Fundamentals of Computer System
- Operators, Expressions and Statements

민기복

Ki-Bok Min, PhD

서울대학교 에너지자원공학과 조교수
Assistant Professor, Energy Resources Engineering

SEOUL NATIONAL UNIVERSITY

Summary of last lecture

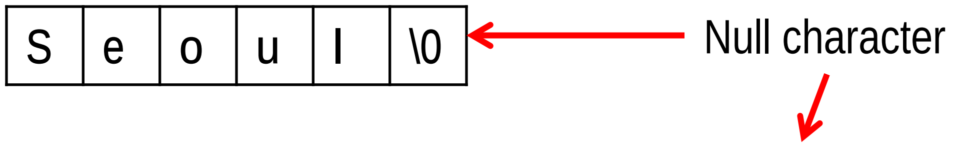
SEOUL NATIONAL UNIVERSITY

- Character string
- printf()
  - format specifier, format modifier, return value
- scanf()
  - format specifier, format modifier, return value
- Miscellaneous
  - Other header files #include <string.h>
  - Manifest constant (명단함수) #define DENSITY 62.4

character string (문자열)

문자열: 널문자로 끝나는 하나의 단위로 취급되는 연속적인 문자

- character string: a series of one or more characters, an array
  - No special variable type – array of type **char**, use “ ”
  - Stored in adjacent memory cells – one character (1 byte) per cell,
  - Null character (널 문자) to mark the end of a string



  - Number of cell = number of characters + 1
  - Array: several memory cells in a row.

Number of elements in the array

char name[40]

array

printf(): an output function (출력함수)

SEOUL NATIONAL UNIVERSITY

- printf(“Control-string”, item1, item2, …);
  - ex) printf(“The value of pi is %6.2f.\n”, PI);

Format modifier (포맷수정자)

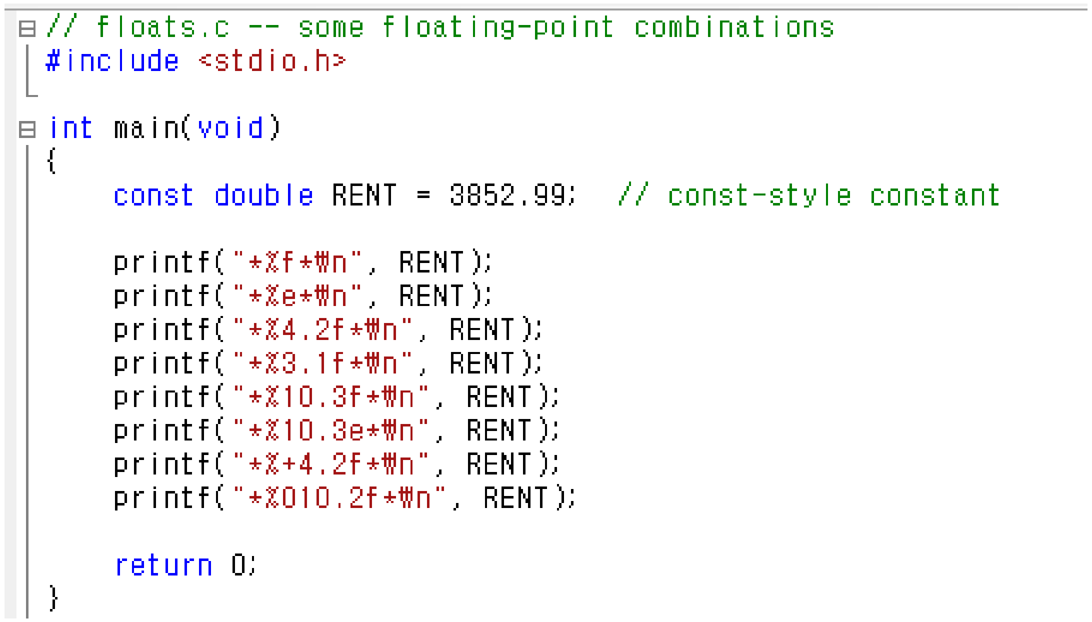
Format specifier (포맷지정자)

- Format Specifier

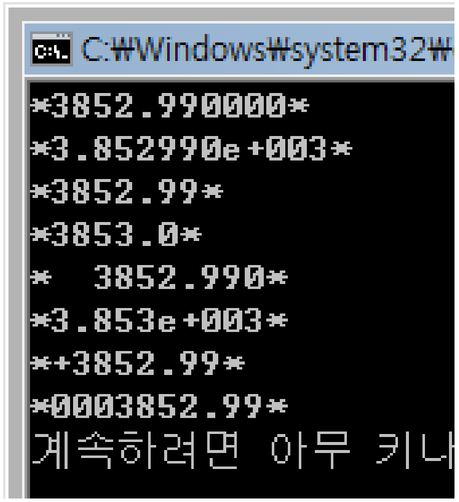
– %d	signed decimal integer
– %c	single character
– %f	floating-point number, decimal notation
– %e	floating-point number, e-notation
– %s	character string
– %u	unsigned decimal integer
– %%	prints a percent sign

printf()
example (1)

SEOUL NATIONAL UNIVERSITY



```
// floats.c -- some floating-point combinations
#include <stdio.h>

int main(void)
{
    const double RENT = 3852.99;  // const-style constant

    printf("*%f*₩n", RENT);
    printf("*%e*₩n", RENT);
    printf("*%4.2f*₩n", RENT);
    printf("*%3.1f*₩n", RENT);
    printf("*%10.3f*₩n", RENT);
    printf("*%10.3e*₩n", RENT);
    printf("*%+4.2f*₩n", RENT);
    printf("*%010.2f*₩n", RENT);

    return 0;
}
```



```
C:₩Windows₩system32₩
*3852.990000*
*3.852990e+003*
*3852.99*
*3853.0*
*  3852.990*
*3.853e+003*
*+3852.99*
*0003852.99*
계속하려면 아무 키나
```

- Default (%f):
  - field width (필드너비): whatever it takes
  - Number of digits to the right of decimal(소수점 아래자리수): 6
- Default (%e):
  - one digit to the left of decimal,
  - six digit to the right

printf() & scanf() return value

SEOUL NATIONAL UNIVERSITY

- printf() : the number of characters it printed (자신이 출력한 문자의 수를 리턴한다).

- scanf() : the number of items that it successfully reads (성공적으로 읽은 항목의 수)

scanf()

SEOUL NATIONAL UNIVERSITY

- **scanf()** function convert string inputs into various forms: integers, floating-point numbers, characters, strings.
- Inverse of **printf(): printf()**와 반대
- scanf("Control-string", &item1, &item2, …);

포멧문자열

전달인자

전달인자의 주소,
printf()와 차이점

  – ex) scanf("%d", &pet);

Mid-term exam

SEOUL NATIONAL UNIVERSITY

- 22 April 13:00 – 15:00
- Venue will be announced later.
- Types of questions;
  - Explanation
  - Multiple choice
  - Short answer
  - Correction
  - Short programming

Mid-term exam examples (1)

SEOUL NATIONAL UNIVERSITY

1. Assume the variable is of type int. Find the value of x.

```
X = (int) 3.8 + 3.3
```

2. What will the following program print?

```
#include <stdio.h>
int main(void)
{
    char c1, c2; Int diff; float num;
    c1 = ‘S’; c2 = ‘O’; diff = cc1 – c2;
    num = diff;
    printf(“%c%c%c:%d %3.2f\n”, c1, c2, c1, diff, num);
    return 0;
}
```

Mid-term exam
example (2)

SEOUL NATIONAL UNIVERSITY

3. Modify the following program so that it prints the letters **a** through **g** instead.

```
#include <stdio.h>
#define TEN 10
int main (void)
 {
      int n = 0;
      while (n++<TEN)
                  printf("%5d", n);
      printf("\n");
      return 0;
}
```

4. Write a program that converts time in minutes to time in hours and minutes. Use **#define** to create a symbolic constant for 60. Use a **while** loop to allow the user to enter values repeatedly and terminate the loop if a value for the time of 0 or less is entered.

Mid-term exam
Strategy

SEOUL NATIONAL UNIVERSITY

- Go through the ‘Review questions’ and ‘Programming exercises’ at the end of each chapter.
- C primer plus 5th Edition (C 기초 플러스 5판)

- Final Exam

Today

SEOUL NATIONAL UNIVERSITY

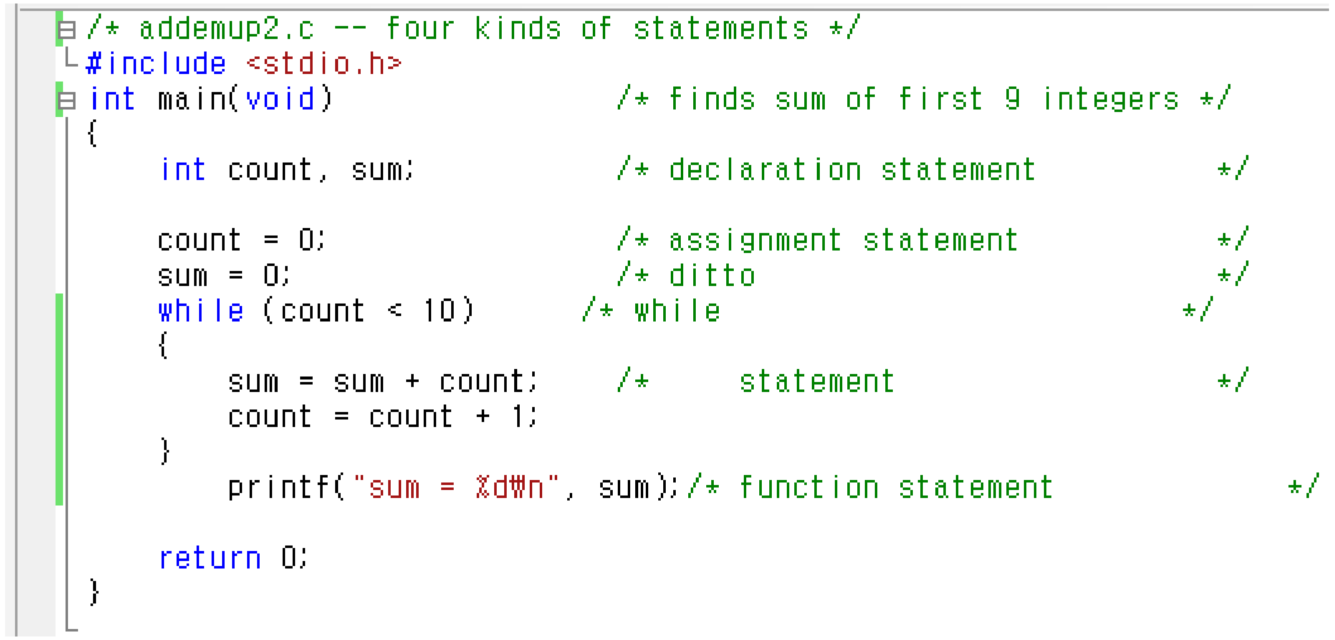
- Operator (연산자):

