4. **Character Strings과 Formatted Input/Output**

□○•▶★

□ Function: strlen()

□ Keywords: const

□ Character strings

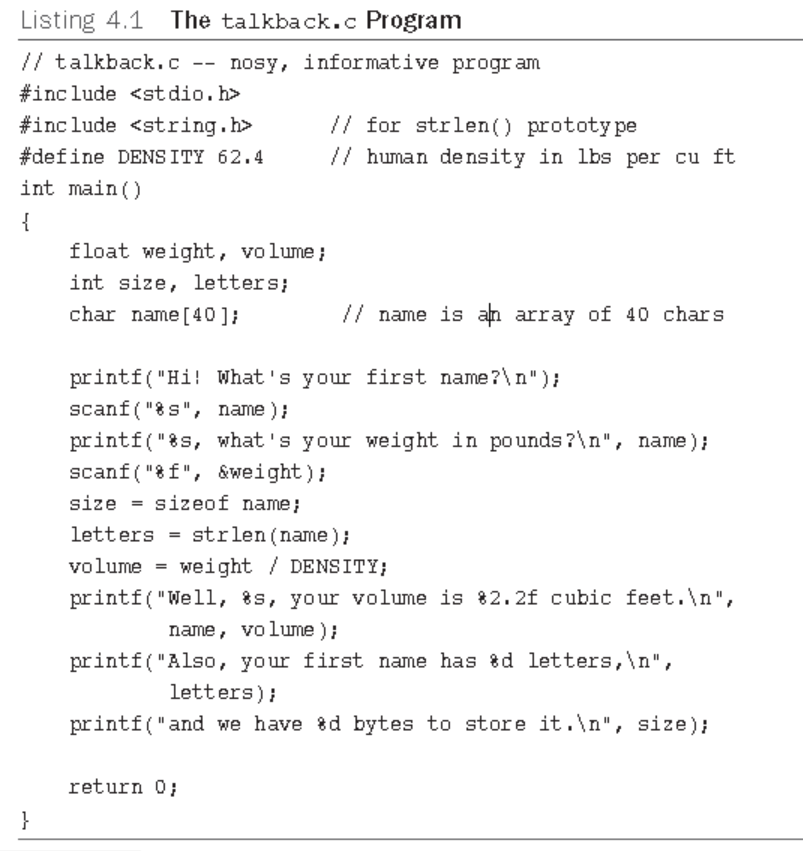
□ character strings을 어떻게 만들고 저장하나?

□ character strings 을 read하고 display하기 위한 scanf()과 printf()의 사용

□ string 길이를 구하기 위한 strlen() 사용 방법

□ symbolic constants 를 사용하기 위한 C preprocessor의 #define directive와 ANSI C의 const modifier

* 1. **소개하는 Program**



* *character string*를 저장하기 위해 *array*를 사용
* string의 입출력 처리는 %s *conversion specification*을 사용

▶ 차이점: 정수, 실수 입력시(scanf )에 사용한 & prefix 를 사용하지 않음

* 62.4 : constant를 나타내기 위해

#define DENSITY 62.4 .

* String name의 길이는 C function strlen()를 사용

Letters = length(name);

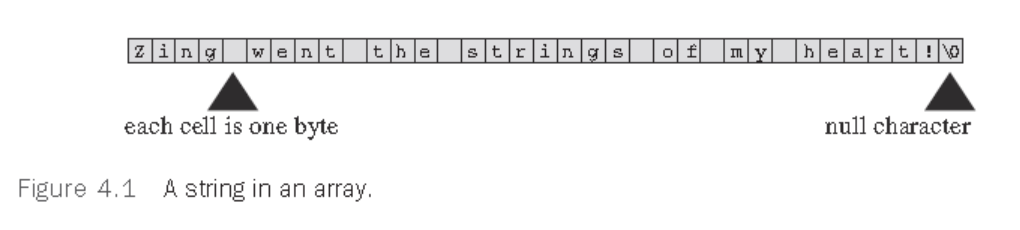
**4.2 Character Strings의 소개**

□ *character string* 은 한 개 이상의 characters

"Zing went the strings of my heart!"

4.2.1 Type char Arrays와 Null Character

□ strings은 type char의 array로 저장됨

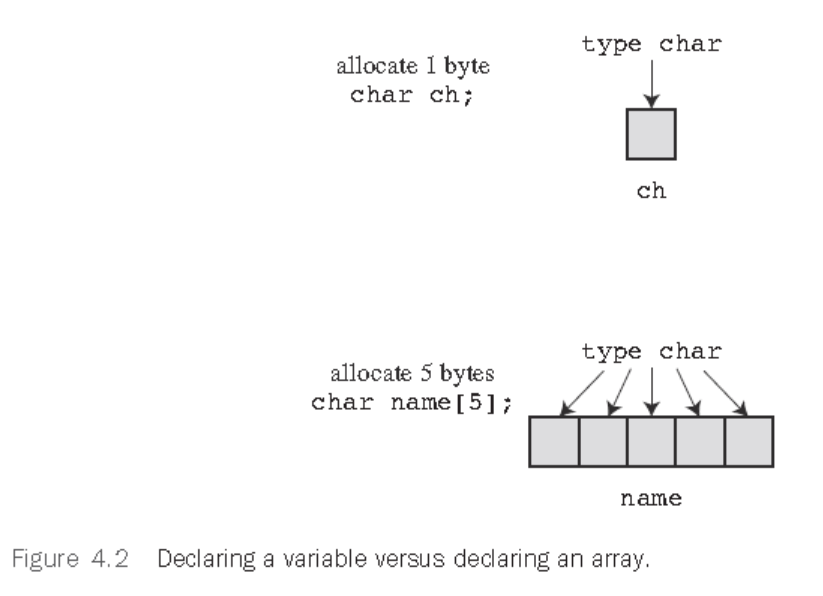


* *null character*

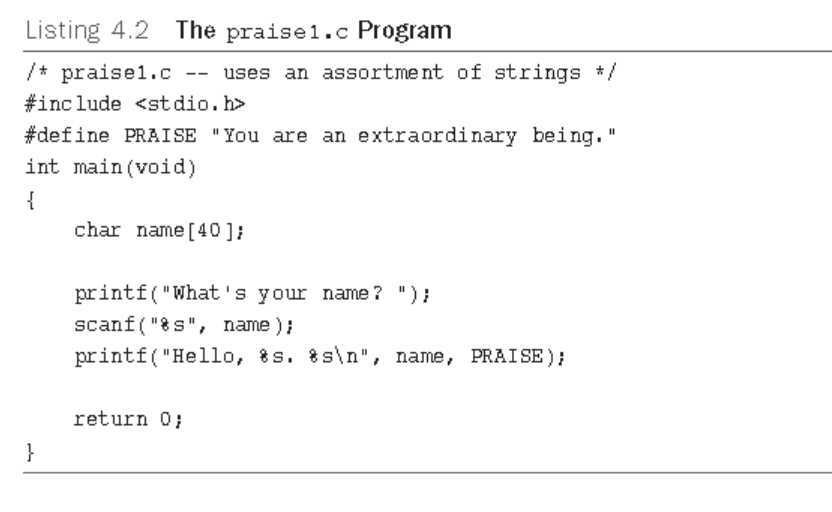
○ \0: C에서 string의 끝을 mark하는데 사용됨

* null character는 숫자 zero가 아님
* C에서 strings은 null character로 끝나는 문자열로 저장됨
  + Array는 ordered sequence of data elements of one type
  + Array를 row에 있는 연속적인 memory cells임
  + 이름 다음의 brackets은 array 임을 나타냄

char name[40];



