Fundamentals of Computer System
- C control statements: LOOPING

민기복

Ki-Bok Min, PhD

서울대학교 에너지자원공학과 조교수
Assistant Professor, Energy Resources Engineering

SEOUL NATIONAL UNIVERSITY

Mid-term exam

SEOUL NATIONAL UNIVERSITY

- 22 April 13:00 – 15:00
- Venue: 302-105 (제2공학관)
- Types of questions;
  - Explanation
  - Multiple choice
  - Short answer
  - Correction
  - Short programming

301
302 제2공학관
311
310
331
캠퍼스순환도로
중앙도서관
DRAG
CLOSE ×
302
제2공학관

제2공학관은 지상 10층으로 이루어진 건물이며, 화학생물공학부 사무실, BK21기계분야사업단연구실, BK21화공분야연구인력양성사업단연구실 등이 있다.

Last week

SEOUL NATIONAL UNIVERSITY

- Operator (연산자):

  = - * % ++ --

  operator precedence (우선순위)

- while loop

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

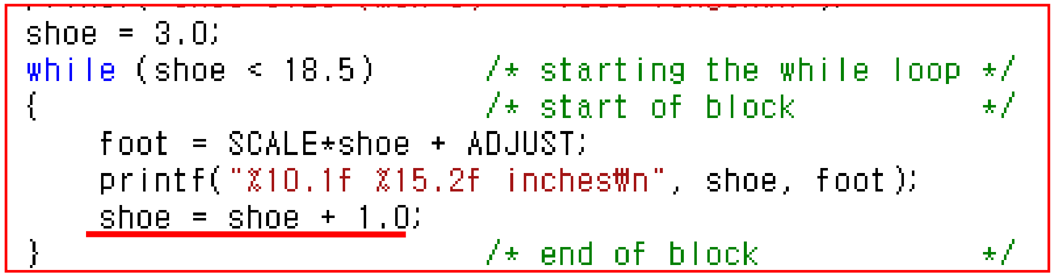
Operator (연산자)
++ -- (example)

SEOUL NATIONAL UNIVERSITY

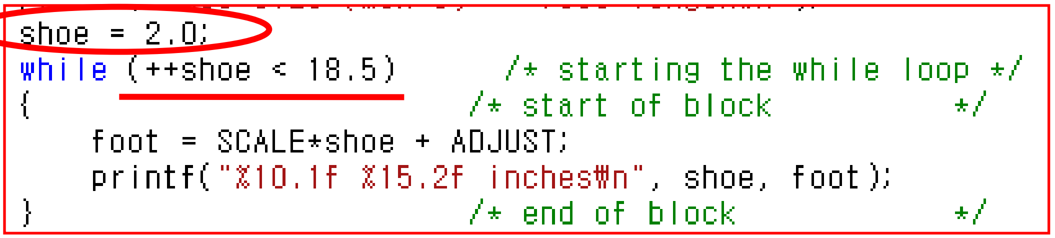
- Increment Operator(증가연산자): increases the value of its operand by 1.

  a++; → a = a + 1;

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



```
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 = 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            */
```

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 → 6 + (c = 3 + 8)

Type Conversion (형변환) & Cast operator (캐스트 연산자)

SEOUL NATIONAL UNIVERSITY

- If you mix types, C uses a set of rules to make type conversions automatically.
  - 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
- mice = 1.6 + 1.7;
- mice = (int) 1.6 + (int) 1.7;

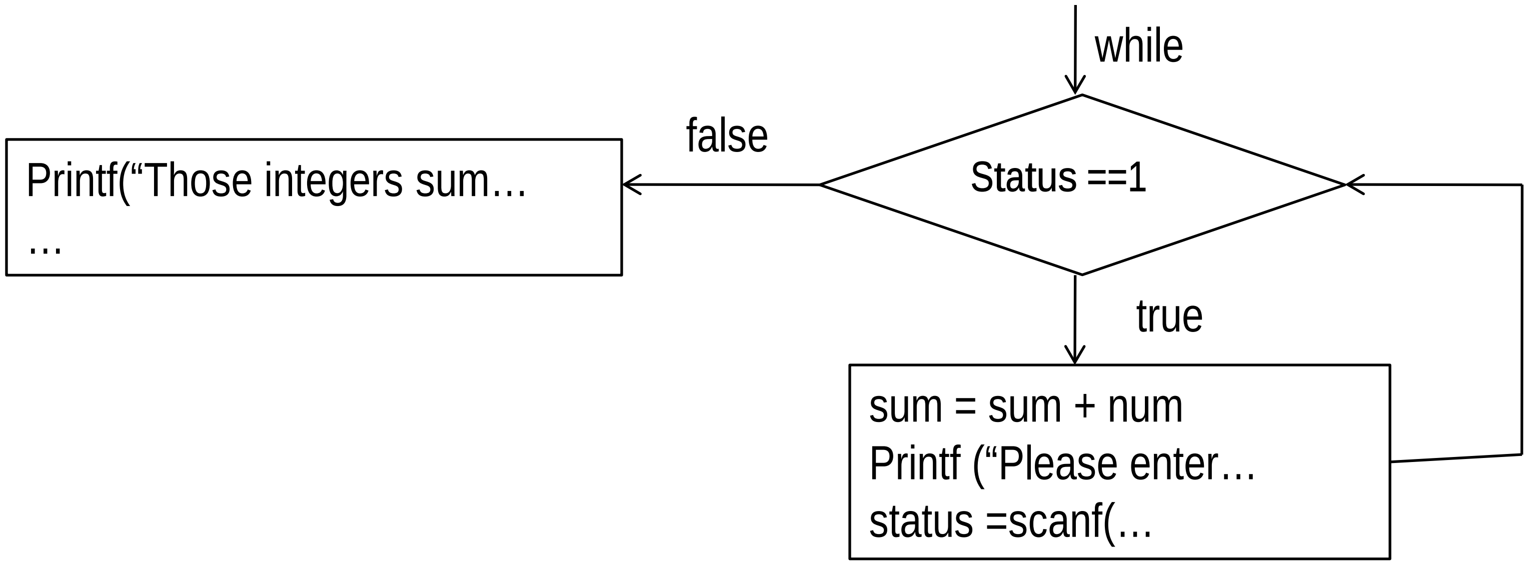
while statement
general form and structure

SEOUL NATIONAL UNIVERSITY

- General form

while (*expression*)

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



Today
Chapter 6. C primer Plus

SEOUL NATIONAL UNIVERSITY

- C control statements: Looping
  - for
  - while

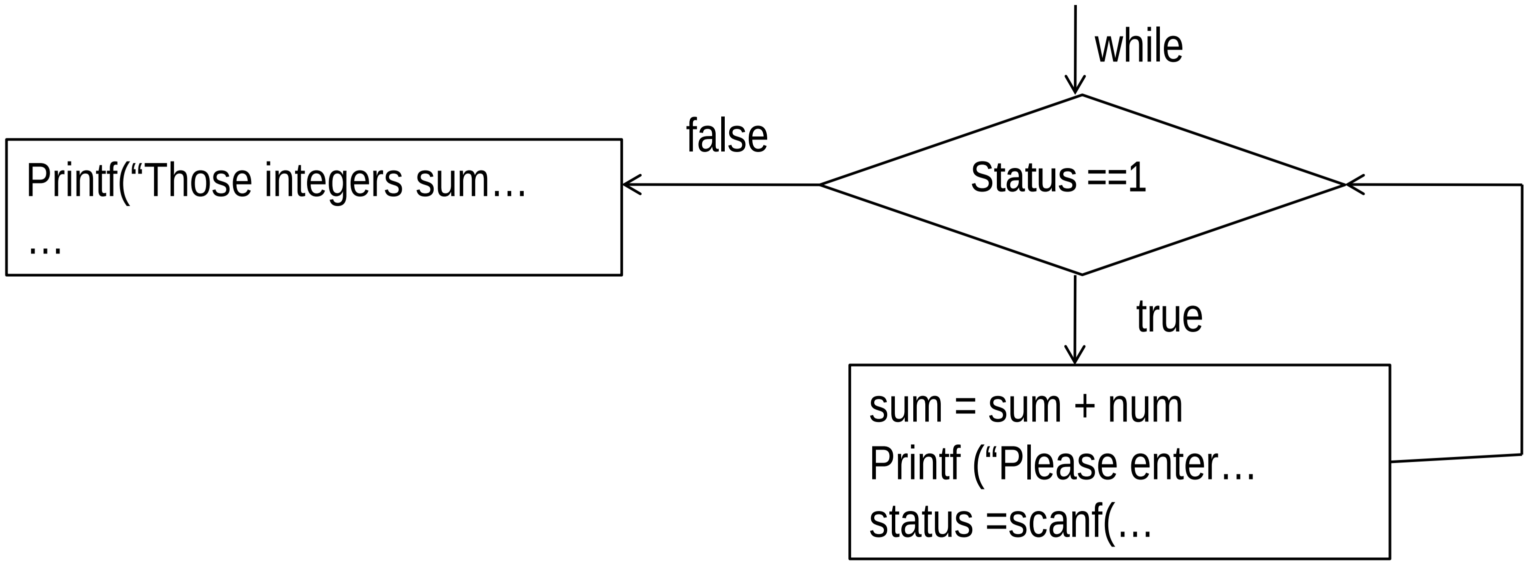
  } Entry-condition loop
  - do while —— Exit-condition loop
- What is true/nested loop
- Introduction to array
- Using a function return value

while statement
general form and structure

- General form

  while (*expression*)

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



Printf(“Those integers sum…
…

sum = sum + num
Printf (“Please enter…
status =scanf(…

while Loop
terminating

SEOUL NATIONAL UNIVERSITY

```
index = 1;
while (index < 5)
printf(“Hello, world!\n”);
```

```
index = 1;
while (--index < 5)
printf(“Hello, world!\n”);
```

- We need the value to make the expression false to escape the loop

while Loop
An entry-condition Loop

SEOUL NATIONAL UNIVERSITY

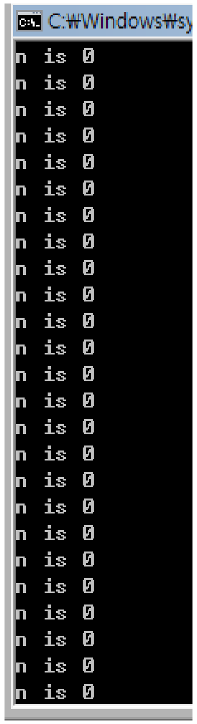
```
index = 10;
while (index++ < 5)
        printf(“Hello, world!\n”);
```

- To execute the loop;

```
index = 3;
```

while Loop
Infinite Loop

```
/* while1.c -- watch your braces         */
/* bad coding creates an infinite loop */
#include <stdio.h>
int main(void)
{
     int n = 0;

     while (n < 3)
        {printf("n is %d₩n", n);
         n++;}
     printf("That's all this program does₩n");

     return 0;
}
```