  = - * % ++ --

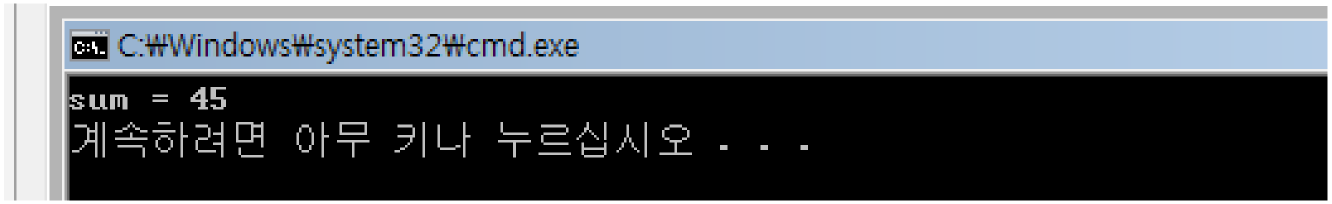
  operator precedence (우선순위)
- while loop
- Automatic type conversion, Type cast (데이터형 캐스트)
- Functions that uses arguments – void pound(n)

while statement

SEOUL NATIONAL UNIVERSITY



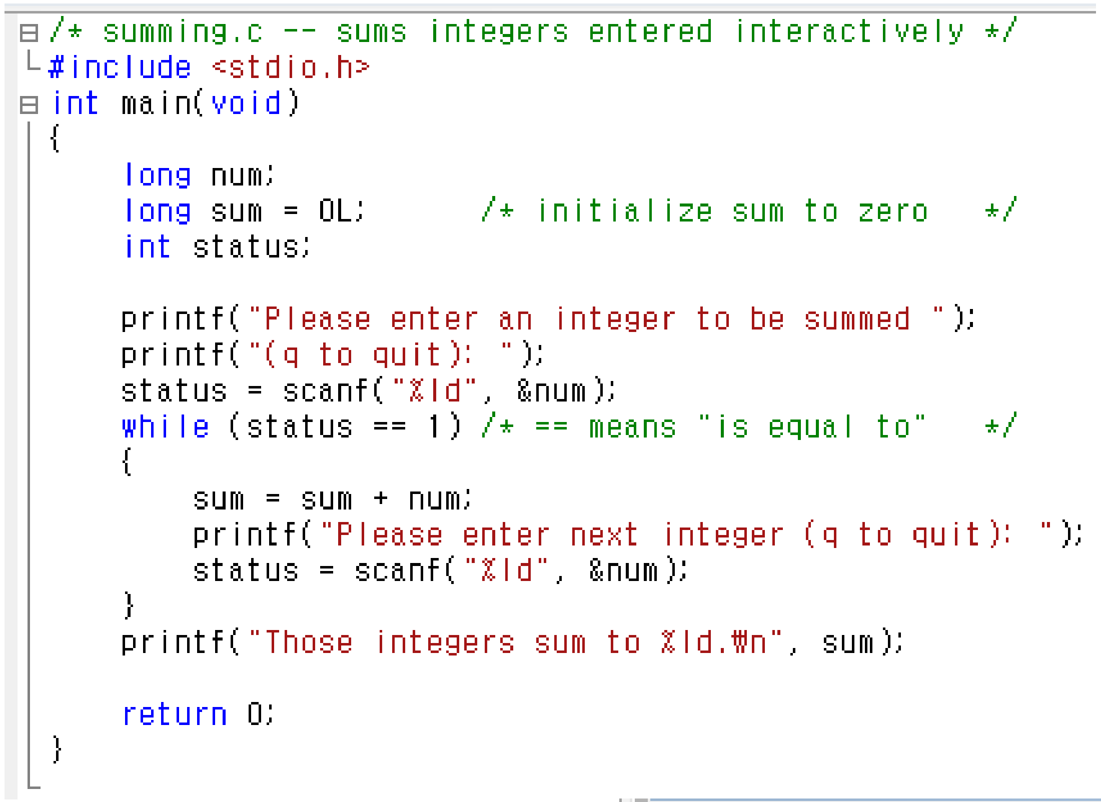
```c
/* addemup2.c -- four kinds of statements */
#include <stdio.h>
int main(void)                /* finds sum of first 9 integers */
{
    int count, sum;           /* declaration statement         */

    count = 0;                /* assignment statement          */
    sum = 0;                  /* ditto                         */
    while (count < 10)      /* while                         */
    {
        sum = sum + count;    /*     statement                 */
        count = count + 1;
    }
        printf("sum = %d₩n", sum);/* function statement         */

    return 0;
}
```



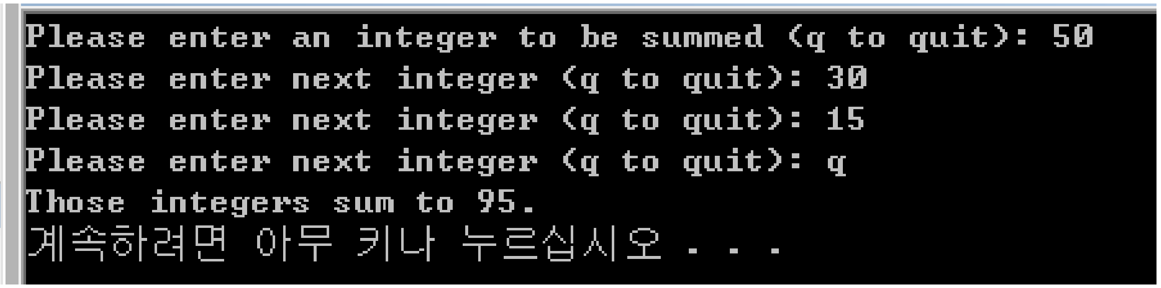
C:₩Windows₩system32₩cmd.exe

```
sum = 45
계속하려면 아무 키나 누르십시오 . . .
```

while statement
example



```
/* summing.c -- sums integers entered interactively */
#include <stdio.h>
int main(void)
{
    long num;
    long sum = 0L;      /* initialize sum to zero   */
    int status;

    printf("Please enter an integer to be summed ");
    printf("(q to quit): ");
    status = scanf("%ld", &num);
    while (status == 1) /* == means "is equal to"   */
    {
        sum = sum + num;
        printf("Please enter next integer (q to quit): ");
        status = scanf("%ld", &num);
    }
    printf("Those integers sum to %ld.₩n", sum);

    return 0;
}
```



```
Please enter an integer to be summed (q to quit): 50
Please enter next integer (q to quit): 30
Please enter next integer (q to quit): 15
Please enter next integer (q to quit): q
Those integers sum to 95.
계속하려면 아무 키나 누르십시오 . . .
```

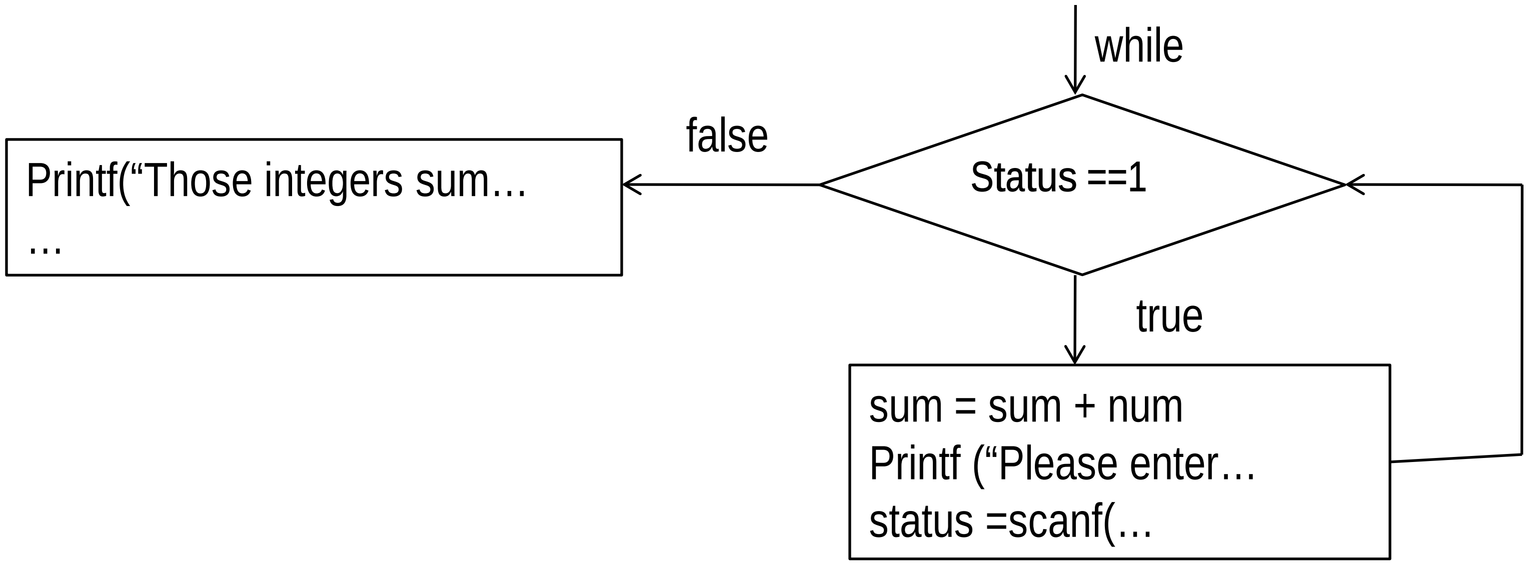
while statement
general form and structure