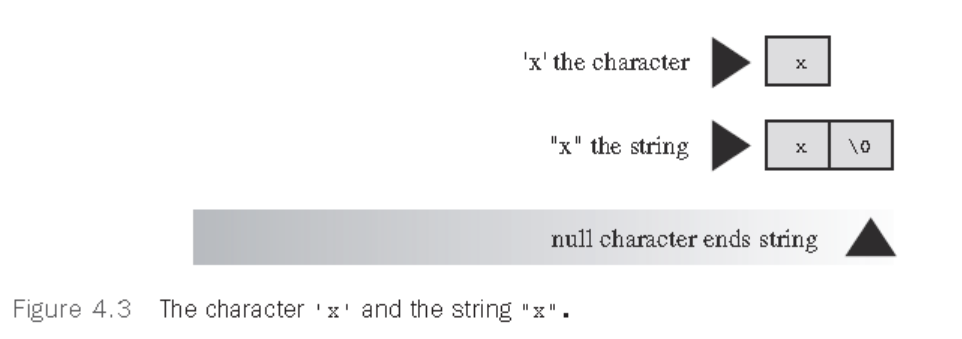
4.2.2 Strings 이용



* %s 는 printf()가 string을 print하는 것을 표시
* scanf()는 null character를 name array의 끝에 추가하여 끝을 표시
* string literal: define PRAISE “You are”
* compiler가 자동으로 null char를 배열의 끝에 추가

□ scanf()가 string을 입력받을 때 first whitespace (blank, tab, or newline)가 나타나면 string이 끝

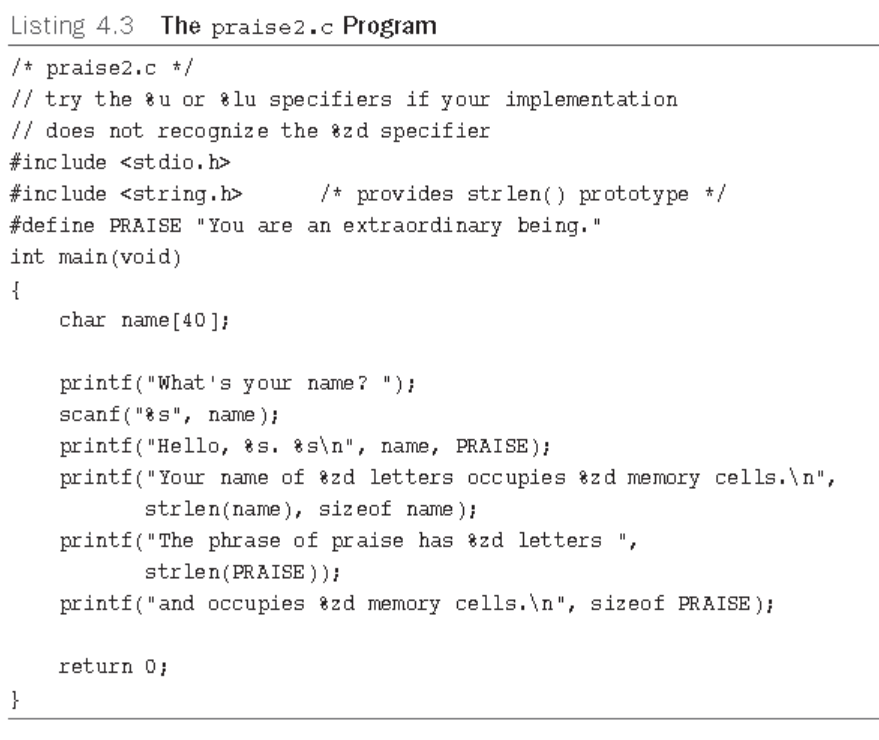
* scanf()는 “a”를 read하기 위해 %s를 사용(Fig 4.3)
  + 1. Strings 대비 Characters



* 'x' 는 basic type ( char )이고, "x"는 derived type으로 array of char
* "x"는 two characters: 'x' 와 '\0'(null character)

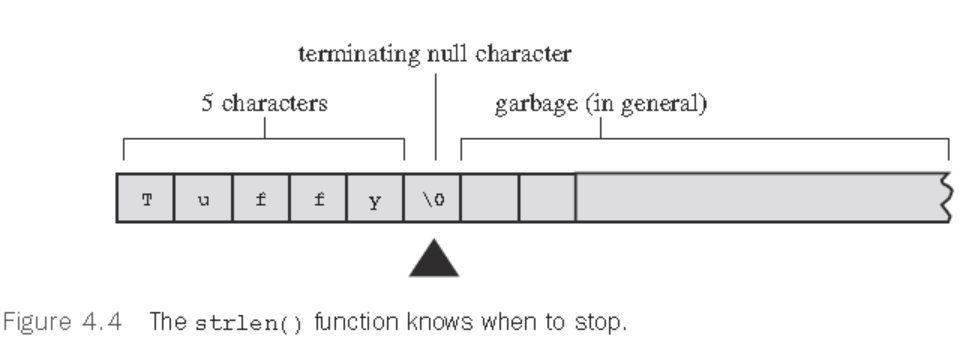
4.2.4 strlen() Function

□ strlen() function는 character 개수로 string length를 계산



#include <string.h>

* string.h file은 strlen()등의 string에 관련된 function prototype을 포함함
  + printf()과 scanf()는 stdio.h header file 에 있는 표준 입출력 function을 포함함



* strlen()는 string의 null 문자를 제외한 문자 개수를 return한다
* sizeof operator는 char name[40];인 경우에는 40을 준다
  + sizeof operator의 값은 %zd specifier를 사용하여 print

**4.3 Constants 와 C Preprocessor**

circumference = 3.14159 \* diameter;

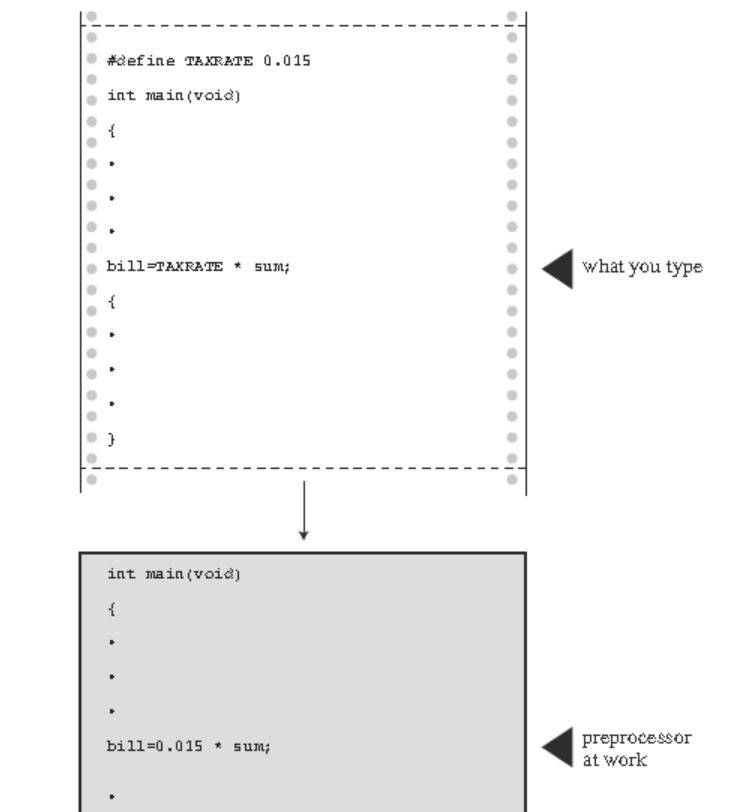
* constant 3.14159는 constant pi (π)의 값임
* *symbolic constant*를 사용하여 표현