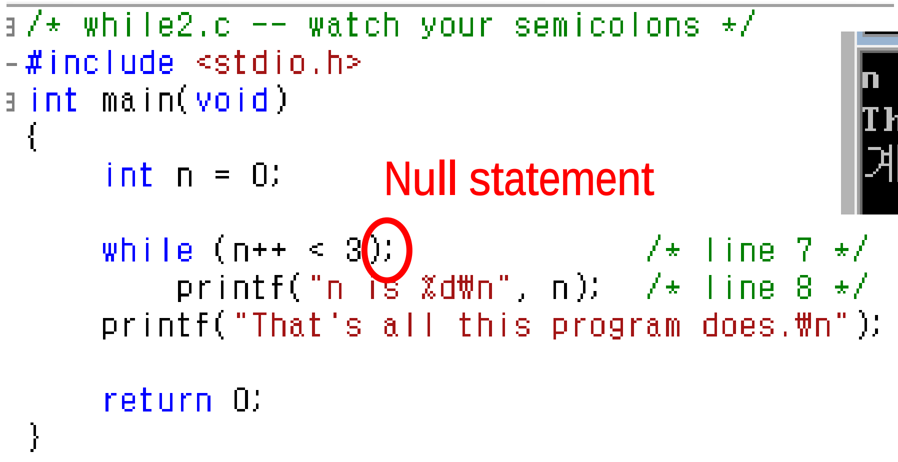


```
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
```

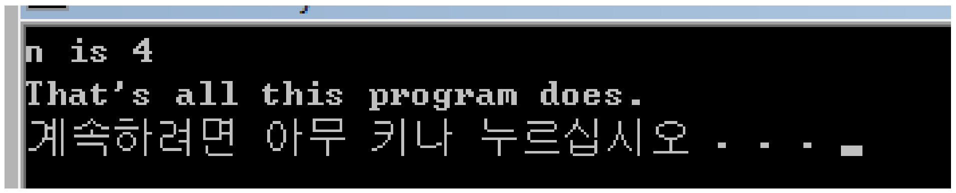
- Only the single statement (simple or compound) is part of the loop
- The statements runs from while to ; or } (compound statement)

while Loop
null statement

- Remembering that while loop ends with first ; or } (compound), what would be the output of the following program.



```
/* while2.c -- watch your semicolons */
#include <stdio.h>
int main(void)
{
    int n = 0;

    while (n++ < 3);              /* line 7 */
        printf("n is %d₩n", n);   /* line 8 */
    printf("That's all this program does.₩n");

    return 0;
}
```



```
n is 4
That's all this program does.
계속하려면 아무 키나 누르십시오 . . .
```

- Null statement does nothing but **while** loop ends there.

while loop
relational expression/operator

SEOUL NATIONAL UNIVERSITY

- Relational expressions (관계수식): make comparisons
- Relational operator: appear in relational expressions

Characters can be also used

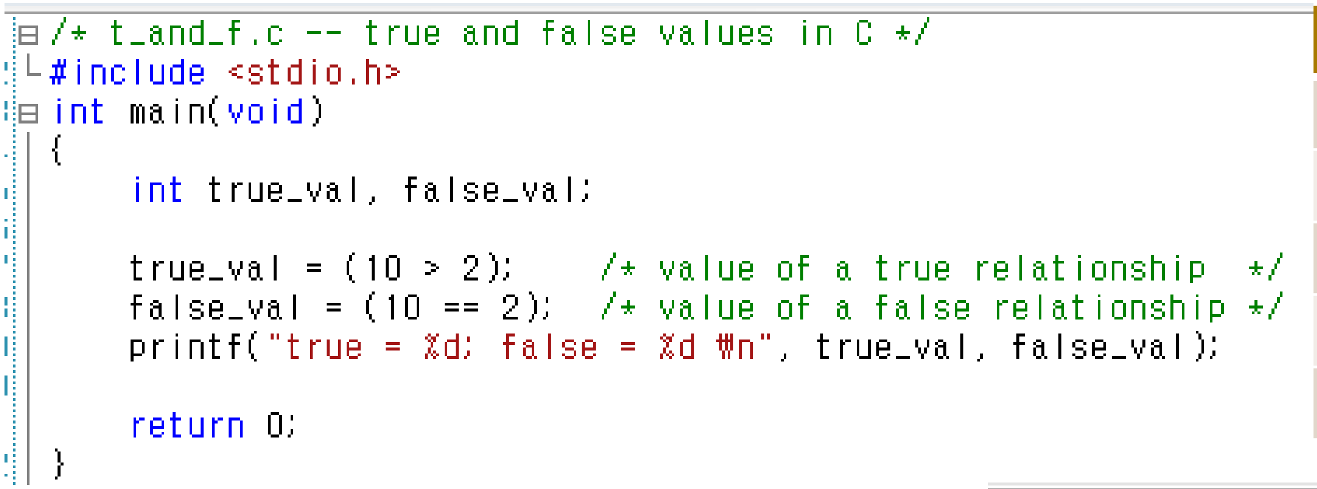
```
while(ch != '$')
{
    count++;
    scanf("%c",&ch);
}
```

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

while loop
what is truth?

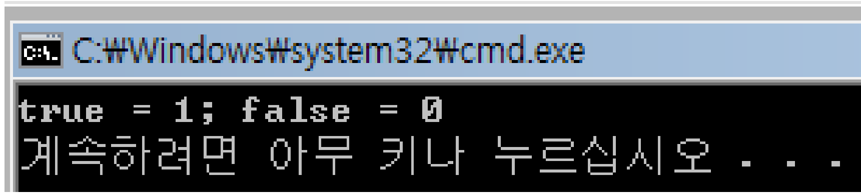
SEOUL NATIONAL UNIVERSITY

- Recall that an expression in C always has a value.



```c
/* t_and_f.c -- true and false values in C */
#include <stdio.h>
int main(void)
{
        int true_val, false_val;

        true_val = (10 > 2);      /* value of a true relationship  */
        false_val = (10 == 2);    /* value of a false relationship */
        printf("true = %d; false = %d \n", true_val, false_val);

        return 0;
}
```

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



C:₩Windows₩system32₩cmd.exe

```
true = 1; false = 0
계속하려면 아무 키나 누르십시오 . . .
```

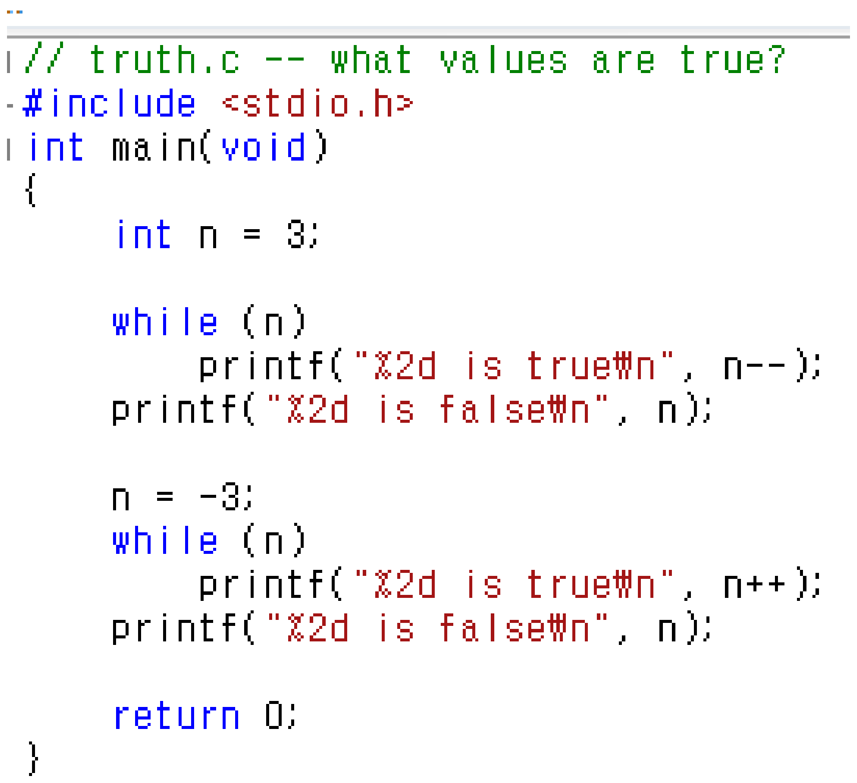
- An infinite while loop

While (1)

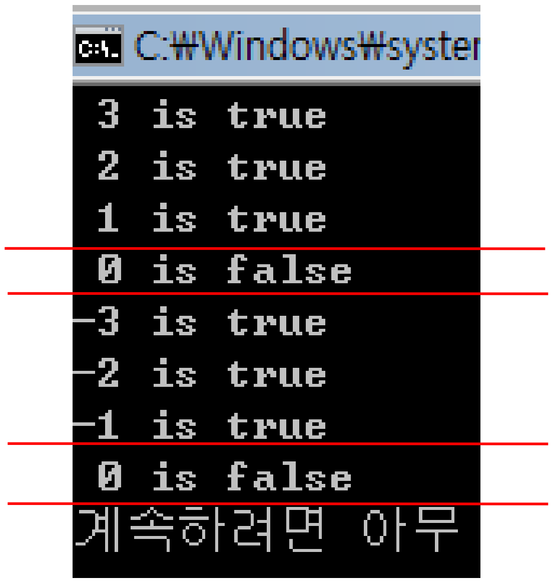
{ … }

while statement
what else is true?

- True: All nonzero values, -1, 5, 1000
- False: 0



```c
// truth.c -- what values are true?
#include <stdio.h>
int main(void)
{
    int n = 3;

    while (n)
        printf("%2d is true₩n", n--);
    printf("%2d is false₩n", n);

    n = -3;
    while (n)
        printf("%2d is true₩n", n++);
    printf("%2d is false₩n", n);

    return 0;
}
```



C:₩Windows₩syster

```
 3 is true
 2 is true
 1 is true
 0 is false
-3 is true
-2 is true
-1 is true
 0 is false
계속하려면 아무
```

while loop
what else is true?

SEOUL NATIONAL UNIVERSITY

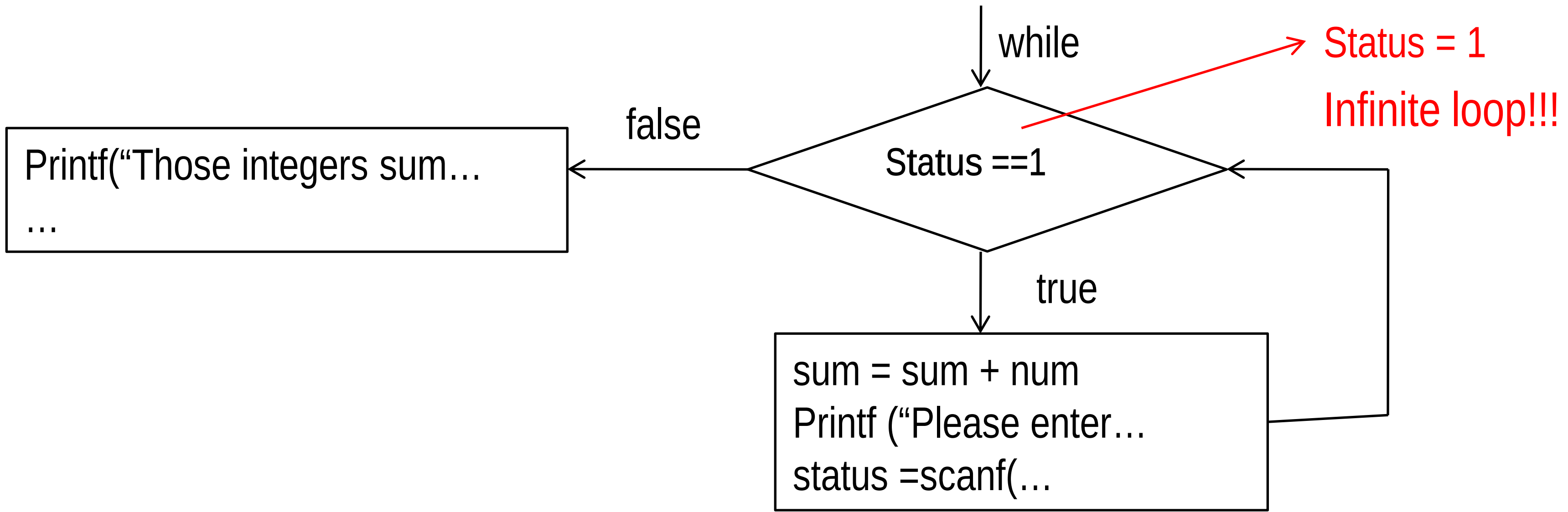
while(n != 0)

while(n)

the same!