SEOUL NATIONAL UNIVERSITY

- General form

while (*expression*)

statement --> One statement without {} or a block with {}



while statement
relational operator

SEOUL NATIONAL UNIVERSITY

- Equality operator ==
  - status == 1　　　test whether status is equal to 1.
  - status = 1　　　assign 1 to status.
- If status is 1, loop is iterated for while statement.

operator	Meaning
<	Is less than
<=	Is less than or equal to
==	Is equal to
>=	Is greater than or equal to
>	Is greater than
!=	Is not equal to

Operator (연산자)
Assignment operator (대입연산자) =

SEOUL NATIONAL UNIVERSITY

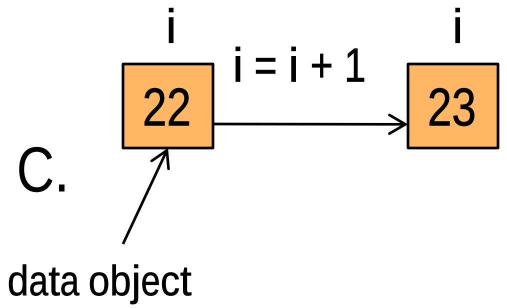
- In C, = doesn't mean 'equals', but it is a value assigning operator.

  <u>football</u> = <u>2002</u>;

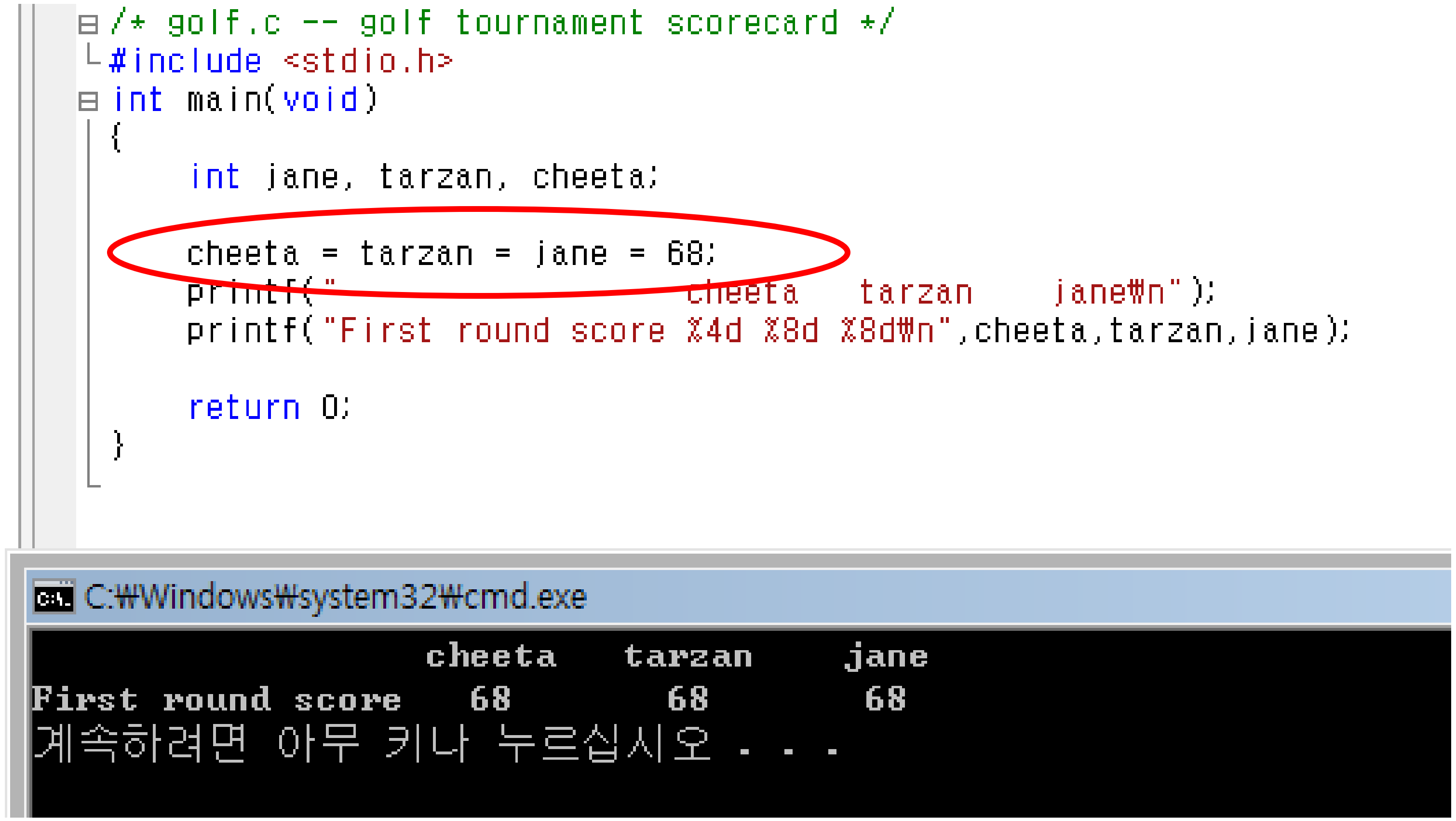
  좌변값(lvalue) 우변값(Rvalue)

  - Assign the value 2002 to the variable named football
  - Direction of operation: ←
  - 2002 = football; //lvalue cannot be a constant

- i = i + 1;
  - Doesn't make sense in math, but it does in C.



Operator (연산자)
Assignment operator (대입연산자) =



```
/* golf.c -- golf tournament scorecard */
#include <stdio.h>
int main(void)
{
    int jane, tarzan, cheeta;

    cheeta = tarzan = jane = 68;
    printf("                    cheeta    tarzan     jane\n");
    printf("First round score %4d %8d %8d\n",cheeta,tarzan,jane);

    return 0;
}
```

C:\Windows\system32\cmd.exe

```
                  cheeta    tarzan     jane
First round score   68        68        68
계속하려면 아무 키나 누르십시오 . . .
```

- Triple assignment is allowed in C.

Operator (연산자)
unary and binary operator

- Addition operator: +
- Subtraction operator: -
- multiplication/division operator: * /

- A = B + C;

Operand (피연산자)

- Income = salary – taxes;

Operator (연산자)

- Binary operator (이항연산자): + - * /

Operator (연산자)
unary and binary operator

SEOUL NATIONAL UNIVERSITY

- Sign operator (부호연산자): + -

  rock = -12;

  dozen = + 12;

- Unary operator (단항연산자): + - ! ++ -- sizeof

  $$\underset{\substack{\uparrow \\ \text{unary operator}}}{-}\,(12 \underset{\substack{\nwarrow \\ \text{binary operator}}}{-} 20);$$

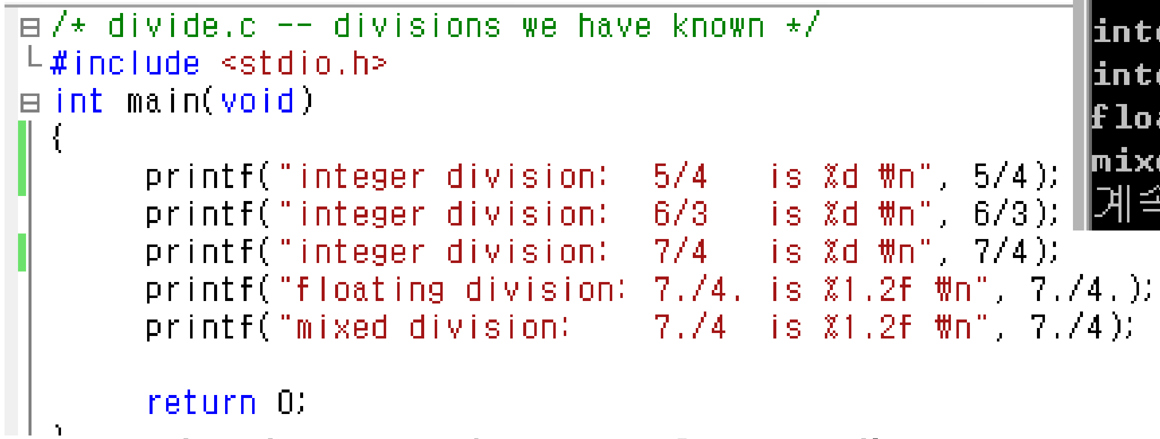
Operator (연산자)
division operator

SEOUL NATIONAL UNIVERSITY

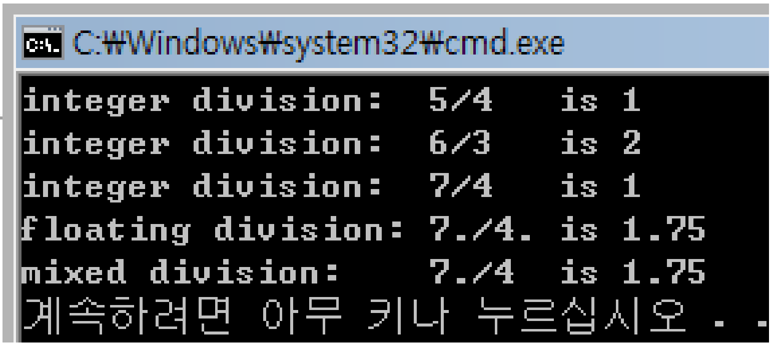
- Division works differently for integer types.

  alpha = 12/7; →

- Any fraction resulting from integer division is discarded: truncation (버림)



```
/* divide.c -- divisions we have known */
#include <stdio.h>
int main(void)
{
    printf("integer division:  5/4   is %d \n", 5/4);
    printf("integer division:  6/3   is %d \n", 6/3);
    printf("integer division:  7/4   is %d \n", 7/4);
    printf("floating division: 7./4. is %1.2f \n", 7./4.);
    printf("mixed division:    7./4  is %1.2f \n", 7./4);

    return 0;
```



C:\Windows\system32\cmd.exe

```
integer division:  5/4   is 1
integer division:  6/3   is 2
integer division:  7/4   is 1
floating division: 7./4. is 1.75
mixed division:    7./4  is 1.75
계속하려면 아무 키나 누르십시오 . .
```

- Mixed type: 7./4 → compiler converts the integer to floating point before division.

