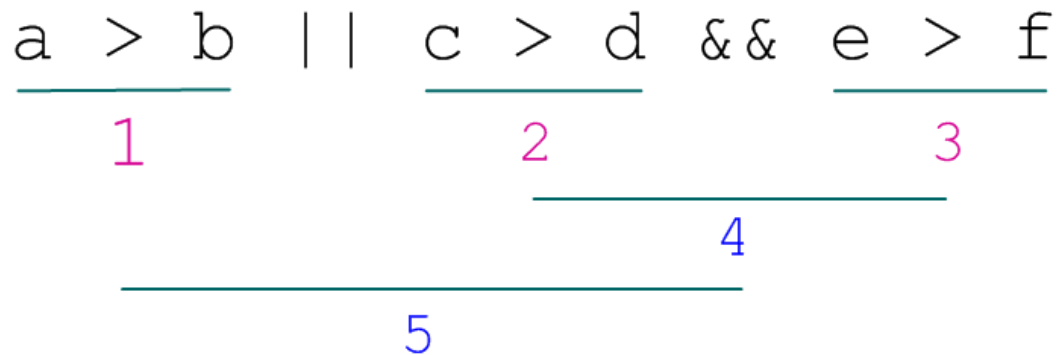
- 산술 변환이 적합치 않을 때는 캐스트 연산자를 사용하여 강제로 타입을 지정한다.
 - (타입)변수
- 실수 평균을 구하려면 피연산자중 하나를 잠시 실수로 바꿔야 한다.

cast

```
int sum = 427;
int num = 5;

printf("평균 = %d\n", sum / num);
printf("평균 = %f\n", sum / (double)num);
```

```
평균 = 85
평균 = 85.400000
```



- 1이 참이면 2,3,4는 평가되지 않는다.

- OR 연산에서 하나라도 참이면 결과는 참

```
a > 1 || b++ > 0
```

- `(a > 1)`이 참이면 `b`가 증가할 것인가?
- 어느 것이 먼저 평가될 것인지는 컴파일러에 따라 다를 수 있다.
- 단축 회로 연산을 예상한 프로그램은 위험하다.