while (*expression*)

statement → One statement with ; or a block with {}



while loop
tip to avoid errors

SEOUL NATIONAL UNIVERSITY

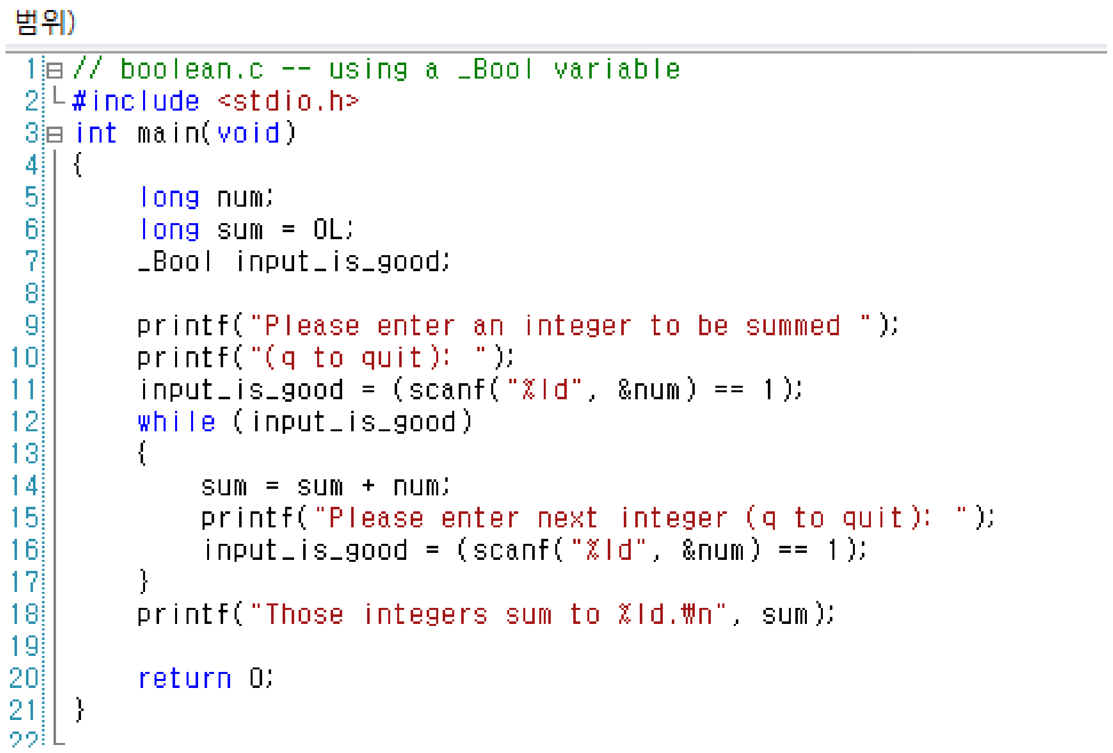
• n = 5;	assigns the value 5 to n.
• n == 5;	check to see whether n has the value 5.
• 5 = n;	syntax error.
• 5 == n;	check to see whether n has the value 5.

Can avoid unwanted errors by putting the constant in the left.

while loop _Bool

SEOUL NATIONAL UNIVERSITY

- _Bool: variables representing true (1) or false (0)
- 1 bit variables → save memory



범위)

```
// boolean.c -- using a _Bool variable
#include <stdio.h>
int main(void)
{
    long num;
    long sum = 0L;
    _Bool input_is_good;

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

    return 0;
}
```

while loop
precedence of relational operators

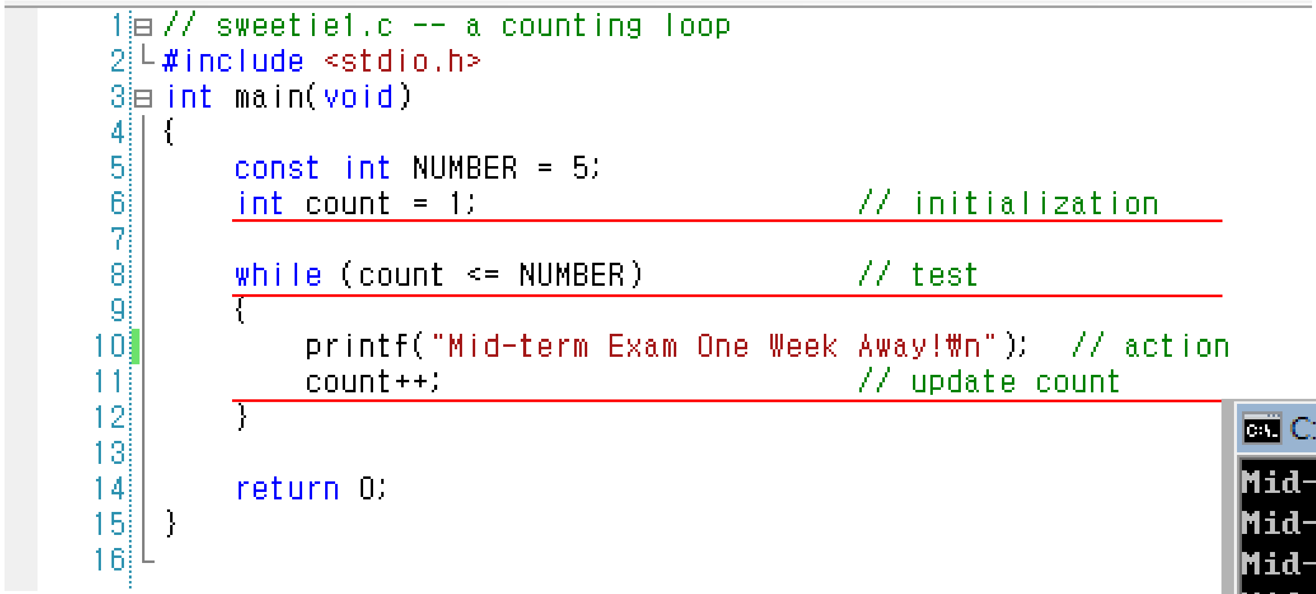
SEOUL NATIONAL UNIVERSITY

operator	Associativity
()	→
+ - ++ -- sizeof	←
* /	→
+ - (binary)	→
< > <= >=	→
== !=	→
=	←

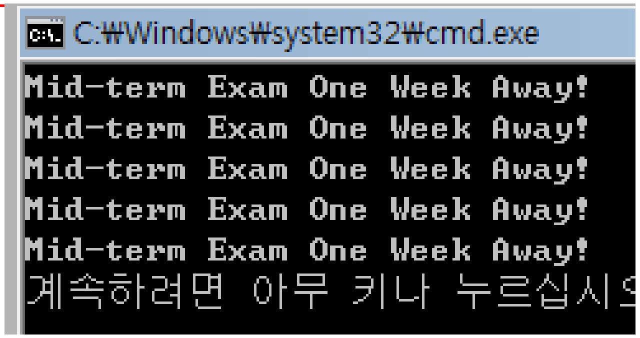
- x > y + 2 ↔ x > (y+2)
- x = y > 2 ↔ x = (y > 2)
- ex != wye == zee ↔ (ex != wye) == zee

for loop
Limitation of while

SEOUL NATIONAL UNIVERSITY



```c
// sweetie1.c -- a counting loop
#include <stdio.h>
int main(void)
{
    const int NUMBER = 5;
    int count = 1;                      // initialization

    while (count <= NUMBER)             // test
    {
        printf("Mid-term Exam One Week Away!₩n");  // action
        count++;                        // update count
    }

    return 0;
}
```

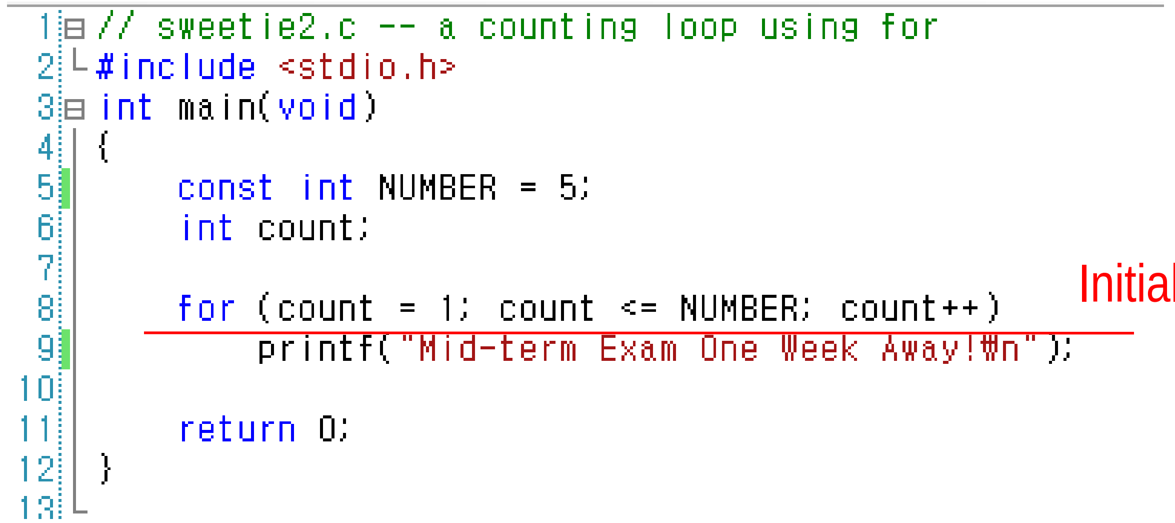


```
C:₩Windows₩system32₩cmd.exe
Mid-term Exam One Week Away!
Mid-term Exam One Week Away!
Mid-term Exam One Week Away!
Mid-term Exam One Week Away!
Mid-term Exam One Week Away!
계속하려면 아무 키나 누르십시
```

- Three actions are involved;
  1. Initialization,
  2. Comparison
  3. The counter is incremented

$\left.\begin{array}{l} \\ \\ \end{array}\right\}$ Can be combined

for loop
Form of for loop



```c
// sweetie2.c -- a counting loop using for
#include <stdio.h>
int main(void)
{
    const int NUMBER = 5;
    int count;

    for (count = 1; count <= NUMBER; count++)
        printf("Mid-term Exam One Week Away!\n");

    return 0;
}
```