How about -3.8? -3 → 0을 향해서 버려라

Operator (연산자)
Increment/Decrement Operator

SEOUL NATIONAL UNIVERSITY

- Increment Operator(증가연산자): increases the value of its operand by 1.

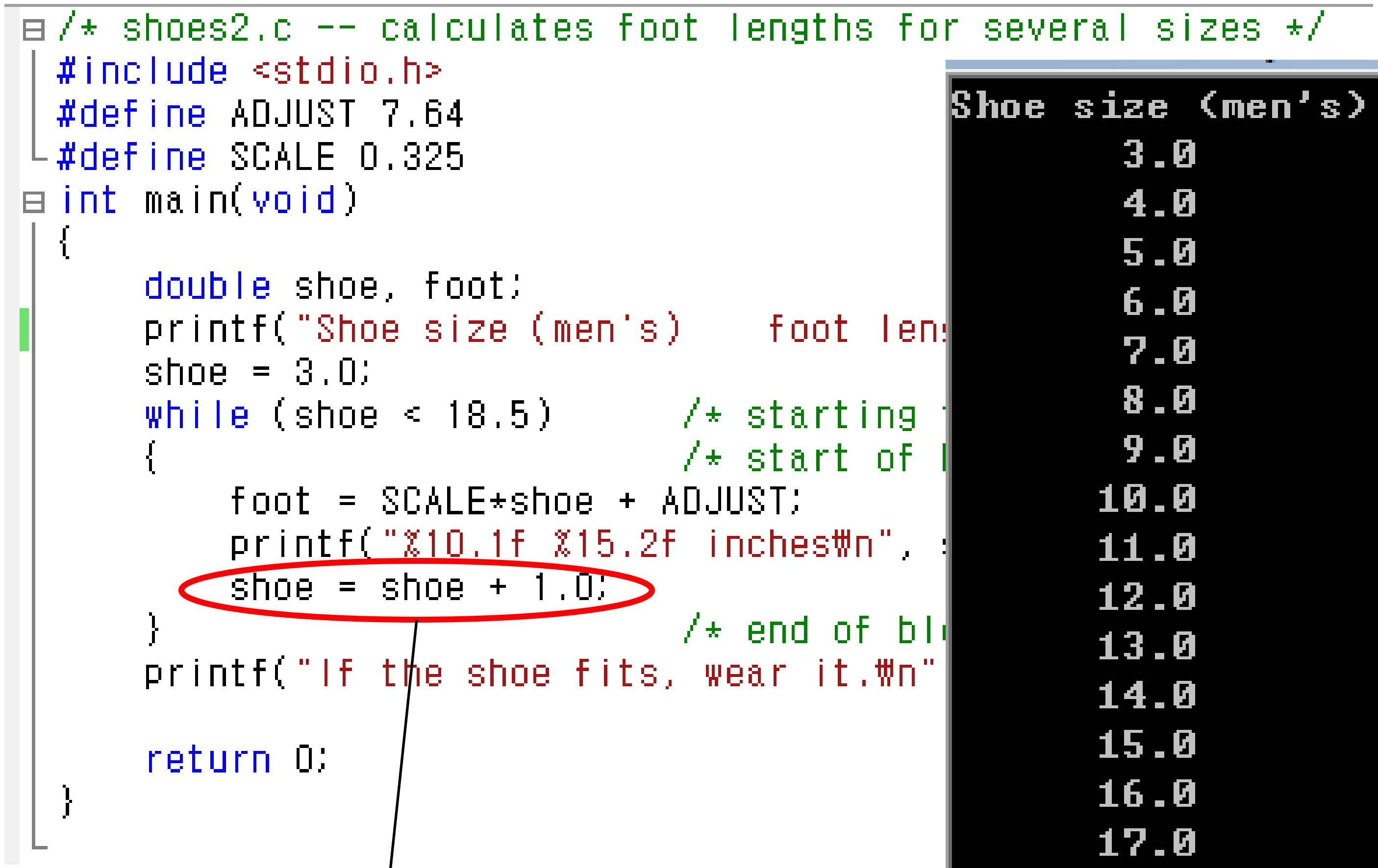
  a++; $\rightarrow$ a = a + 1;

- Two types;
  - Prefix (전위모드): ++a
  - Postfix (후위 모드): a++

Operator (연산자)

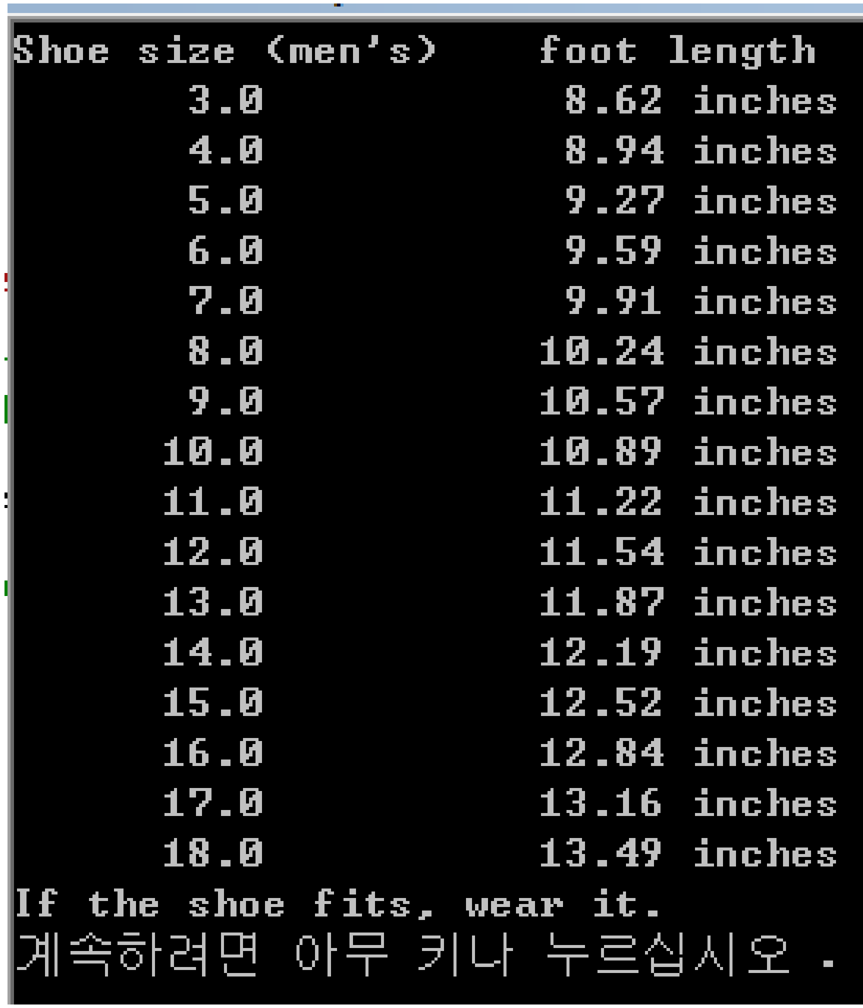
++ -- (example)

SEOUL NATIONAL UNIVERSITY



```
/* shoes2.c -- calculates foot lengths for several sizes */
#include <stdio.h>
#define ADJUST 7.64
#define SCALE 0.325
int main(void)
{
    double shoe, foot;
    printf("Shoe size (men's)    foot len
    shoe = 3.0;
    while (shoe < 18.5)          /* starting
    {                            /* start of 
        foot = SCALE*shoe + ADJUST;
        printf("%10.1f %15.2f inches\n", 
        shoe = shoe + 1.0;
    }                            /* end of bl
    printf("If the shoe fits, wear it.\n"

    return 0;
}
```