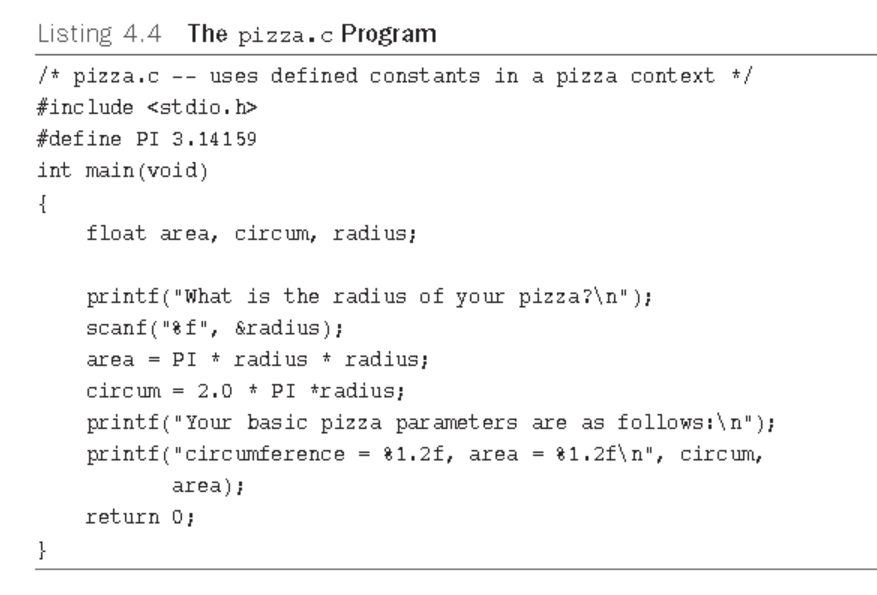
circumference = pi \* diameter;

#define TAXRATE 0.015

owed = TAXRATE \* housevalue;

* *compile-time* 대체





4.3.1 *const* Modifier

const int MONTHS = 12; // MONTHS a symbolic constant for 12

* const int가 #define 보다 더 유연하고 효율적

4.3.2 Manifest Constants on the Job

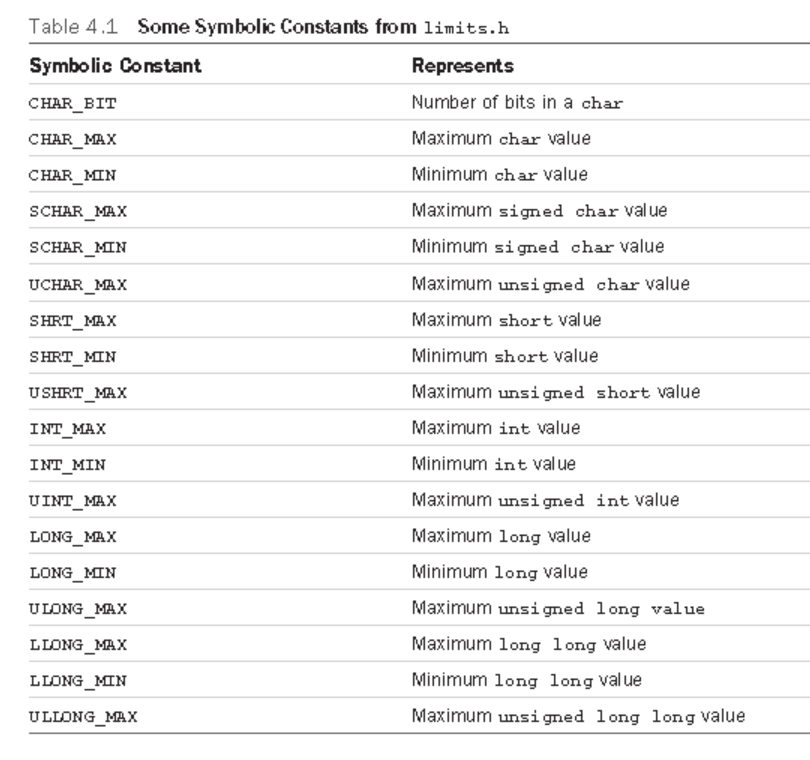
□ C header files인 limits.h과 float.h는 정수, 실수의 최대 값에 대한 상수를 제공