Initializing, testing, updating

for (initialize; test; update)

statement

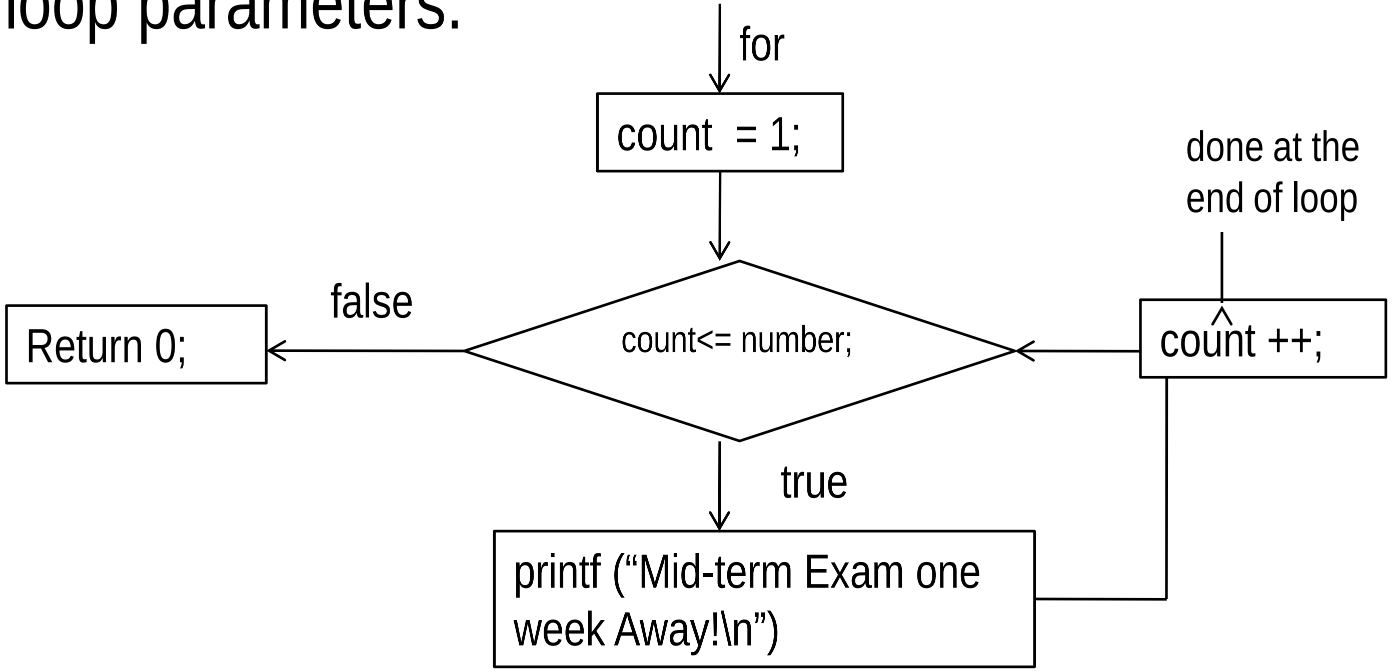
- Loop continues until test becomes 0 or false

for loop
Form of for loop

SEOUL NATIONAL UNIVERSITY

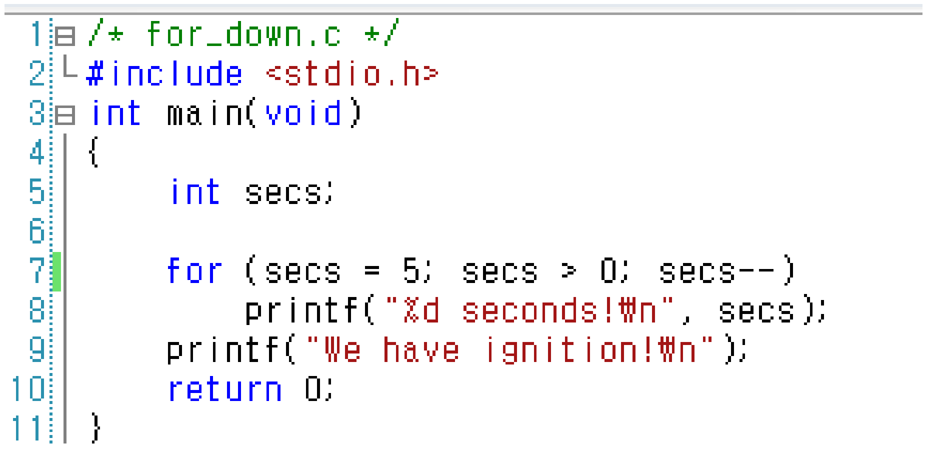
```
for (initializing; test; upgrade)
    statement  ------------------------------>  One statement with ; or
                                                a block with {}
```

- The first line of for loop tells us immediately all the information about the loop parameters.

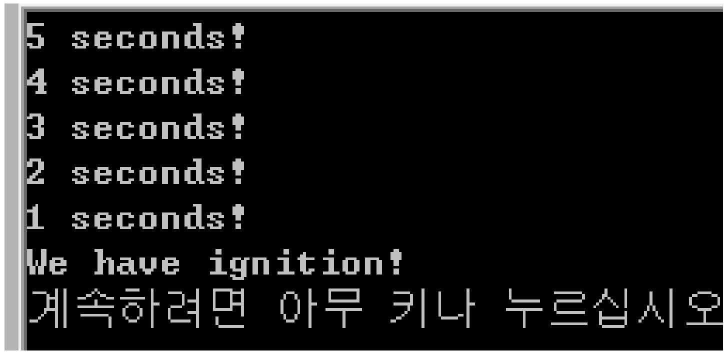


for loop
Flexibility

SEOUL NATIONAL UNIVERSITY



```c
/* for_down.c */
#include <stdio.h>
int main(void)
{
	int secs;

	for (secs = 5; secs > 0; secs--)
		printf("%d seconds!₩n", secs);
	printf("We have ignition!₩n");
	return 0;
}
```



```
5 seconds!
4 seconds!
3 seconds!
2 seconds!
1 seconds!
We have ignition!
계속하려면 아무 키나 누르십시오
```

- Use decrement operator to count down
- --secs ???

for loop
Flexibility

- Count by twos, tens, etc.

```
for (n = 2; n <60; n = n + 13)
```

- Count by characters instead of by numbers

```
for (ch = ‘a’; ch <= ‘z’; ch++)
```

- Test some condition other than the number of iterations

```
for (num = 1; num*num*num <=216; num++)
```

- Let the quantity increase geometrically(기하급수적으로) instead of arithmetically

```
for (debt = 100.0;debt < 150.0; debt = debt *1.1)
```

for loop
Flexibility

SEOUL NATIONAL UNIVERSITY

- Use any legal expression for the third expression.

```
for (x = 1; y <=75; y = (++x *5) + 50)
```

- Leave one or more expression blank

```
for (n = 3; ans <= 25;  )
```

  - But in this case, you need some statement to finish the loop

- The first expression need not initialize a variable. It could be something like **printf()**

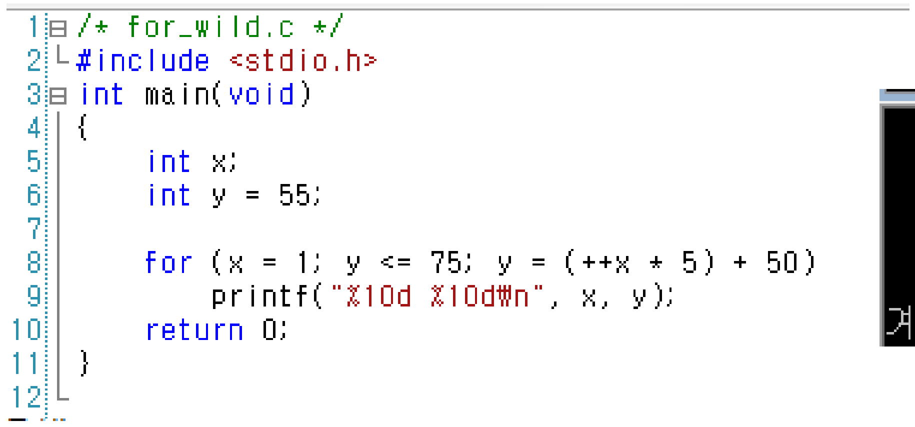
```
for (printf("Keep entering!\n");   num!=1;   )
```

- The parameters of the loop expressions can be altered by actions within the loop

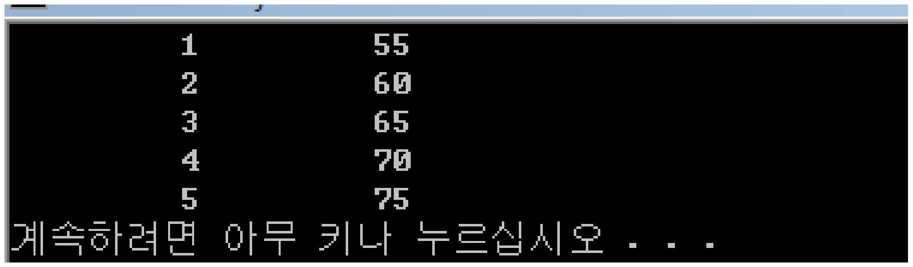
```
for (n = 1; n < 10000; n = n + delta)
```

for loop
Flexibility – an example

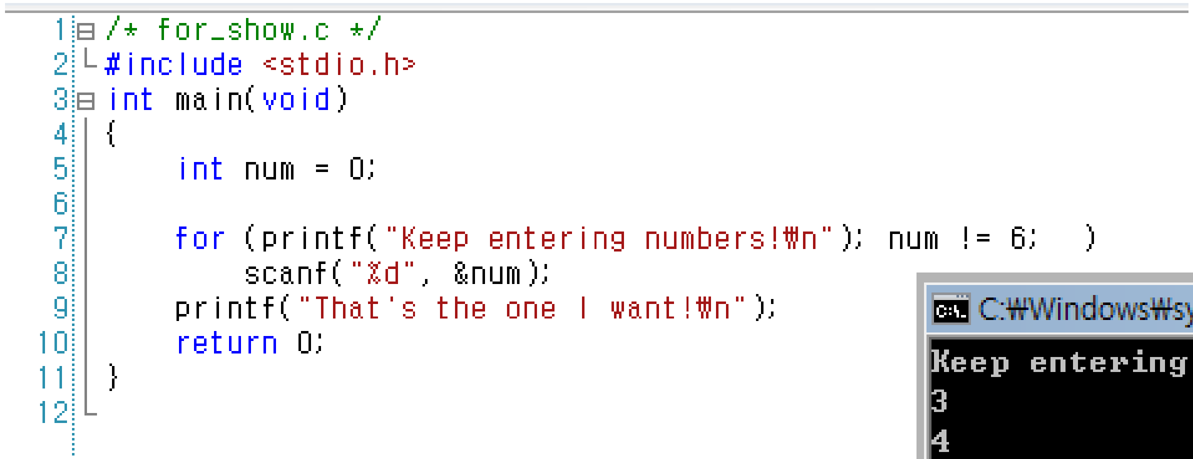
SEOUL NATIONAL UNIVERSITY



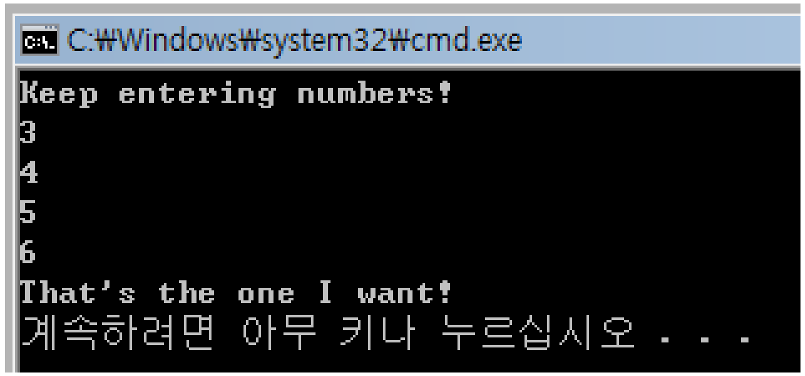
```c
/* for_wild.c */
#include <stdio.h>
int main(void)
{
    int x;
    int y = 55;

    for (x = 1; y <= 75; y = (++x * 5) + 50)
        printf("%10d %10d₩n", x, y);
    return 0;
}
```



```
          1          55
          2          60
          3          65
          4          70
          5          75
계속하려면 아무 키나 누르십시오 . . .
```



```c
/* for_show.c */
#include <stdio.h>
int main(void)
{
    int num = 0;

    for (printf("Keep entering numbers!₩n"); num != 6;   )
        scanf("%d", &num);
    printf("That's the one I want!₩n");
    return 0;
}
```