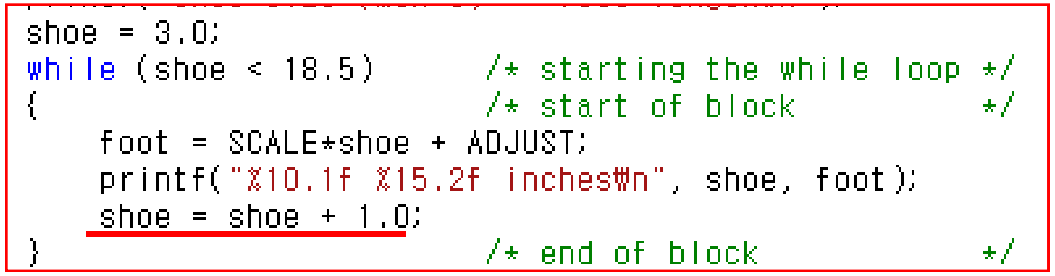
++shoe



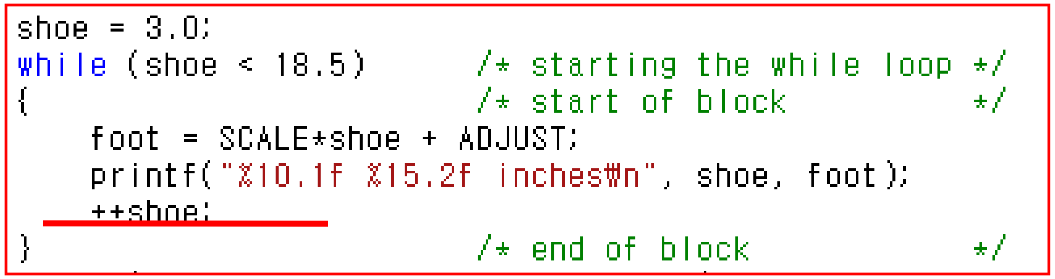
```
Shoe size (men's)     foot length
        3.0              8.62 inches
        4.0              8.94 inches
        5.0              9.27 inches
        6.0              9.59 inches
        7.0              9.91 inches
        8.0             10.24 inches
        9.0             10.57 inches
       10.0             10.89 inches
       11.0             11.22 inches
       12.0             11.54 inches
       13.0             11.87 inches
       14.0             12.19 inches
       15.0             12.52 inches
       16.0             12.84 inches
       17.0             13.16 inches
       18.0             13.49 inches
If the shoe fits, wear it.
계속하려면 아무 키나 누르십시오 .
```

Operator (연산자)
++ -- (example)

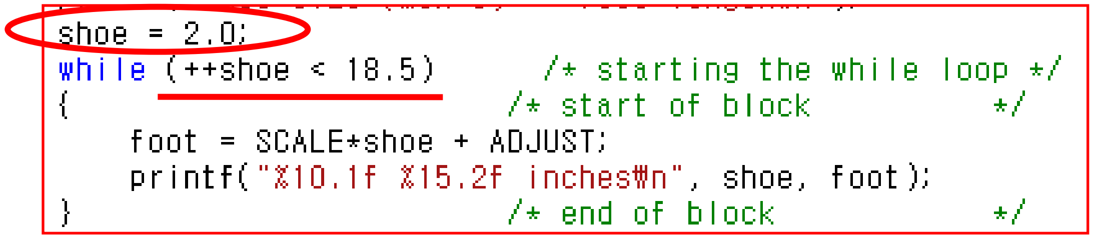
- Same results with the following codes.



```
shoe = 3.0;
while (shoe < 18.5)       /* starting the while loop */
{                         /* start of block          */
    foot = SCALE*shoe + ADJUST;
    printf("%10.1f %15.2f inches₩n", shoe, foot);
    shoe = shoe + 1.0;
}                         /* end of block            */
```



```
shoe = 3.0;
while (shoe < 18.5)       /* starting the while loop */
{                         /* start of block          */
    foot = SCALE*shoe + ADJUST;
    printf("%10.1f %15.2f inches₩n", shoe, foot);
    ++shoe;
}                         /* end of block            */
```



```
shoe = 2.0;
while (++shoe < 18.5)       /* starting the while loop */
{                         /* start of block          */
    foot = SCALE*shoe + ADJUST;
    printf("%10.1f %15.2f inches₩n", shoe, foot);
}                         /* end of block            */
```

- The value of shoe is increased by 1 and then compared with 18.5
-Initial was changed from 3 to 2.
why?
- Two processes controlling the loop in one place!!!

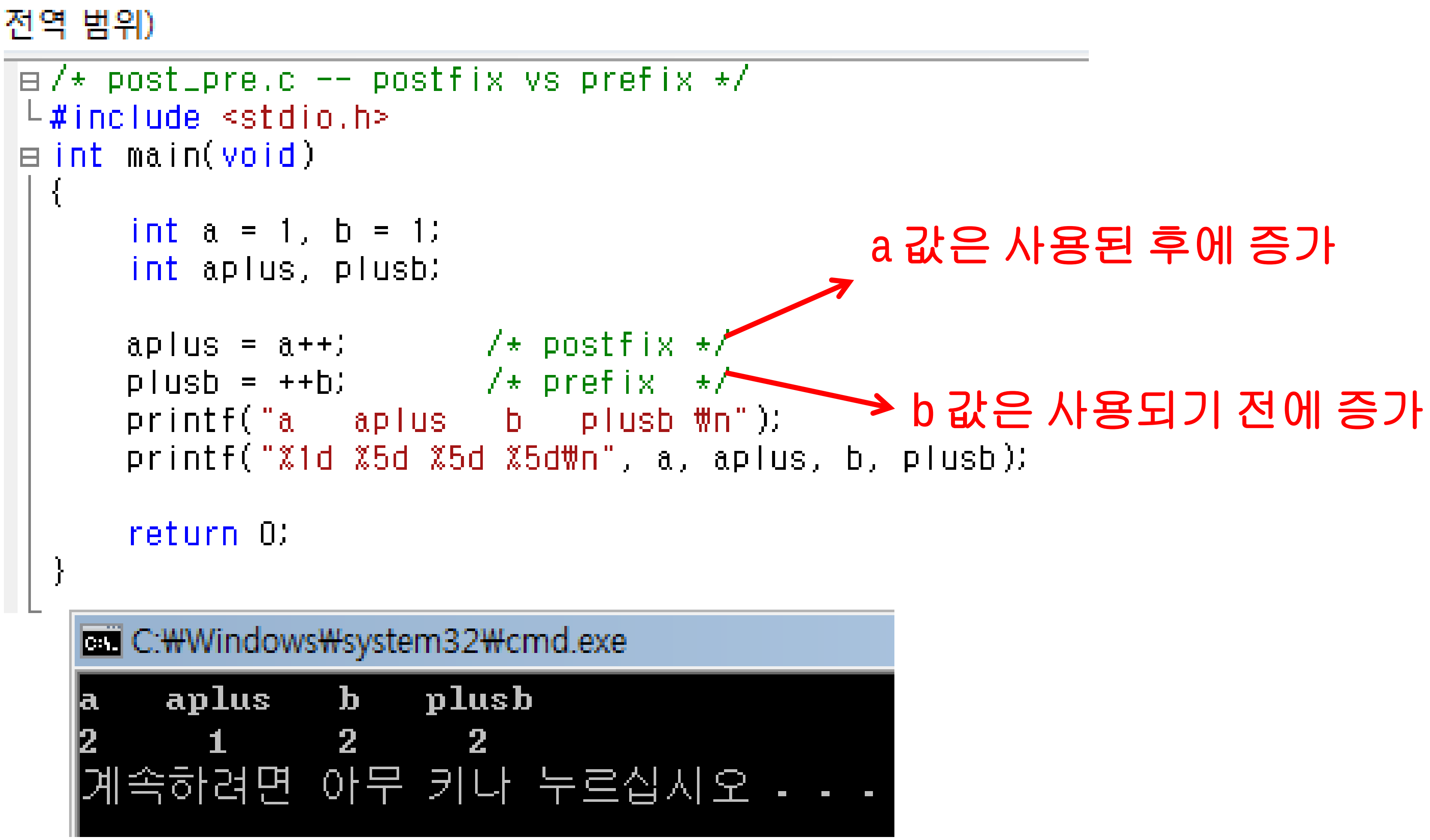
Operator (연산자)

++ --

- advantage:
  - Compact form
  - Very useful for **loop**
  - Similar to actual machine language instructions
- Disadvantage
  - Easier to make errors

Operator (연산자)
a++   vs. ++a-

SEOUL NATIONAL UNIVERSITY



전역 범위)

```
/* post_pre.c -- postfix vs prefix */
#include <stdio.h>
int main(void)
{
    int a = 1, b = 1;
    int aplus, plusb;

    aplus = a++;        /* postfix */
    plusb = ++b;        /* prefix  */
    printf("a   aplus   b   plusb \n");
    printf("%1d %5d %5d %5d\n", a, aplus, b, plusb);

    return 0;
}
```