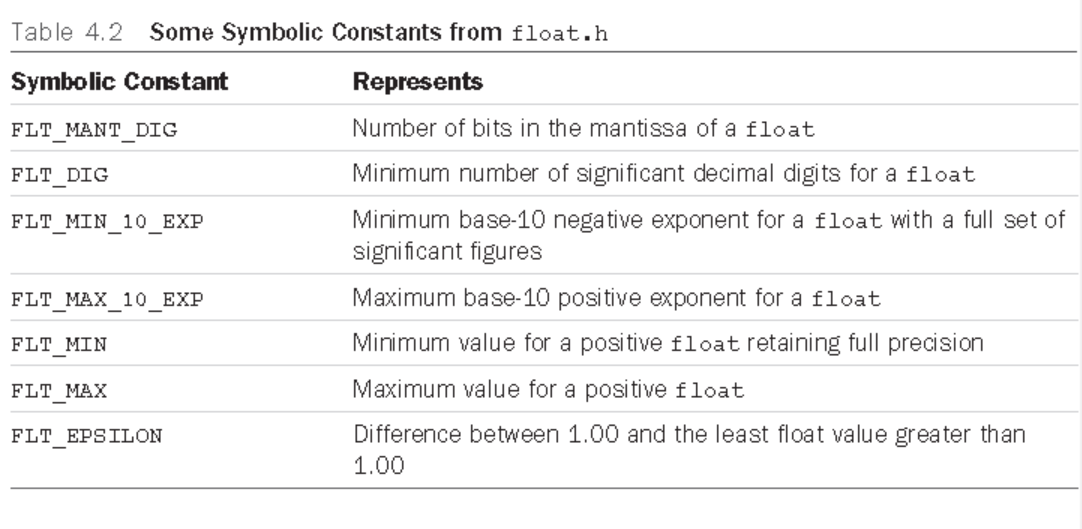
#define INT\_MAX +32767

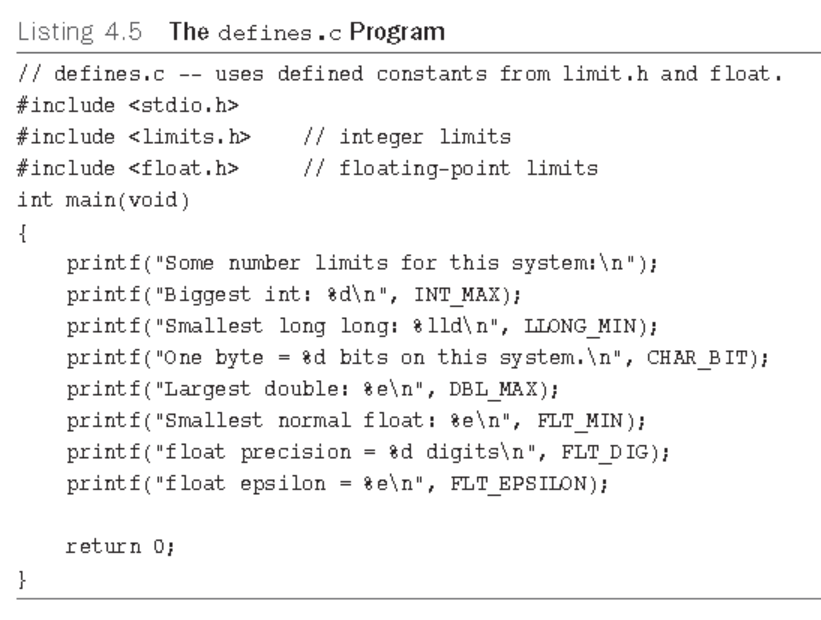
#define INT\_MIN -32768

printf("Maximum int value on this system = %d\n", INT\_MAX);

* symbolic constants의 종류







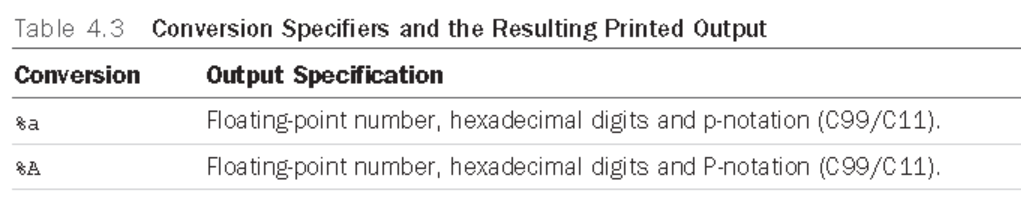
**4.4 printf()과 scanf() 활용**

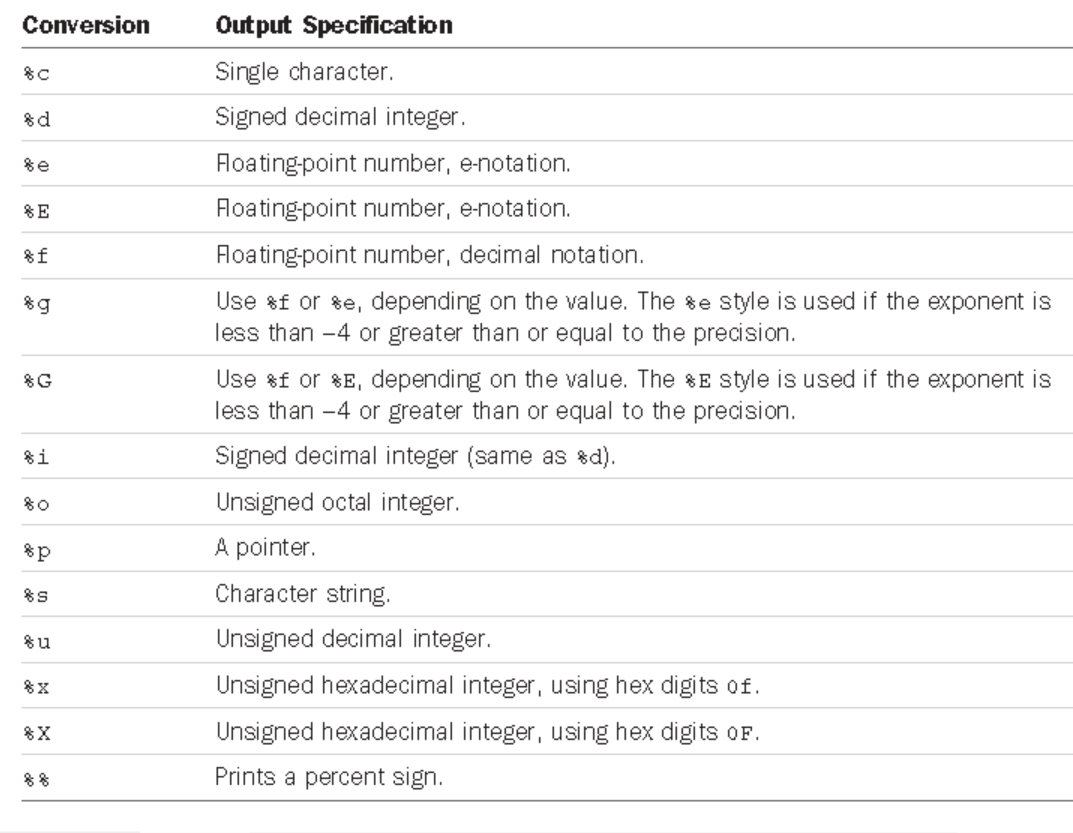
□ printf()와 scanf()를 *input/output functions* , 또는 *I/O functions* 이라 부른다

4.4.1 printf() Function

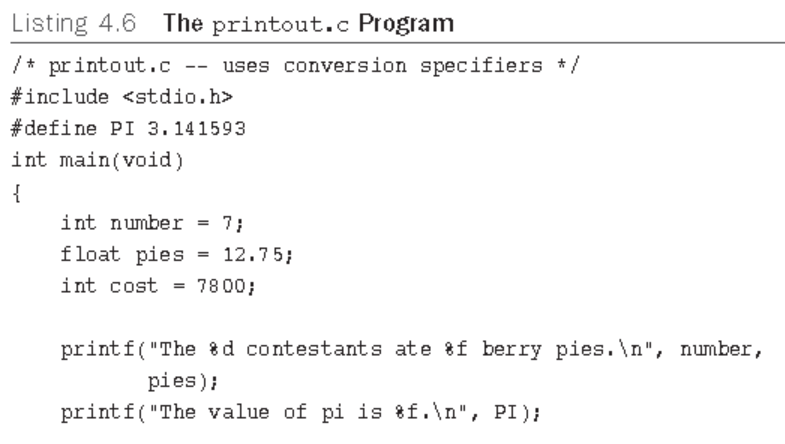
*□ conversion specifications*

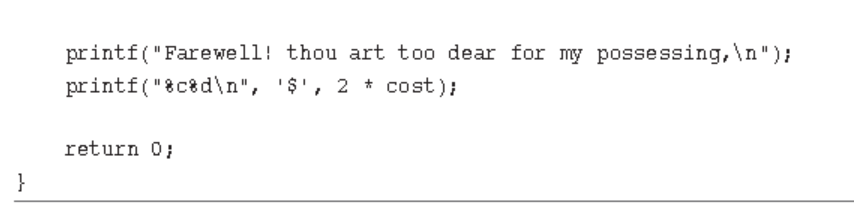
* data가 displayable form으로 convert되는 것을 specify한다
* integer는 %d, character 는 %c 표기를 사용한다.