C:₩Windows₩system32₩cmd.exe

```
Keep entering numbers!
3
4
5
6
That's the one I want!
계속하려면 아무 키나 누르십시오 . . .
```

More assignment operators

+=   -=   *=   /=   %=

SEOUL NATIONAL UNIVERSITY

scores += 20	↔	scores = score + 20
dimes -= 2	↔	dimes = dimes +2
bunnies *= 2	↔	bunnies = bunnies *2
time /= 2.73	↔	time = time / 2.73
reduce %= 3	↔	reduce = reduce % 3
x *= 3 * y + 12	↔	x = x * (3 * y + 12)

These assignment operators has low priorities as =

More assignment operators
The comma operator

SEOUL NATIONAL UNIVERSITY

- Can include more than one initialization or update expression

  for (ounces= 1, cost=FIRST_OZ; ounces <=16; ounces++, cost+ = NEXT_OZ)

- Expressions are evaluated → (left to right)
  - Ex) ounces++, cost = ounces * FIRST_OZ

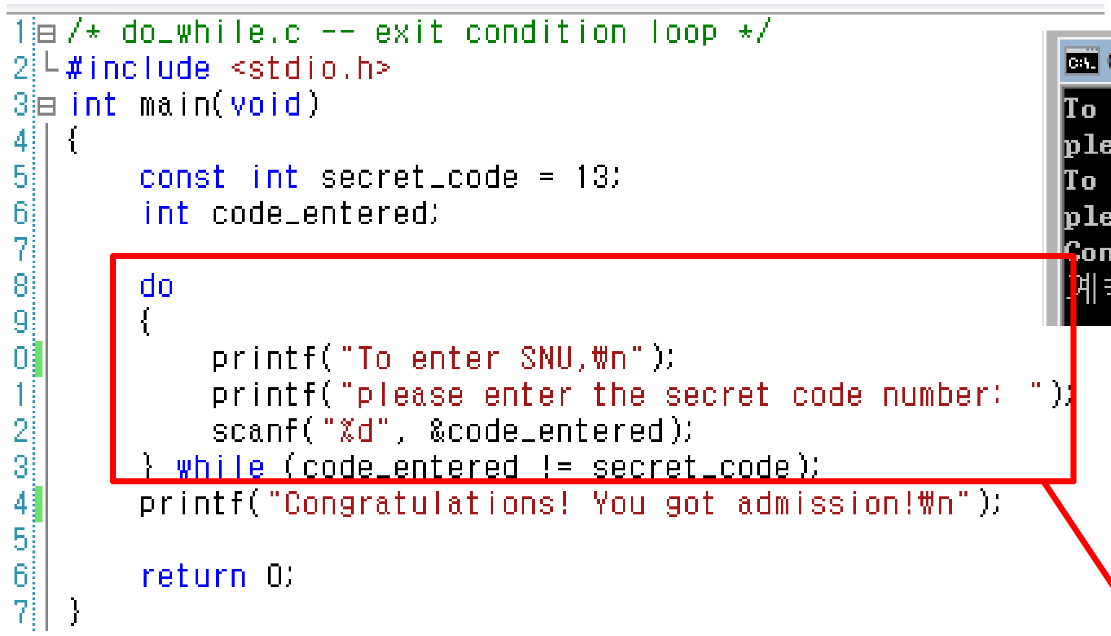
do while
An exit-condition loop

SEOUL NATIONAL UNIVERSITY

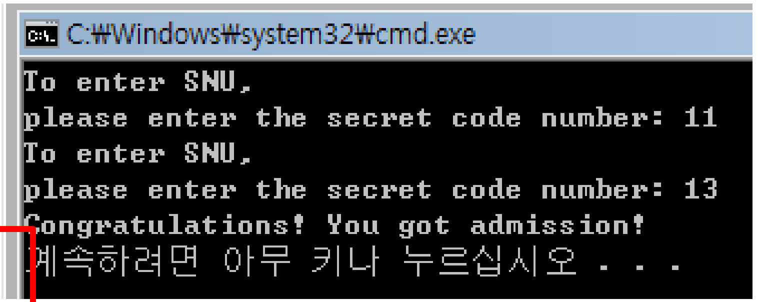
- while loop & for loop : *entry-condition* loop
  - Test is checked before each iteration
  - The statement in the loop may not execute
- do while loop: *exit-condition* loop
  - The statements are executed at least once

do while
An exit-condition loop

SEOUL NATIONAL UNIVERSITY



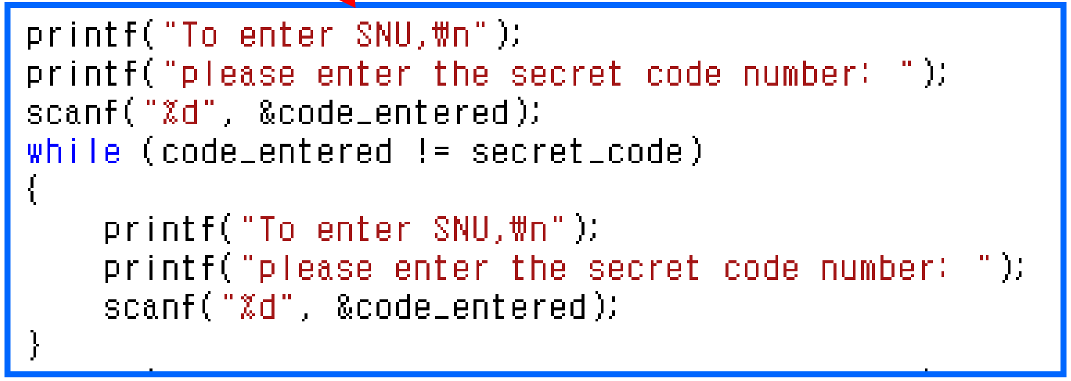
```c
/* do_while.c -- exit condition loop */
#include <stdio.h>
int main(void)
{
    const int secret_code = 13;
    int code_entered;

    do
    {
        printf("To enter SNU,\n");
        printf("please enter the secret code number: ");
        scanf("%d", &code_entered);
    } while (code_entered != secret_code);
    printf("Congratulations! You got admission!\n");

    return 0;
}
```



C:₩Windows₩system32₩cmd.exe

```
To enter SNU,
please enter the secret code number: 11
To enter SNU,
please enter the secret code number: 13
Congratulations! You got admission!
계속하려면 아무 키나 누르십시오 . . .
```

while loop - a little longer



```c
printf("To enter SNU,\n");
printf("please enter the secret code number: ");
scanf("%d", &code_entered);
while (code_entered != secret_code)
{
    printf("To enter SNU,\n");
    printf("please enter the secret code number: ");
    scanf("%d", &code_entered);
}
```

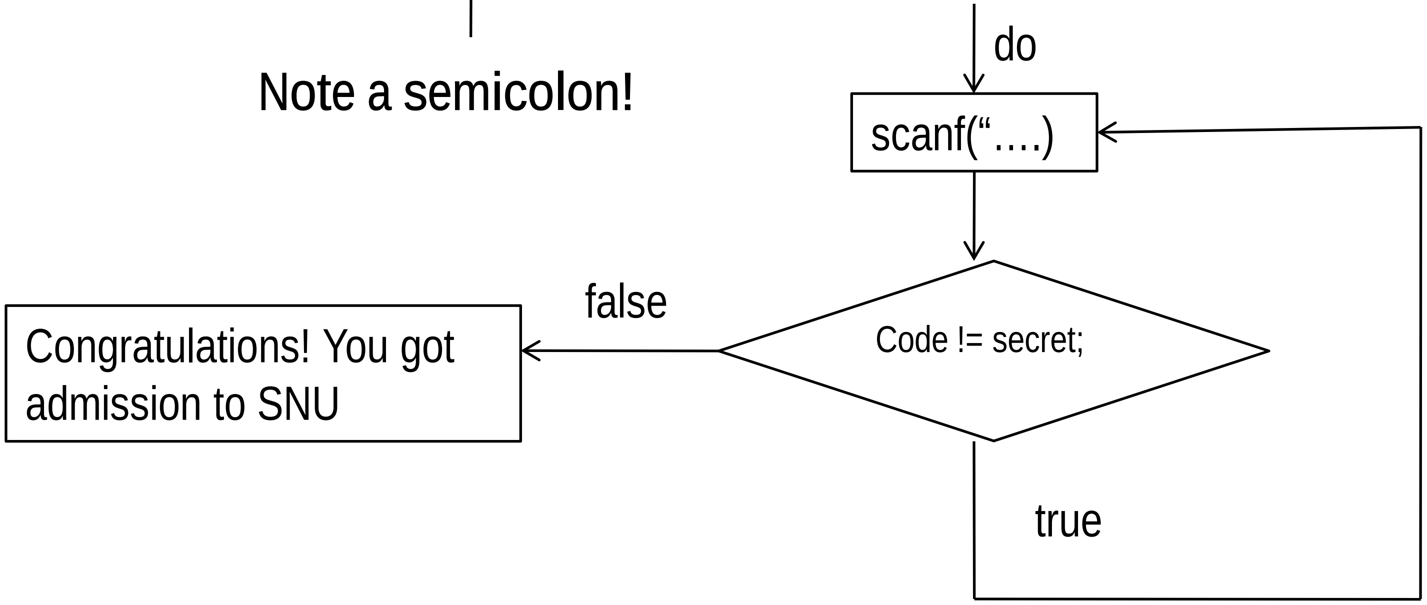
do while
Form of do while loop

SEOUL NATIONAL UNIVERSITY

```
do
    statement
while (expression);
```

One statement with ; or
a block with {}

Note a semicolon!