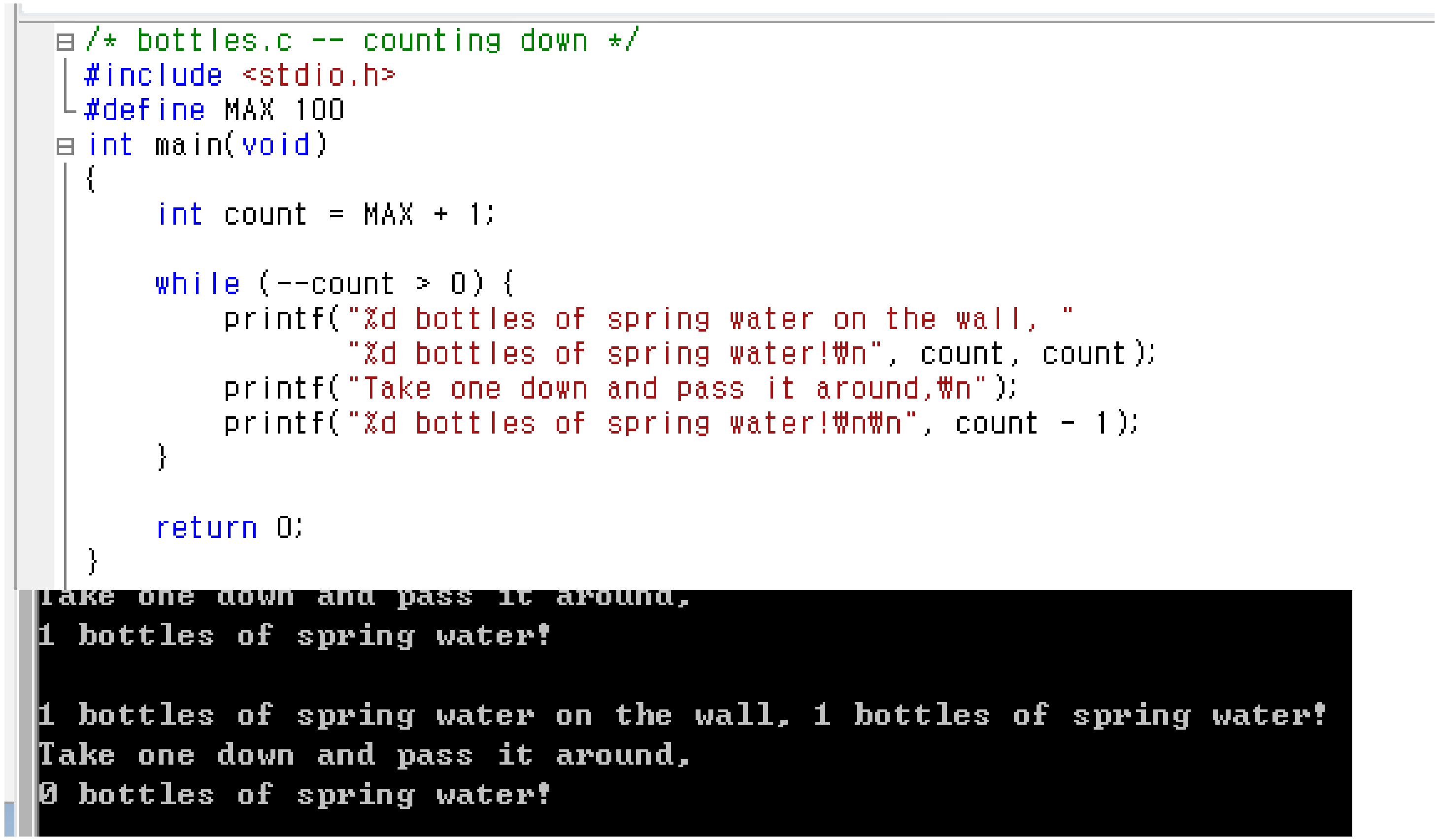
```
a    aplus    b    plusb
2      1      2      2
계속하려면 아무 키나 누르십시오 . . .
```

- When a++ or ++a is used alone, it does not matter

Operator (연산자)
Decrement operator(감소연산자)

SEOUL NATIONAL UNIVERSITY

- Decrement operator use -- instead of ++



```
/* bottles.c -- counting down */
#include <stdio.h>
#define MAX 100
int main(void)
{
     int count = MAX + 1;

     while (--count > 0) {
          printf("%d bottles of spring water on the wall, "
                 "%d bottles of spring water!₩n", count, count);
          printf("Take one down and pass it around,₩n");
          printf("%d bottles of spring water!₩n₩n", count - 1);
     }

     return 0;
}
```

```
Take one down and pass it around,
1 bottles of spring water!

1 bottles of spring water on the wall, 1 bottles of spring water!
Take one down and pass it around,
0 bottles of spring water!
```

Operator (연산자)
++ -- Precedence (우선순위)

SEOUL NATIONAL UNIVERSITY

ans = num/2 + 5 * (1 + num++);   ?????

n=3;

y = n++ + n++;   6 or 7 depending on compiler. No standard.

- Don’t use ++ or -- on a variable that;
  - is part of more than one argument of a function.
  - Appears more than once in an expression

Operator (연산자)
sizeof

- sizeof : returns the size of its operand (in bytes)
- Operand can be a specific data object
  - Ex)
    ```
    char name[40];
    sizeof name;
    ```
- Operand can be a type (such as float). Use () in this case
  - Ex) sizeof (int)

Operator (연산자)
modulus operator, % (나머지 연산자)

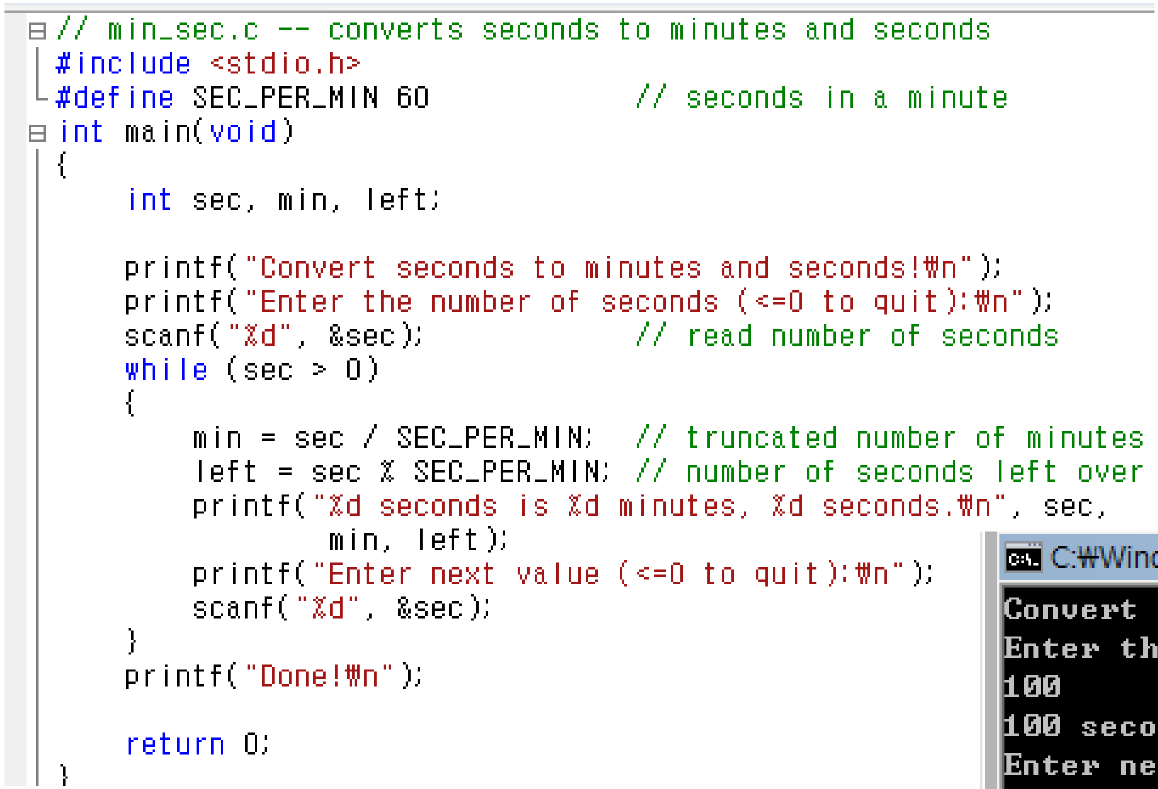
SEOUL NATIONAL UNIVERSITY

- Gives the remainder that results when the integer in the left is divided by the integer in the right.
- 13%5   is 3
- Can be very useful.
- Negative numbers? - follows the sign of first operand

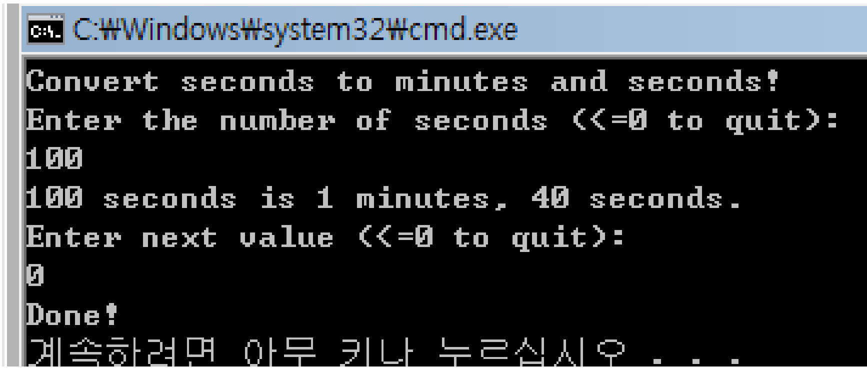
11/5 is 2	11%5 is 1
11/-5 is -2	11%-5 is 1
-11/-5 is 2	-11%-5 is -1
-11/5 is -2	-11%5 is -1

a%b=a-(a/b)*b

Operator (연산자)
modulus operator, % (나머지 연산자)



```c
// min_sec.c -- converts seconds to minutes and seconds
#include <stdio.h>
#define SEC_PER_MIN 60            // seconds in a minute
int main(void)
{
    int sec, min, left;

    printf("Convert seconds to minutes and seconds!₩n");
    printf("Enter the number of seconds (<=0 to quit):₩n");
    scanf("%d", &sec);            // read number of seconds
    while (sec > 0)
    {
        min = sec / SEC_PER_MIN;  // truncated number of minutes
        left = sec % SEC_PER_MIN; // number of seconds left over
        printf("%d seconds is %d minutes, %d seconds.₩n", sec,
                min, left);
        printf("Enter next value (<=0 to quit):₩n");
        scanf("%d", &sec);
    }
    printf("Done!₩n");

    return 0;
}
```