4.4.2 printf() 사용

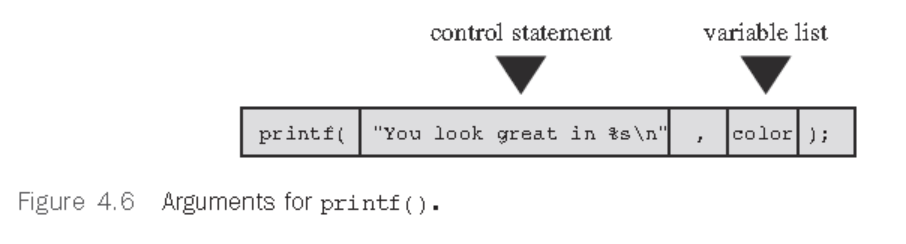


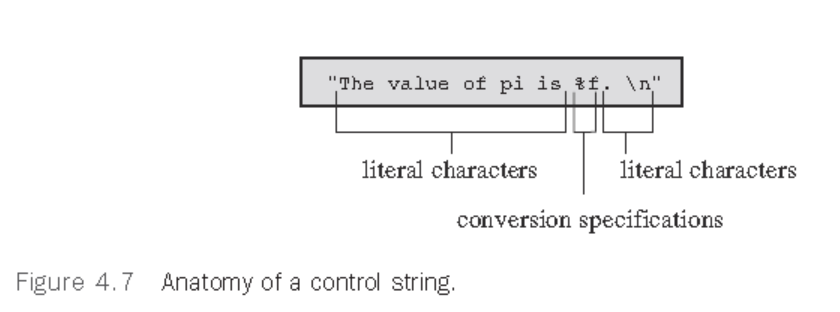


printf( *Control-string* , *item1* , *item2* ,...);

□ 실제로 print되는 characters 표현을 기술

□ Conversion specifications





* % 문자 자체를 출력하려면 %%

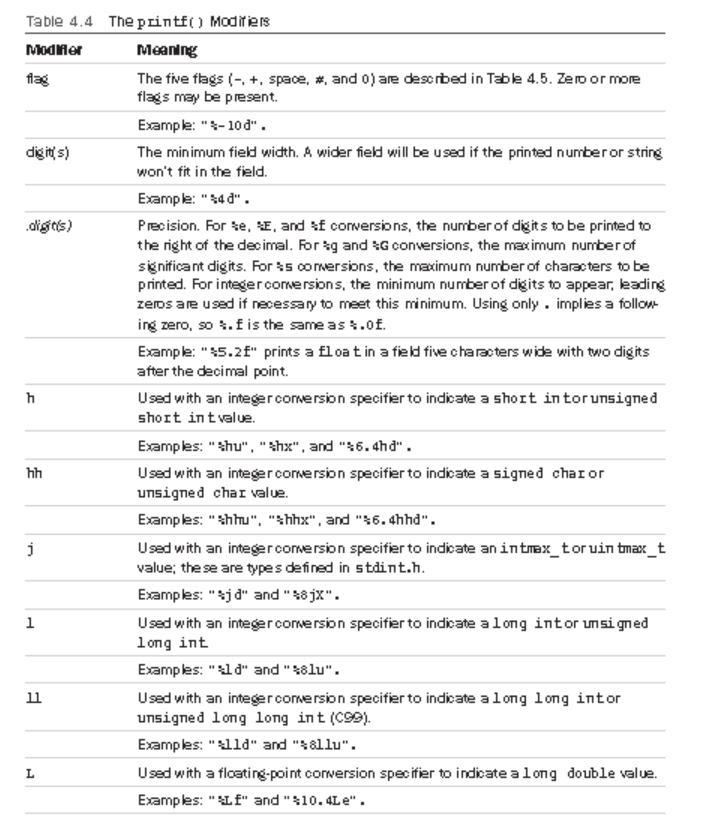
pc = 2\*6;

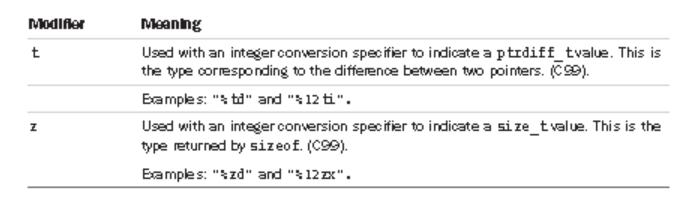
printf("Only %d%% of Sally's gribbles were edible.\n", pc);

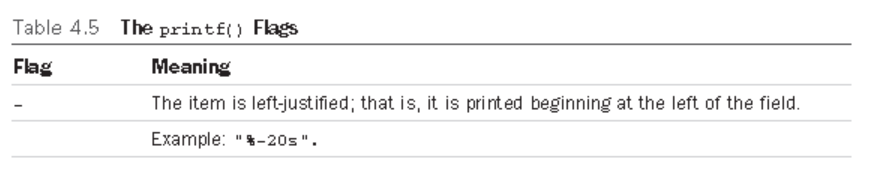
Only 12% of Sally's gribbles were edible.

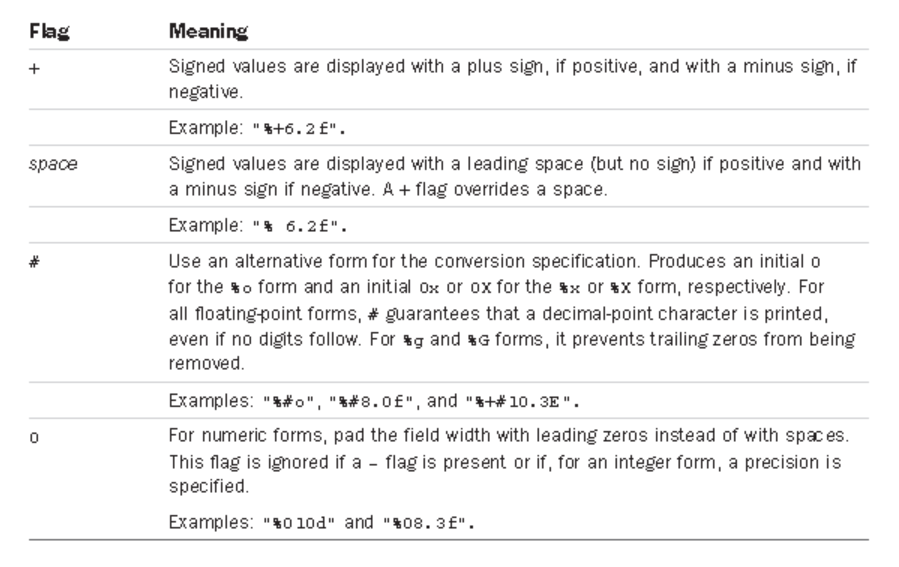
4.4.3 printf()의 Conversion Specification Modifiers