Which loop?
while, for, do while

SEOUL NATIONAL UNIVERSITY

- In general, entry-condition loop (while, for) better than exit-condition loop (do while);
  - Better to look before you leap
  - Easier to read a program when a test is in the beginning
  - In many cases, it is important that the loop be skipped entirely if the test is not initially met

Which loop?
while versus for

SEOUL NATIONAL UNIVERSITY

```
for ( ; test ; )          ↔     while (test)
```

```
initialize;
while(test)
{
        body;
        update;
}
```

↔

```
for (initialize; test; update)
        body;
```

- Initializing & updating → for
- Other than this → while

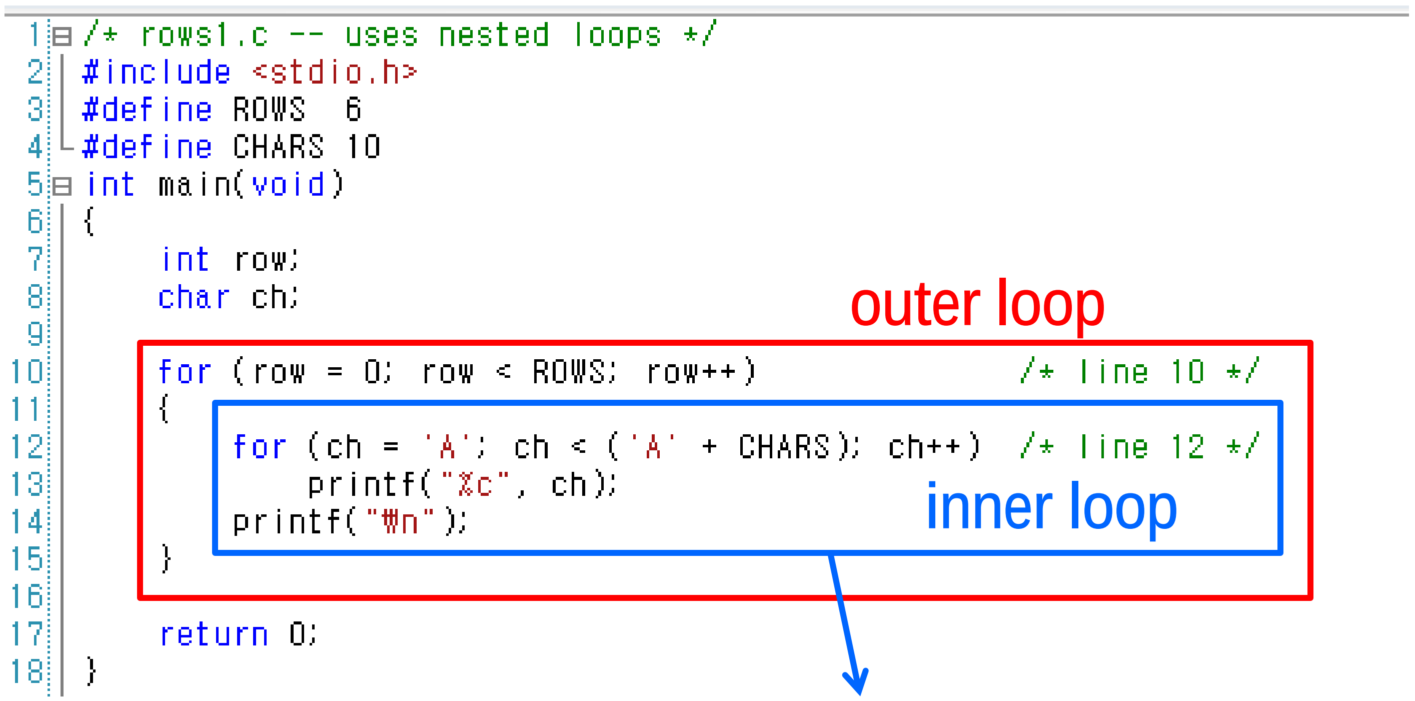
Ex)

```
while (scanf("%ld", &num)==1)

For(count = 1; count<=100;count++)
```

Nested loop (중첩루프)

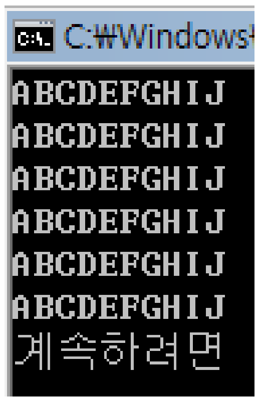
SEOUL NATIONAL UNIVERSITY

- Nested loop: one loop inside another loop
- Useful for many cases; e.g.) data in rows and columns



```
 1  /* rows1.c -- uses nested loops */
 2  #include <stdio.h>
 3  #define ROWS   6
 4  #define CHARS 10
 5  int main(void)
 6  {
 7      int row;
 8      char ch;
 9  
10      for (row = 0; row < ROWS; row++)              /* line 10 */
11      {
12          for (ch = 'A'; ch < ('A' + CHARS); ch++)  /* line 12 */
13              printf("%c", ch);
14          printf("\n");
15      }
16  
17      return 0;
18  }
```

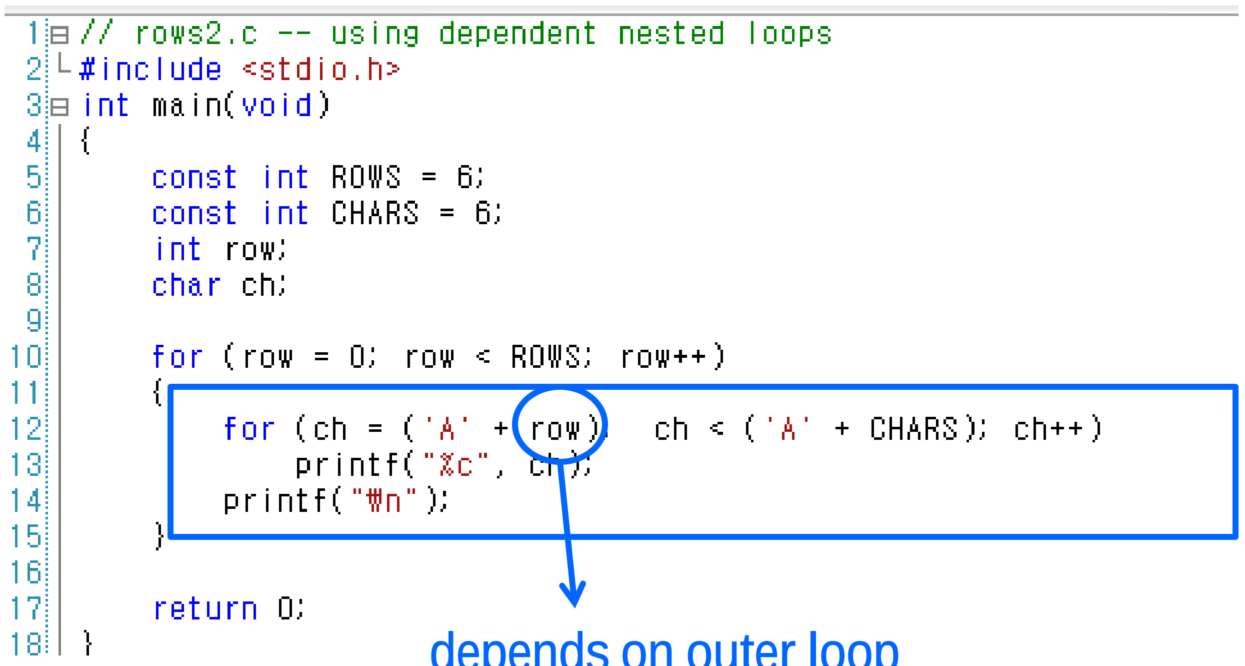
Run 10 times for each iteration of outer loop



```
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
계속하려면
```

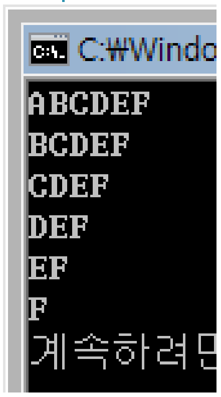
A nested variation

SEOUL NATIONAL UNIVERSITY



```
// rows2.c -- using dependent nested loops
#include <stdio.h>
int main(void)
{
        const int ROWS = 6;
        const int CHARS = 6;
        int row;
        char ch;

        for (row = 0; row < ROWS; row++)
        {
            for (ch = ('A' + row);  ch < ('A' + CHARS); ch++)
                printf("%c", ch);
            printf("\n");
        }

        return 0;
}
```

depends on outer loop



C:\Windo

```
ABCDEF
BCDEF
CDEF
DEF
EF
F
계속하려면
```

Introducing arrays
- very brief introduction

SEOUL NATIONAL UNIVERSITY

- Arrays (배열): important! & Useful!
- Array: a series of values of the same type stored sequentially. The whole arrays bears a single name.
- 배열: 동일한 데이터 형을 가진 여러 값들이 연속적으로 저장되어 있는 것. 배열 전체가 하나의 이름 사용.

Index, subscript(첨자) or offset

- int score[10];
- score is an array with 10 elements. Each of element can hold a type int value

SEOUL NATIONAL UNIVERSITY

Introducing arrays
- very brief introduction

int score[10]

72	75	80	25	120	1685	0	-56	2567	23
score[0]	score[1]	score[2]	score[3]	score[4]	score[5]	Score[6]	score[7]	score[8]	score[9]

- Numbering starts from 0 (not 1!!!).
- Each element can be assigned a **int** value.

  score[4] = 120;   score[9]=23;

- 배열원소를 같은 데이터형의 일반 변수를 사용하는 것과 동일한 방식으로 사용가능

  scanf("%d", &score[4]);

- C doesn't check whether you use a correct index.

  score[10] = 15; score[23]=253; $\rightarrow$ wrong but compiler does not detect this.

Introducing arrays
- very brief introduction

- An array can be of any data type.
  - float nannies[22]; /* holds 22 floating numbers */
  - Char alpha[26]; /* holds 26 characters */
  - Long big[500]; /* hold 500 long integers */
- Strings are a special case of char array.
  - String < char array

y	o	u		d	o		i	t	.

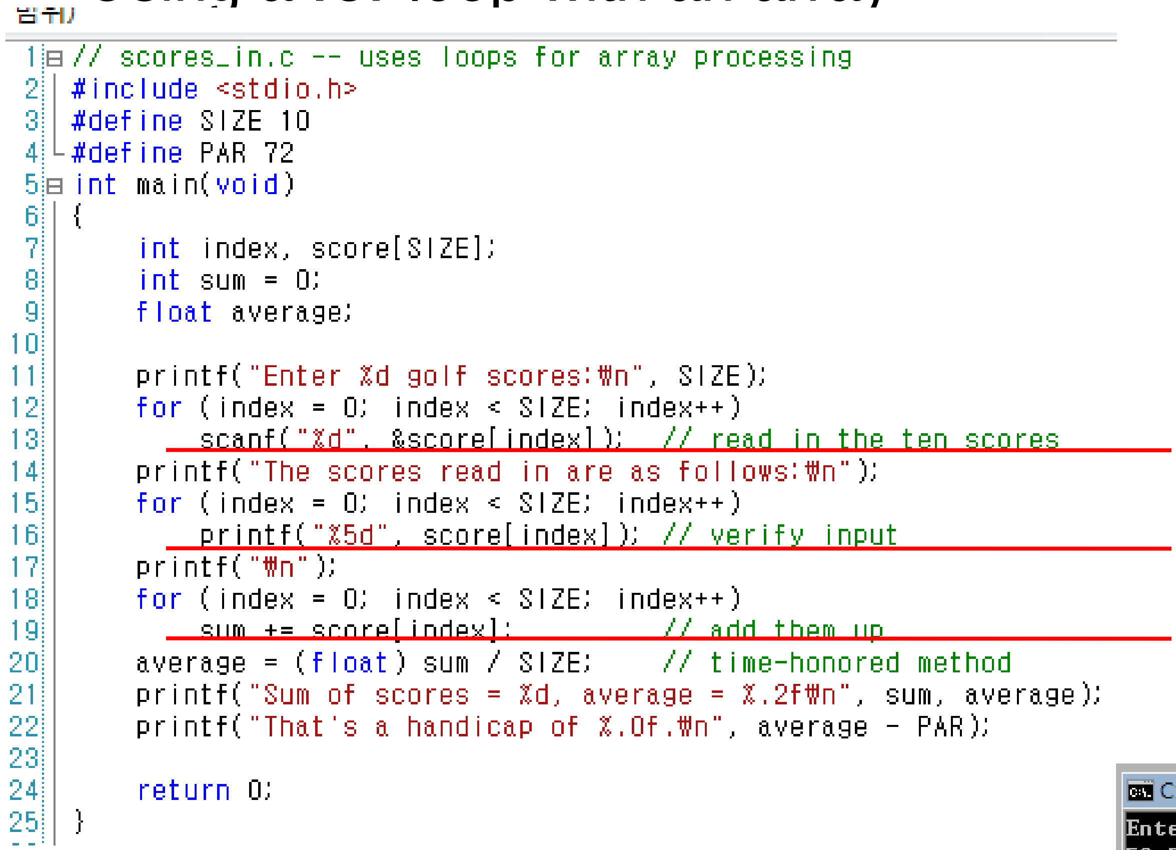
Char array(O), string (X)

y	o	u		d	o		i	t	.	\0

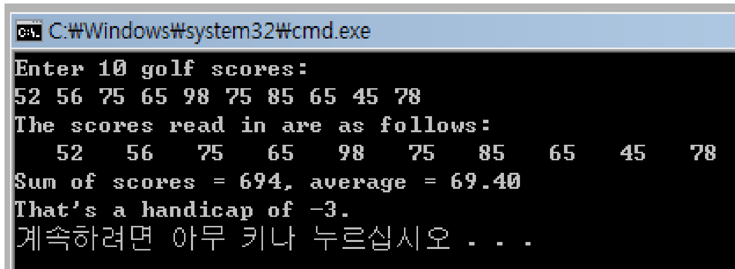
Char array(O), string (O)