C:₩Windows₩system32₩cmd.exe

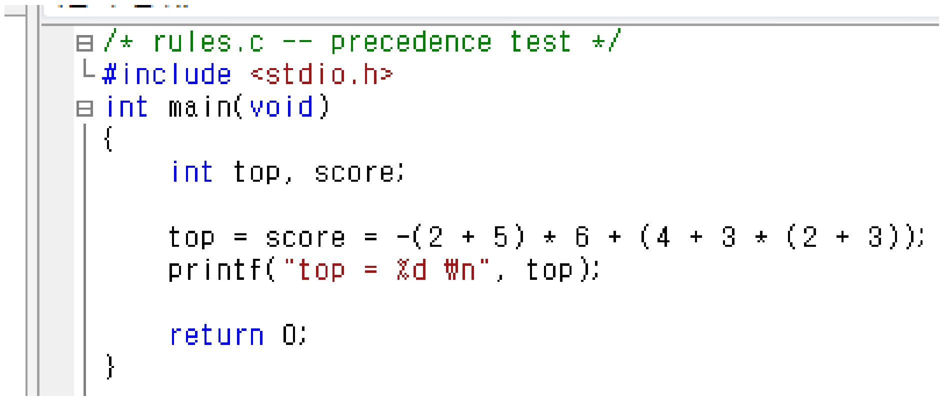
```
Convert seconds to minutes and seconds!
Enter the number of seconds (<=0 to quit):
100
100 seconds is 1 minutes, 40 seconds.
Enter next value (<=0 to quit):
0
Done!
계속하려면 아무 키나 누르십시오 . . .
```

Operator (연산자)
Operator Precedence (우선순위)

SEOUL NATIONAL UNIVERSITY

- Each operator is assigned a *precedence* level.

operator	Associativity
()	→
+ - (unary), ++ --	←
* /	→
+ - (binary)	→
=	←



```
/* rules.c -- precedence test */
#include <stdio.h>
int main(void)
{
    int top, score;

    top = score = -(2 + 5) * 6 + (4 + 3 * (2 + 3));
    printf("top = %d ₩n", top);

    return 0;
}
```

  - ex) x*y++ (x*y)++ or $\underline{\text{(x) * (y++)}}$

- Don’t confuse precedence with the order of evaluation (a++ or ++a)

- y=2; n=3; nextnum = (y + n++)*6 (2 + 3)*6 = 30, n=4

Expression (수식) and Statement(명령문)
Expression

- Expression (수식): consists of a combination of operators and operands.
  - Ex)

    4

    -6

    a*(b+c/d)/20

    a>3

Expression (수식) and Statement(명령문)

Expression

SEOUL NATIONAL UNIVERSITY

- Every expression has a value
- With = sign, the same value in the left
- Relational expression (q>3):
  - True: 1
  - False: 0

expression	Value
-4+6	2
c = 3 + 8	11
5 > 3	1
6 + (c = 3 + 8)	17
q = 5 * 2	10

Looks strange but legal in C $\longrightarrow$ 6 + (c = 3 + 8)

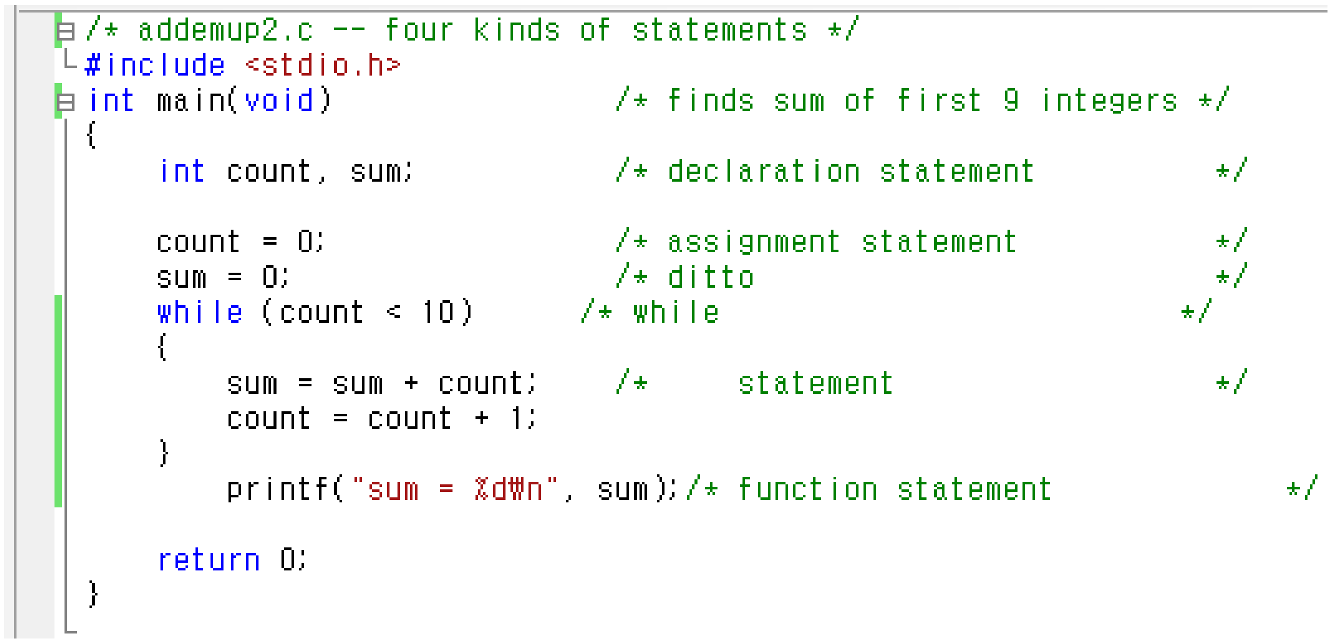
Expression (수식) and Statement(명령문)
Statement

- Statements are;
  - Complete instructions to the computer,
  - Primary building blocks of a program, and
  - Indicated by semicolon (;).
- C considers any expression to be a statement if you append a semicolon.
  - Expression statements (수식명령문)

```
8;
3+4;
```

Expression (수식) and Statement(명령문)
Types of statement

SEOUL NATIONAL UNIVERSITY

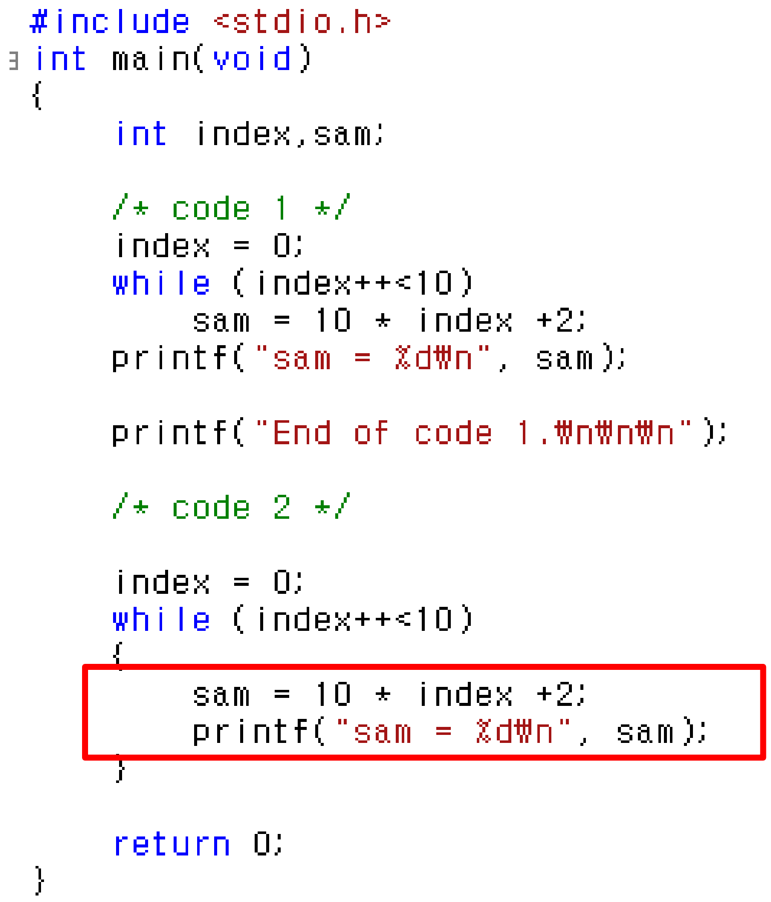


```c
/* addemup2.c -- four kinds of statements */
#include <stdio.h>
int main(void)                /* finds sum of first 9 integers */
{
    int count, sum;           /* declaration statement          */

    count = 0;                /* assignment statement           */
    sum = 0;                  /* ditto                          */
    while (count < 10)      /* while                          */
    {
        sum = sum + count;    /*      statement                 */
        count = count + 1;
    }
        printf("sum = %d₩n", sum);/* function statement             */

    return 0;
}
```

- Declaration statement (선언명령문)
- Assignment statement (대입명령문)
- Structured statement (구조화 명령문)
- Function statement (함수호출 명령문)

Expression (수식) and Statement(명령문) compound statements (blocks)

- A compound statement is two or more statements grouped together by enclosing them in braces {} -also called a block.



```
#include <stdio.h>
int main(void)
{
    int index,sam;

    /* code 1 */
    index = 0;
    while (index++<10)
        sam = 10 * index +2;
    printf("sam = %d₩n", sam);

    printf("End of code 1.₩n₩n₩n");

    /* code 2 */

    index = 0;
    while (index++<10)
    {
        sam = 10 * index +2;
        printf("sam = %d₩n", sam);
    }

    return 0;
}
```