□ conversion specifier의 modifiers: % 와 conversion character 사이에 modifier를 추가

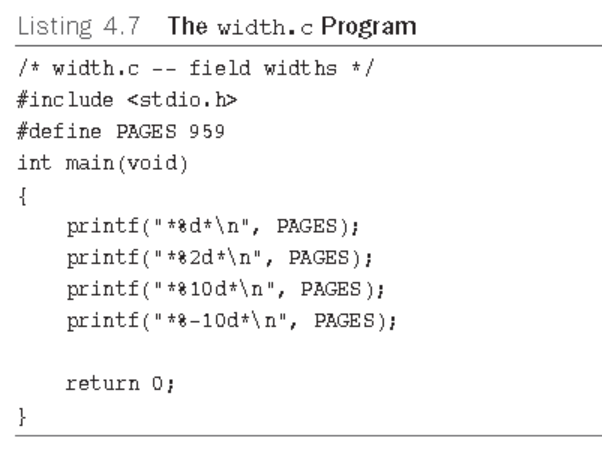






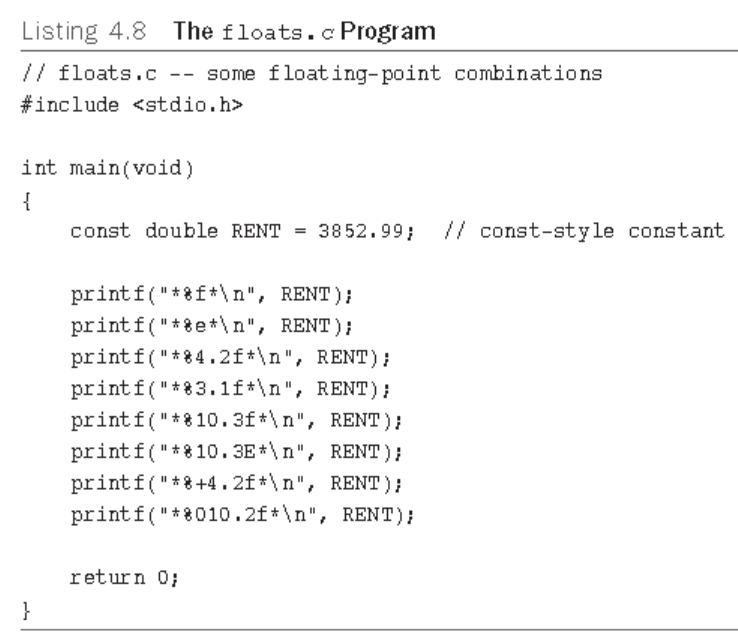


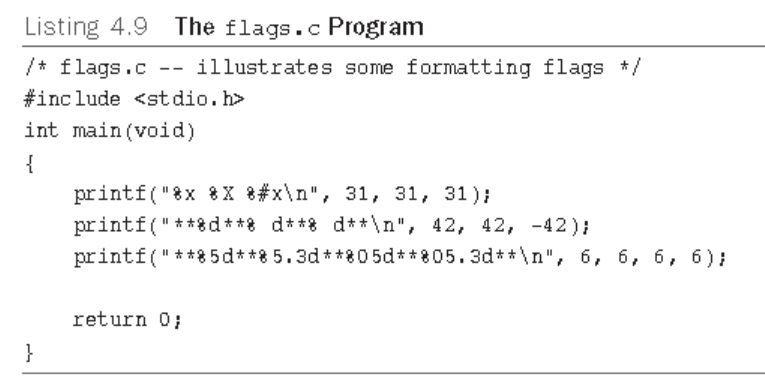
4.4.3.1 Modifiers와 Flags의 사용

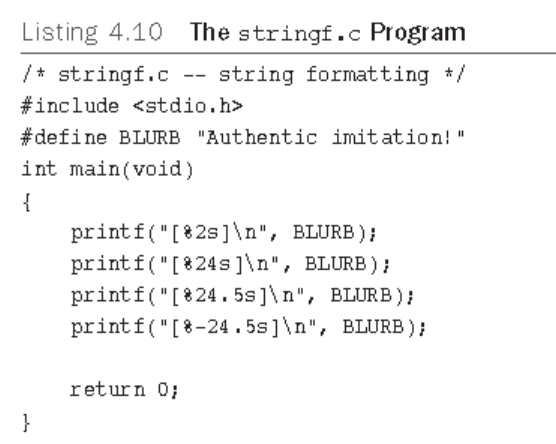


□%2d . 2칸에 출력, PAGES는 3자리이므로 3칸에 출력

□%-10d . 10칸에 출력, -는 left end에 자리 맞춤







// conversion specifications를 작성하는 실습, 출력결과는 퀴즈 대상

4.4.3.2 coding 작성 연습하기

□ 제시된 display 내용을 만들기 위한 printf() statement 작성하는 연습

Display 내용: The NAME family just may be $XXX.XX dollars richer!

printf("The %s family just may be $%.2f richer!\n", name, cash);//작성 능력이 있을 것

4.4.4 Conversion Specification 사용시 주의할 부분?

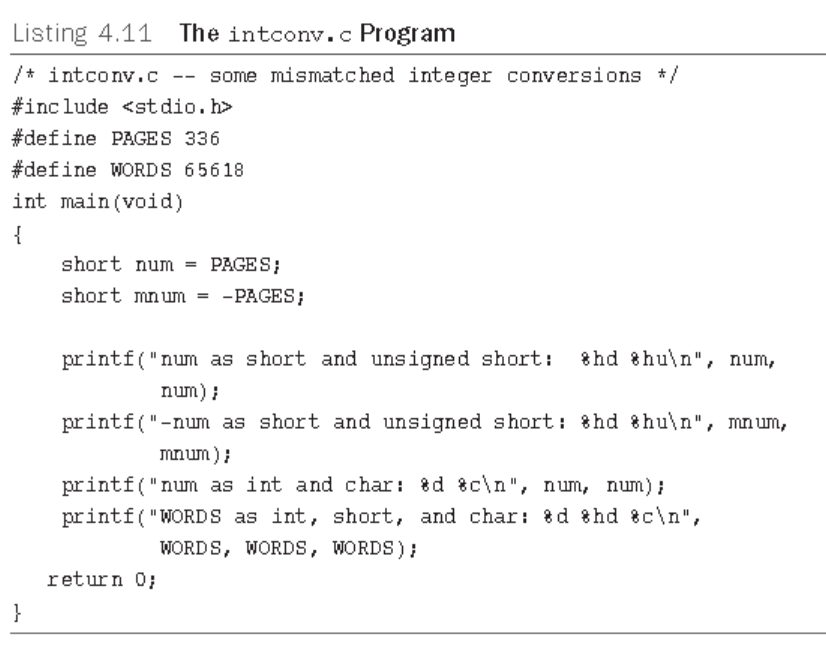
□ Conversion specifications은 데이터 변환하여 출력

* %d 는 정수로 출력

4.4.4.1 Mismatched Conversions

□ type int 값을 print하는데 %d, %x, %o를 사용

□ type double 값을 print하는데 %f , %e , %g 를 사용



* signed integers 를 표현하기 위해 *two’s complement*를 사용
  + 0 ~ 32767: 양의 정수
  + 32768 ~ 65535: 음의 정수, 65535가 −1, 65534 가 −2
  + −336는 65536 - 336 = 65200