Introducing arrays
- very brief introduction

- Using a for loop with an array



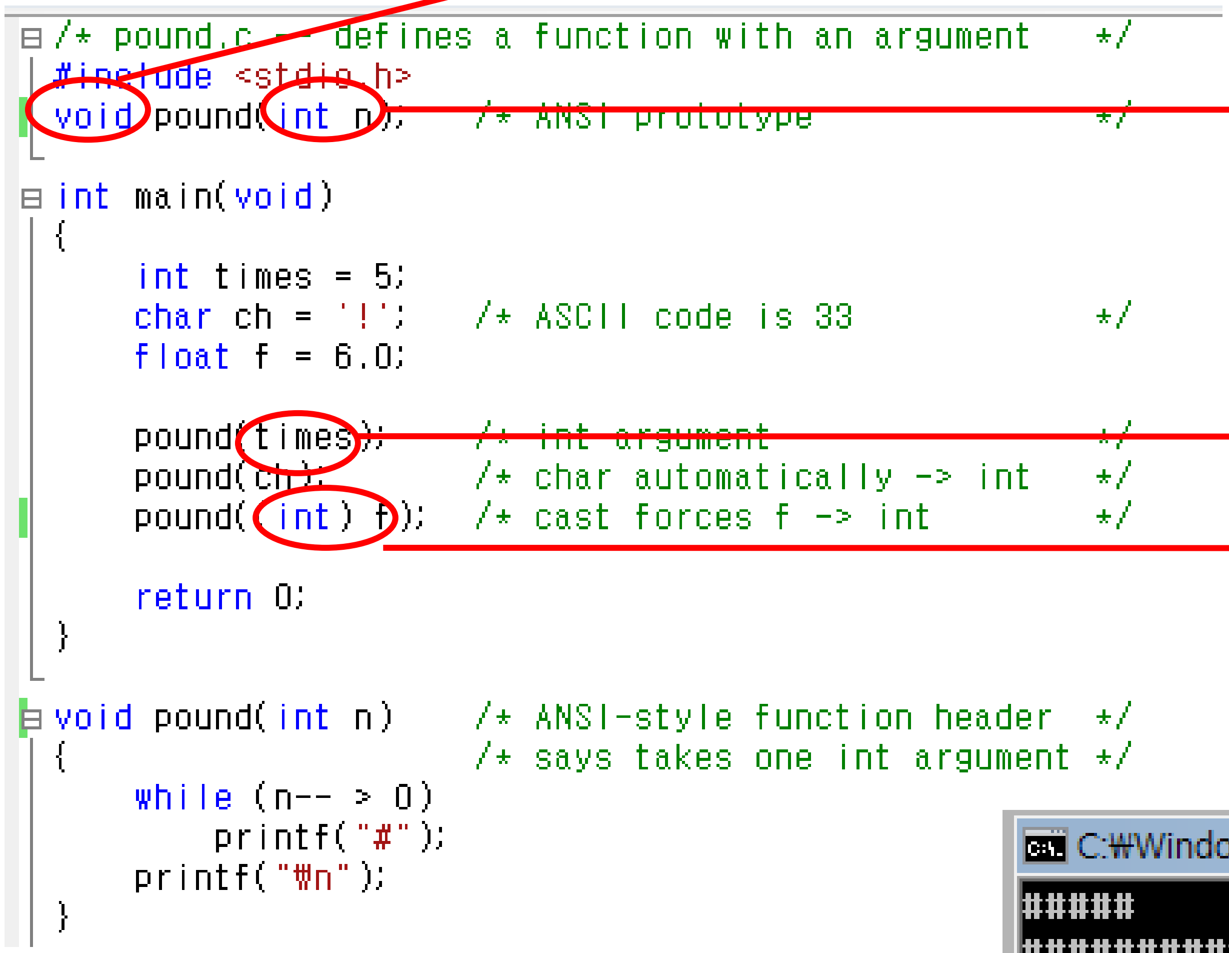
```c
// scores_in.c -- uses loops for array processing
#include <stdio.h>
#define SIZE 10
#define PAR 72
int main(void)
{
	int index, score[SIZE];
	int sum = 0;
	float average;

	printf("Enter %d golf scores:₩n", SIZE);
	for (index = 0; index < SIZE; index++)
		scanf("%d", &score[index]);  // read in the ten scores
	printf("The scores read in are as follows:₩n");
	for (index = 0; index < SIZE; index++)
		printf("%5d", score[index]); // verify input
	printf("₩n");
	for (index = 0; index < SIZE; index++)
		sum += score[index];         // add them up
	average = (float) sum / SIZE;    // time-honored method
	printf("Sum of scores = %d, average = %.2f₩n", sum, average);
	printf("That's a handicap of %.0f.₩n", average - PAR);

	return 0;
}
```



```
Enter 10 golf scores:
52 56 75 65 98 75 85 65 45 78
The scores read in are as follows:
   52   56   75   65   98   75   85   65   45   78
Sum of scores = 694, average = 69.40
That's a handicap of -3.
계속하려면 아무 키나 누르십시오 . . .
```

Function with argument (last week)

SEOUL NATIONAL UNIVERSITY



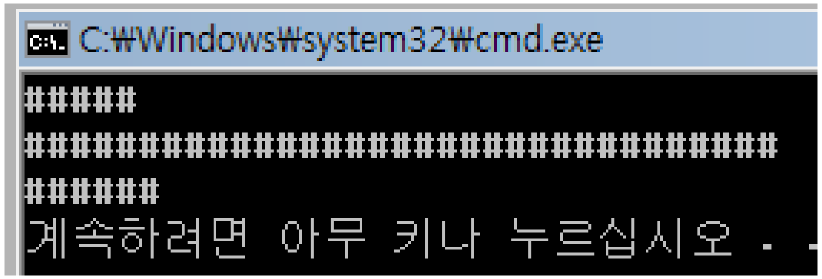
```
/* pound.c -- defines a function with an argument   */
#include <stdio.h>
void pound(int n);     /* ANSI prototype              */

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");
}
```

No return value

argument (전달인자)
n: formal argument
(형식전달인자)

Times→5: actual
argument (실질전달인자)

Used type cast

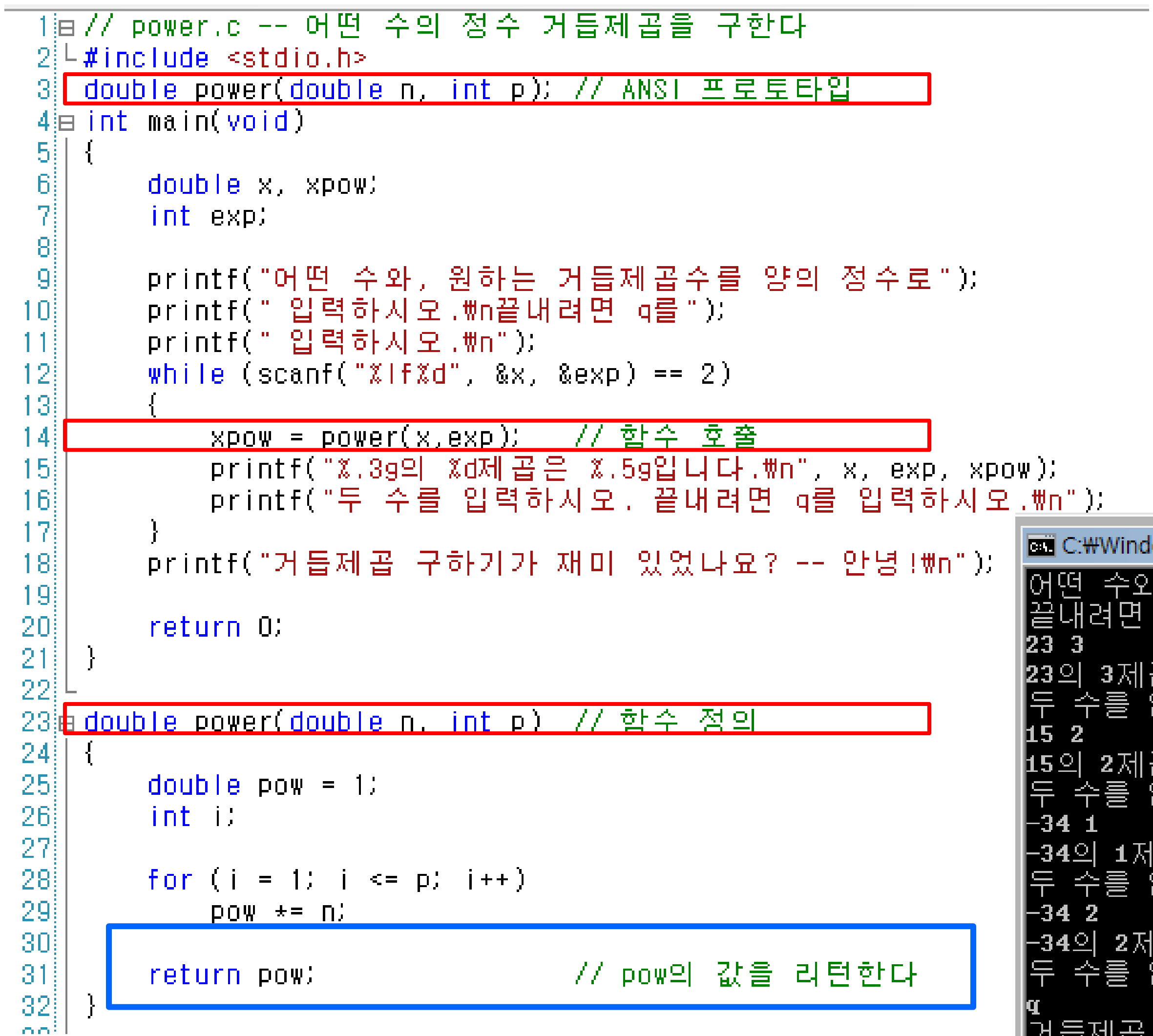


C:\Windows\system32\cmd.exe

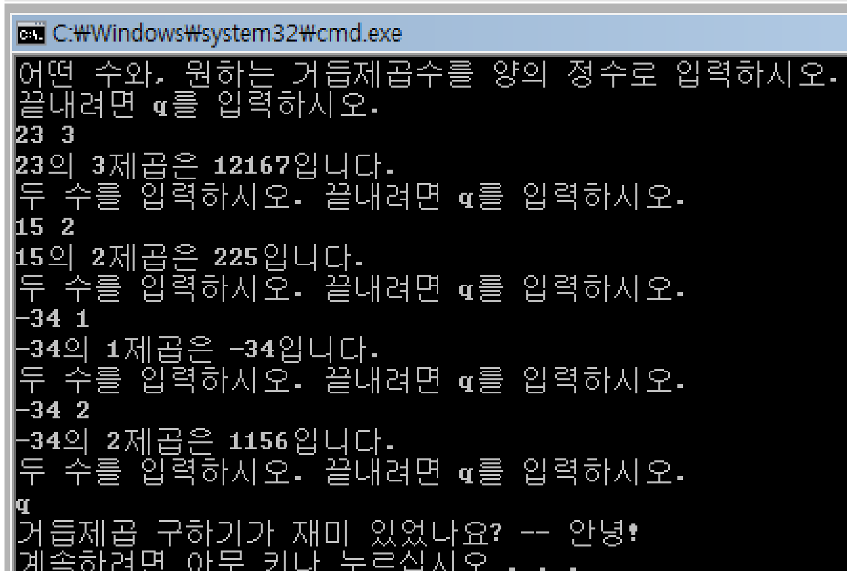
```
#####
#################################
######
계속하려면 아무 키나 누르십시오 . .
```