{복합명령문}

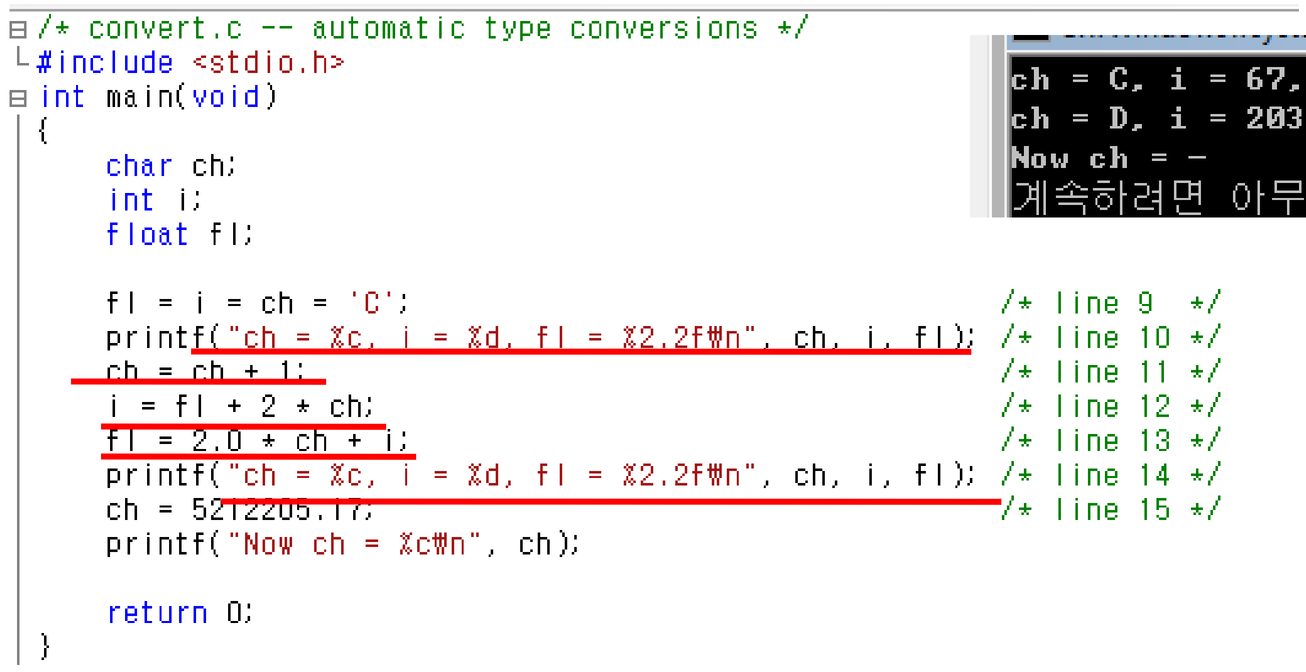
Type Conversion (형변환)

SEOUL NATIONAL UNIVERSITY

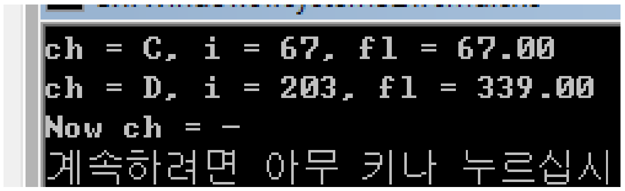
- Statements and expressions should use variables and constants of just one type.
- You mix types, C uses a set of rules to make type conversions autumatically.
  - Char & short → int (promotions, 올림변환)
  - Any two types → higher rankings
    - (High to low) Double - float – unsigned long – long – unsigned int - int
  - Final result of the calculations → type of the variables
  - When passed as function arguments,

    char and short → int   float → double

Type Conversion (형변환)
Example



```
/* convert.c -- automatic type conversions */
#include <stdio.h>
int main(void)
{
    char ch;
    int i;
    float fl;

    fl = i = ch = 'C';                                  /* line 9  */
    printf("ch = %c, i = %d, fl = %2.2f₩n", ch, i, fl); /* line 10 */
    ch = ch + 1;                                        /* line 11 */
    i = fl + 2 * ch;                                    /* line 12 */
    fl = 2.0 * ch + i;                                  /* line 13 */
    printf("ch = %c, i = %d, fl = %2.2f₩n", ch, i, fl); /* line 14 */
    ch = 5212205.17;                                    /* line 15 */
    printf("Now ch = %c₩n", ch);

    return 0;
}
```



```
ch = C, i = 67, fl = 67.00
ch = D, i = 203, fl = 339.00
Now ch = -
계속하려면 아무 키나 누르십시
```

Cast operator (캐스트 연산자)

SEOUL NATIONAL UNIVERSITY

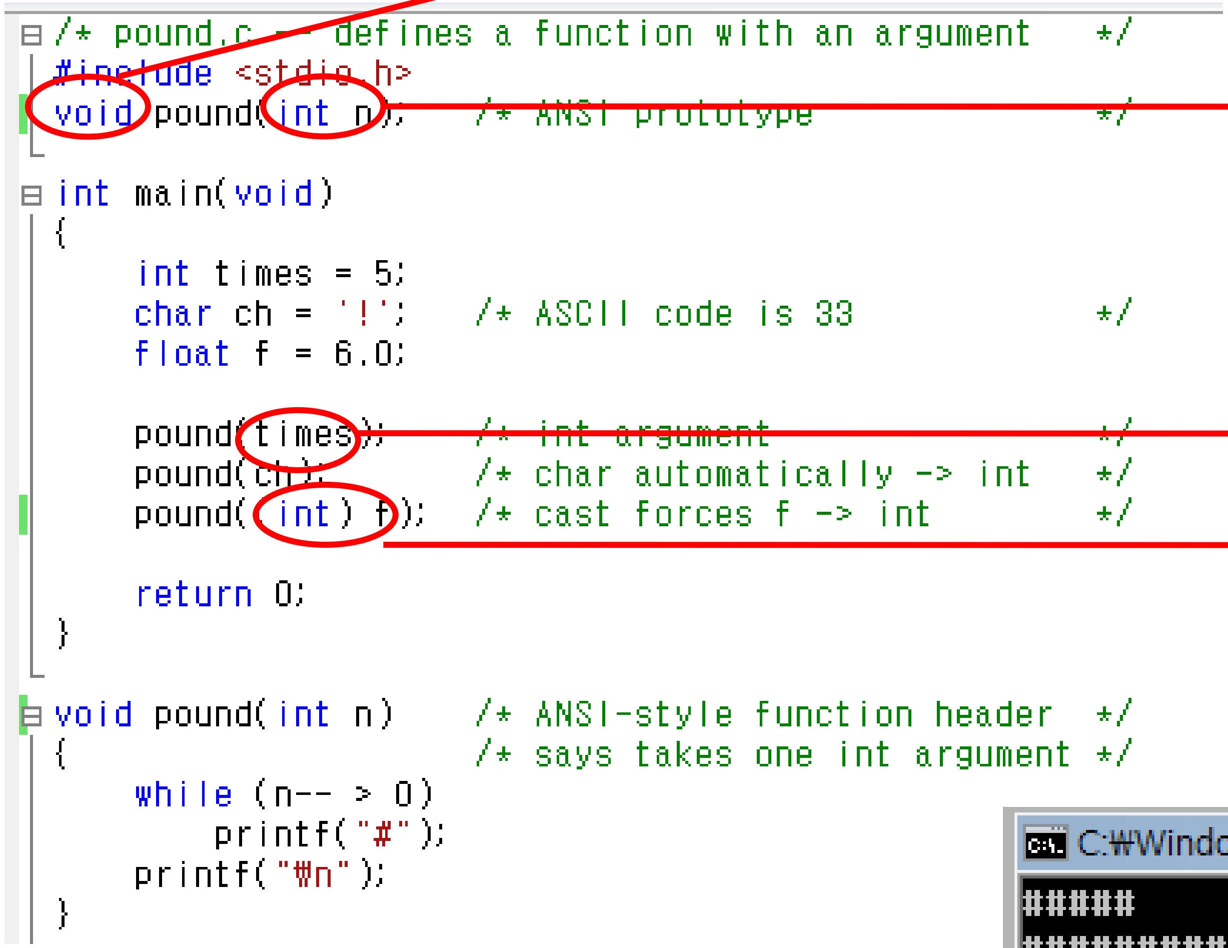
- You can demand the precise type of data conversion - *cast*.
- Precede the quantity with the name of the desired type in ().

- mice = 1.6 + 1.7;
- mice = (int) 1.6 + (int) 1.7;

Function with argument

SEOUL NATIONAL UNIVERSITY

No return value

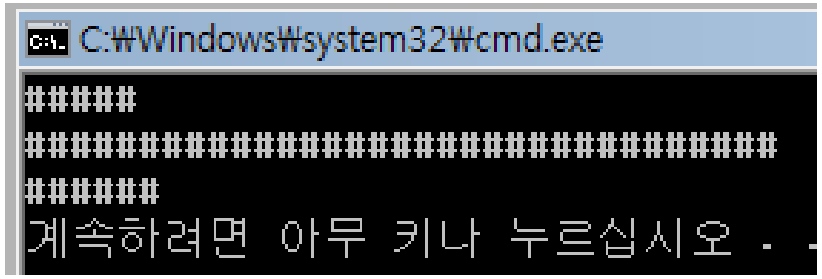


```
/* pound.c -- defines a function with an argument    */
#include <stdio.h>
void pound(int n);      /* ANSI prototype            */

int main(void)
{
    int times = 5;
    char ch = '!';      /* ASCII code is 33          */
    float f = 6.0;

    pound(times);       /* int argument              */
    pound(ch);          /* char automatically -> int */
    pound((int) f);     /* cast forces f -> int      */

    return 0;
}

void pound(int n)       /* ANSI-style function header */
{                       /* says takes one int argument */
    while (n-- > 0)
        printf("#");
    printf("\n");
}
```

argument (전달인자)
n: formal argument
(형식전달인자)

Times→5: actual
argument (실질전달인자)

Used type cast



C:\Windows\system32\cmd.exe

```
#####
#################################
######
계속하려면 아무 키나 누르십시오 . .
```

Summary

- while loop
- Operator (연산자):

  =   -   *   %   ++   --

  operator precedence (우선순위)
- Automatic type conversion, Type cast (데이터형 캐스트)
- Functions that uses arguments – void pound(n)

Next Lecture
Chapter 6. C primer Plus

SEOUL NATIONAL UNIVERSITY

- C control statements: Looping

- for
- While
- Do while

- Using return value