▶ 65200는 signed int이면 −336, unsigned int 이면 65200

1 == 0001

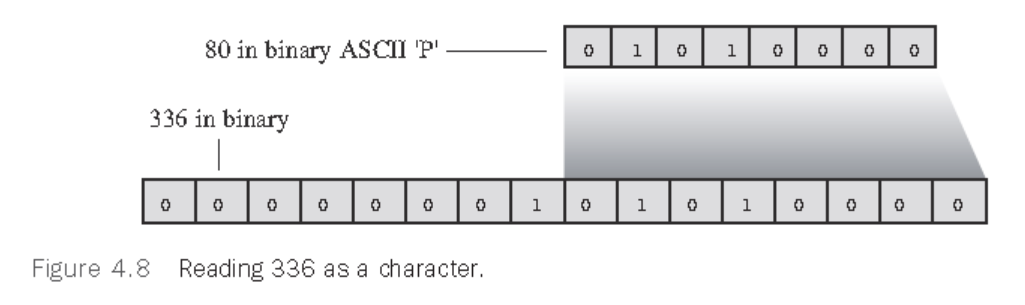
-1 == 1110(1의 보수) or 1111(2의 보수)

-2 == 1101(1의 보수) or 1110(2의 보수)

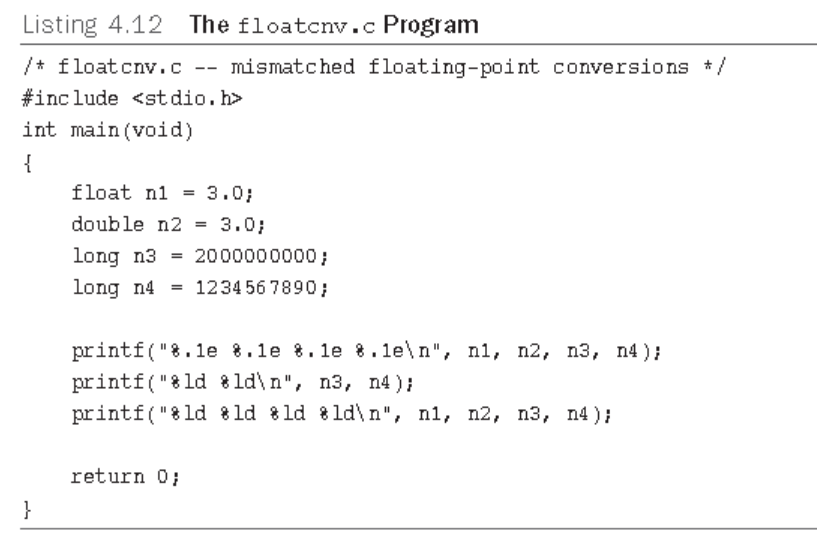
* short int는 2 bytes 이고 char는 1 byte.

• *printf(“%c”, num); //num 값은 336*

* *1char = 1byte = 256 -> modulo 256*
* *256으로 나눈 나머지 = 80에 해당하는 문자 코드 출력*



* printf(“%hd”, WORDS;//WORDS = 65618
  + 65618는 4-byte int 값
  + %hd는 2byte 정수 표기(2 byte 최대값 = 32767)



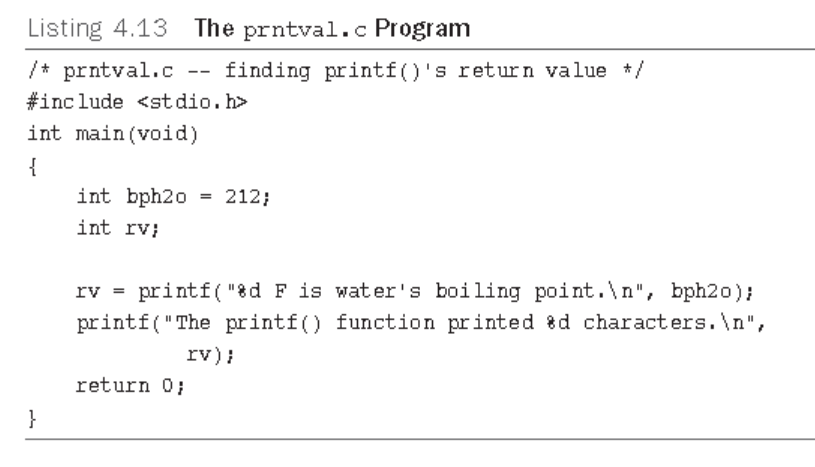
* + integer, floating type의 format specifier가 일치하지 않을 때 엉뚱한 값 출력
    - %e에 대하여 long 출력 문제
    - %ld에 대하여 float, double의 출력문제

4.4.5 printf()의 return 값

□ C function은 일반적으로 return 값을 가짐

□ printf() function은 print되는 number of characters를 return함

* output error가 있으면, printf()는 음의 수를 return한다
* return value을 사용하는 이유는 output errors를 검사하기 위함



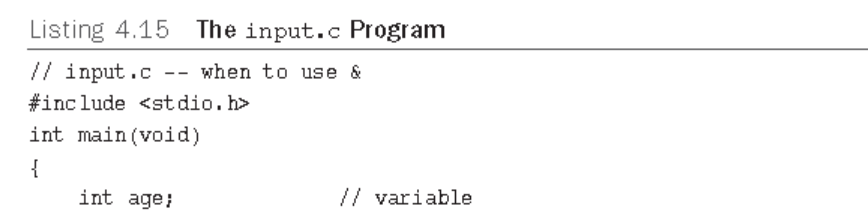
**4.5 scanf() 사용**

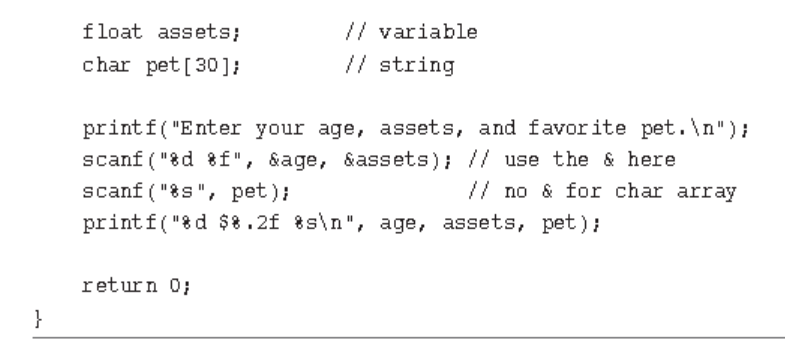
□ 입력의 string character-by-character를 specifier로 변환

* string 입력을 변환: integers, floating-point numbers, characters, and C strings

□ scanf() function는 pointers to variable에 대한 pointer를 사용.