Loop using a function return value

SEOUL NATIONAL UNIVERSITY



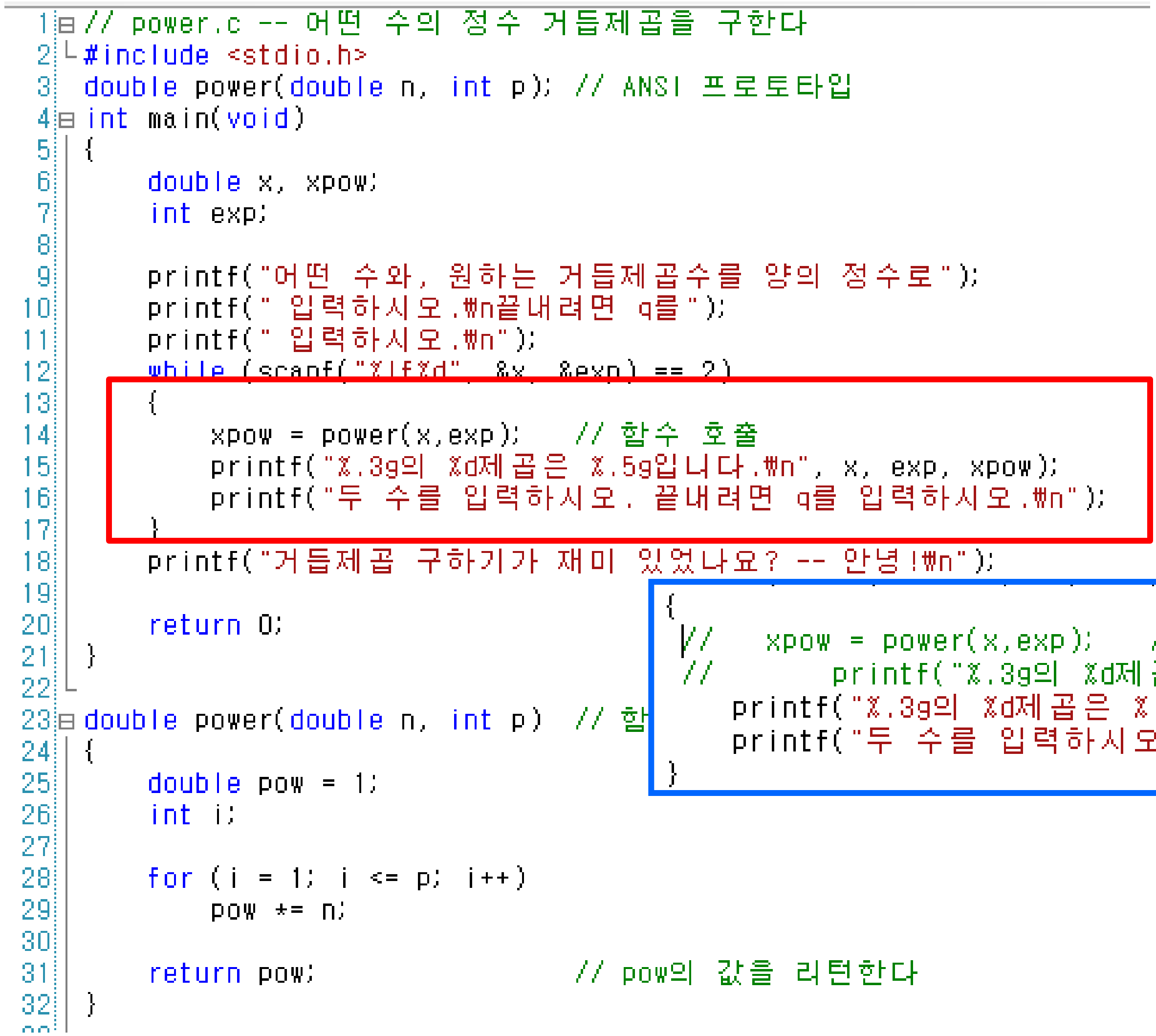
```c
// power.c -- 어떤 수의 정수 거듭제곱을 구한다
#include <stdio.h>
double power(double n, int p); // ANSI 프로토타입
int main(void)
{
    double x, xpow;
    int exp;

    printf("어떤 수와, 원하는 거듭제곱수를 양의 정수로");
    printf(" 입력하시오.₩n끝내려면 q를");
    printf(" 입력하시오.₩n");
    while (scanf("%lf%d", &x, &exp) == 2)
    {
        xpow = power(x,exp);    // 함수 호출
        printf("%.3g의 %d제곱은 %.5g입니다.₩n", x, exp, xpow);
        printf("두 수를 입력하시오. 끝내려면 q를 입력하시오.₩n");
    }
    printf("거듭제곱 구하기가 재미 있었나요? -- 안녕!₩n");

    return 0;
}

double power(double n, int p)  // 함수 정의
{
    double pow = 1;
    int i;

    for (i = 1; i <= p; i++)
        pow *= n;

    return pow;                 // pow의 값을 리턴한다
}
```



C:₩Windows₩system32₩cmd.exe

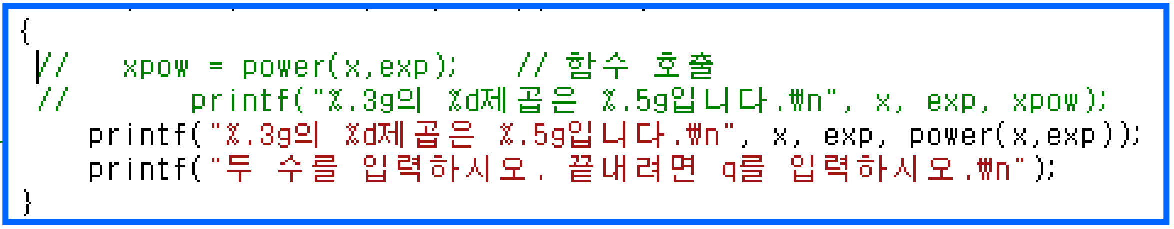
```
어떤 수와, 원하는 거듭제곱수를 양의 정수로 입력하시오.
끝내려면 q를 입력하시오.
23 3
23의 3제곱은 12167입니다.
두 수를 입력하시오. 끝내려면 q를 입력하시오.
15 2
15의 2제곱은 225입니다.
두 수를 입력하시오. 끝내려면 q를 입력하시오.
-34 1
-34의 1제곱은 -34입니다.
두 수를 입력하시오. 끝내려면 q를 입력하시오.
-34 2
-34의 2제곱은 1156입니다.
두 수를 입력하시오. 끝내려면 q를 입력하시오.
q
거듭제곱 구하기가 재미 있었나요? -- 안녕!
계속하려면 아무 키나 누르십시오 . . .
```

Loop using a function return value



```c
// power.c -- 어떤 수의 정수 거듭제곱을 구한다
#include <stdio.h>
double power(double n, int p); // ANSI 프로토타입
int main(void)
{
    double x, xpow;
    int exp;

    printf("어떤 수와, 원하는 거듭제곱수를 양의 정수로");
    printf(" 입력하시오.₩n끝내려면 q를");
    printf(" 입력하시오.₩n");
    while (scanf("%lf%d", &x, &exp) == 2)
    {
        xpow = power(x,exp);    // 함수 호출
        printf("%.3g의 %d제곱은 %.5g입니다.₩n", x, exp, xpow);
        printf("두 수를 입력하시오. 끝내려면 q를 입력하시오.₩n");
    }
    printf("거듭제곱 구하기가 재미 있었나요? -- 안녕!₩n");

    return 0;
}

double power(double n, int p)  // 함
{
    double pow = 1;
    int i;

    for (i = 1; i <= p; i++)
        pow *= n;

    return pow;                 // pow의 값을 리턴한다
}
```

same results



```c
{
//    xpow = power(x,exp);    // 함수 호출
//        printf("%.3g의 %d제곱은 %.5g입니다.₩n", x, exp, xpow);
    printf("%.3g의 %d제곱은 %.5g입니다.₩n", x, exp, power(x,exp));
    printf("두 수를 입력하시오. 끝내려면 q를 입력하시오.₩n");
}
```

Today
Chapter 6. C primer Plus

- C control statements: Looping
  - for
  - While

    } Entry-condition loop
  - Do while —— Exit-condition loop
- What is true/nested loop
- Introduction to array
- Using a function return value

29 April
Chapter 7. C primer Plus

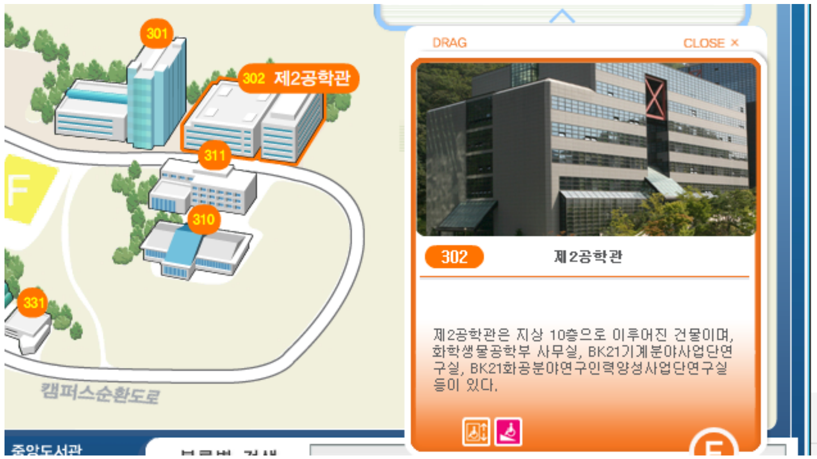
SEOUL NATIONAL UNIVERSITY

- C control statements: Branching and Jumps
  - if, else
  - Switch
  - Continue, break, goto
- Logical operators: && ||
- Character I/O functions: getchar() and putchar()

Mid-term exam

SEOUL NATIONAL UNIVERSITY

- 22 April 13:00 – 15:00
- Venue: 302-105 (제2공학관)



제2공학관은 지상 10층으로 이루어진 건물이며, 화학생물공학부 사무실, BK21기계분야사업단연구실, BK21화공분야연구인력양성사업단연구실 등이 있다.

- Good Luck for your exam!!!