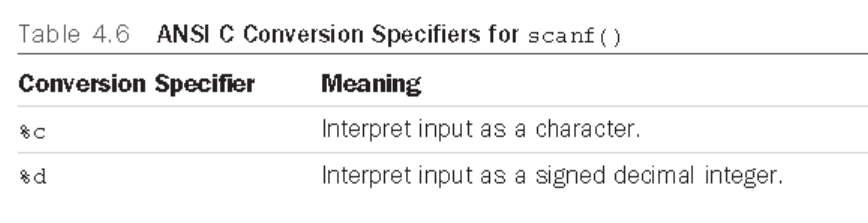
* scanf()를 정수, 실수 등의 숫자 type의 값을 입력시에는 변수 앞에 &를 사용
* scanf()를 string을 읽어 character array에 저장할 때는 &를 사용하지 않음

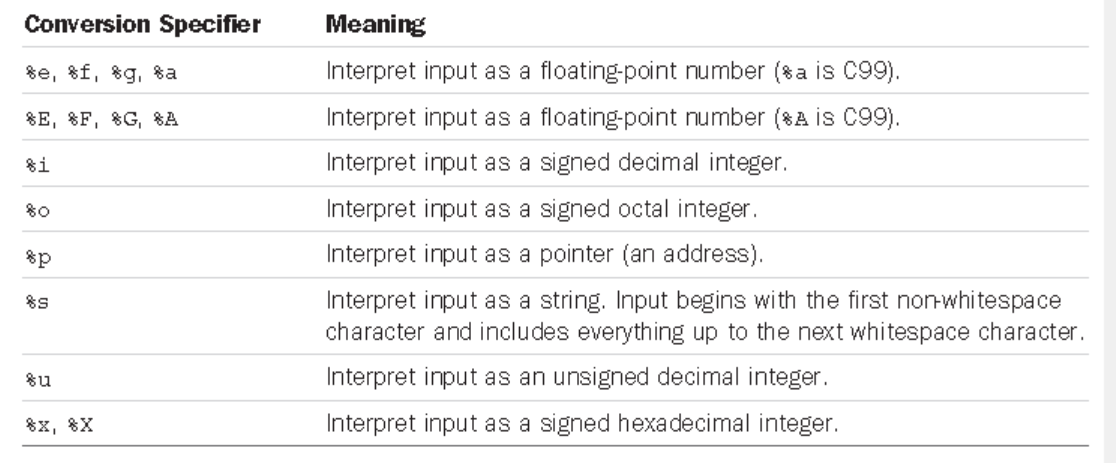


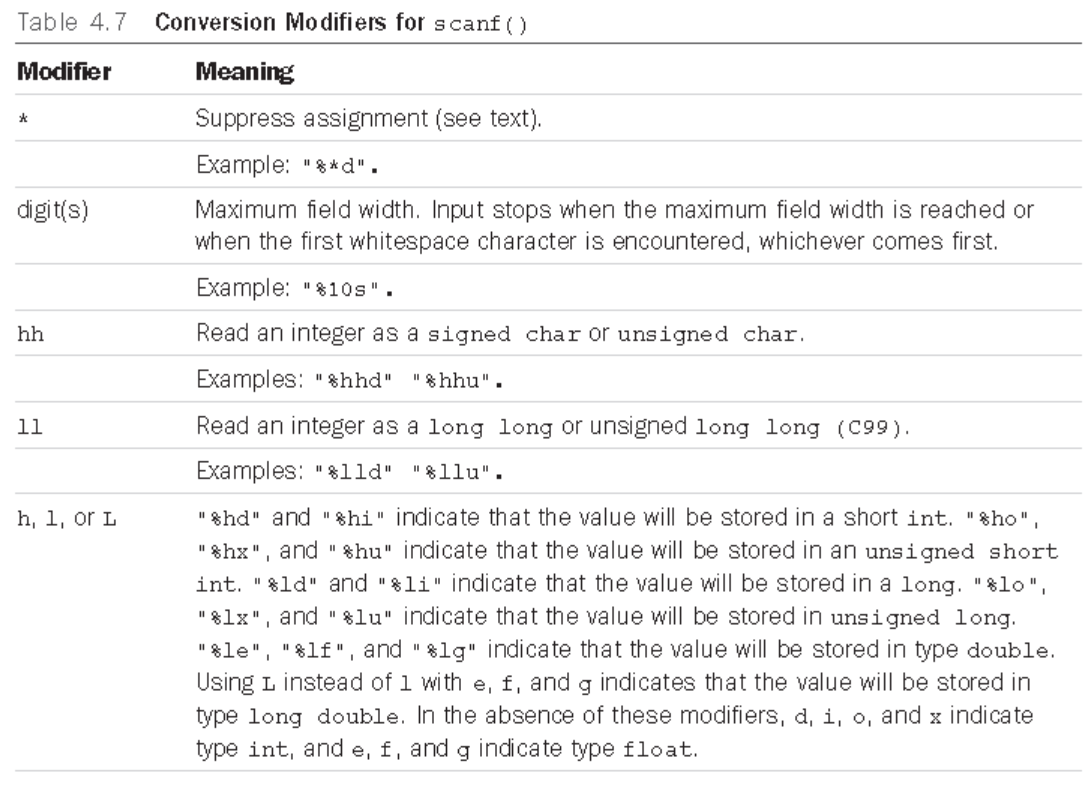


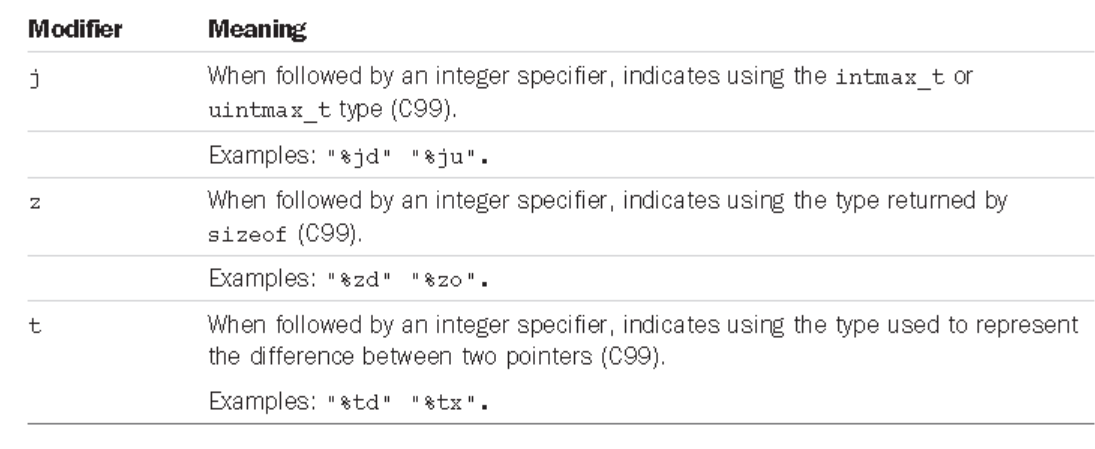
* scanf() function는 whitespace (newlines, tabs, spaces)를 사용하여 입력 변수의 끝을 표시

.









4.5.1 Format String의 Regular Characters

□ specifiers 사이의 comma는 그대로 출력

scanf("%d,%d", &n, &m);

* 88,121

4.5.2 scanf()의 Return Value

□ scanf() function는 read가 완료된 입력 변수의 갯수를return

4.5.3 printf()와 scanf()의 \* Modifier

□ 미리 field width 만큼 이동

* %\*d에 대하여 \*에 대한 값과 d에 대한 argument list를 포함해야 함

