Object oriented programming In C++

Function 2

Professor 최학남

xncui@inha.ac.kr

Office: high-tech 401

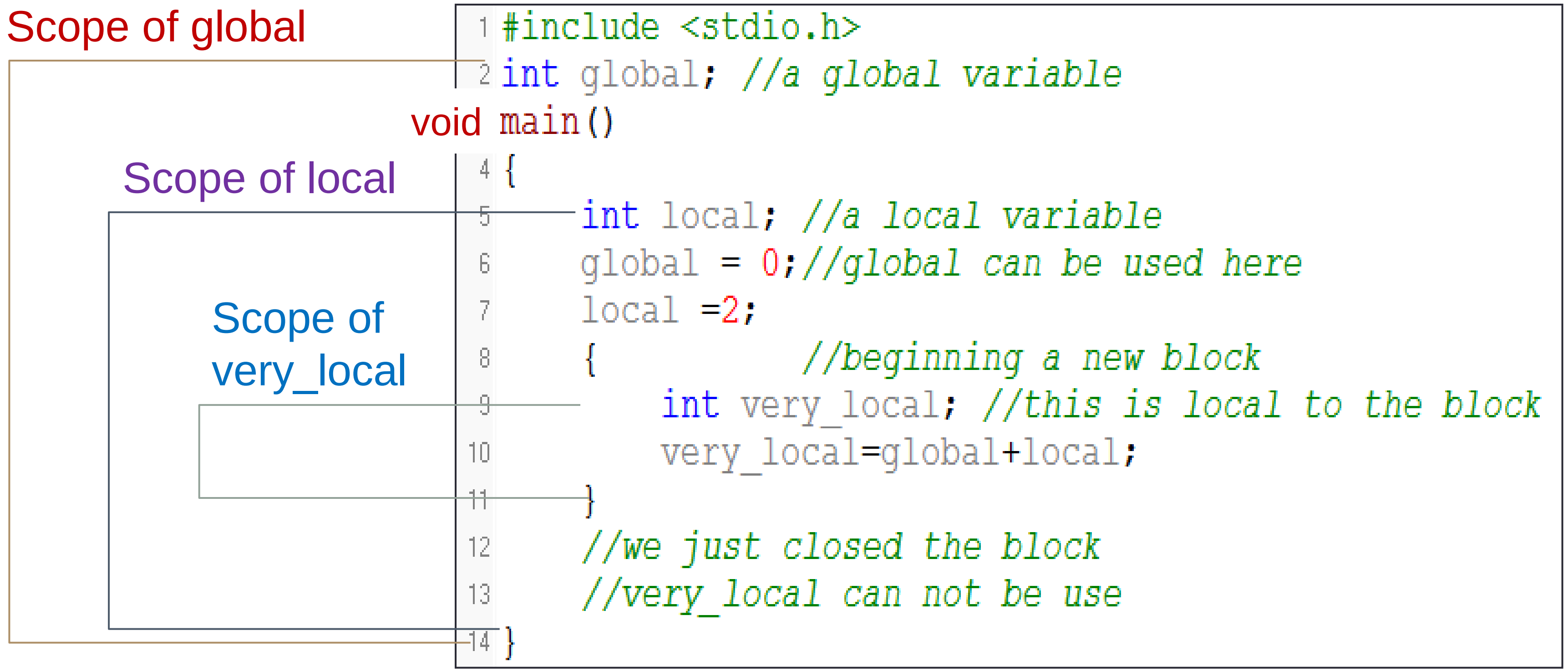
Local and Global Variables

- Local variables
  - ✓ The local variables are defined within the body of the function or the block
  - ✓ The variable defined is local to that function or block only
  - ✓ Other functions can not access these variables

Local and Global Variables

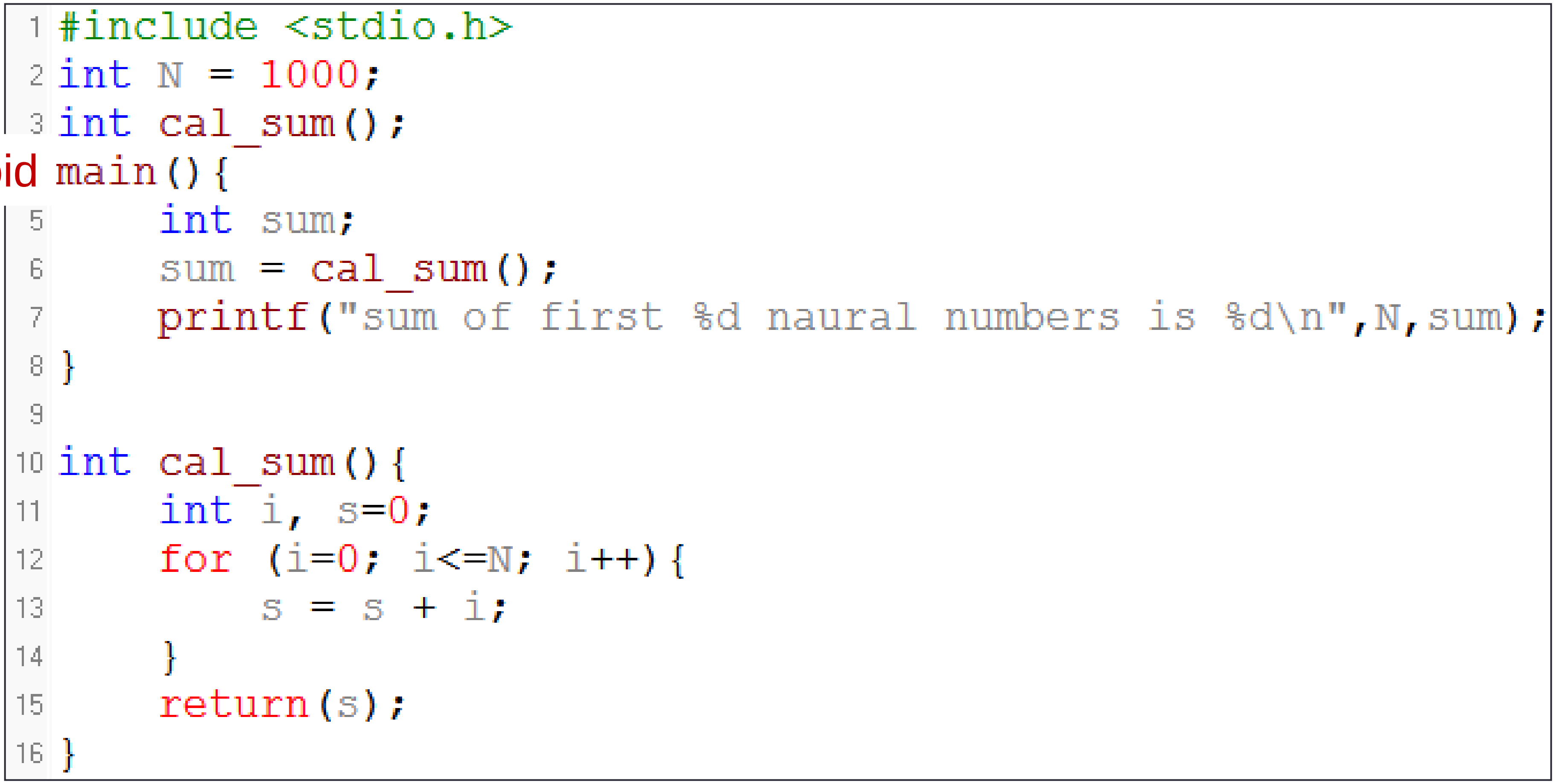
- Global variables
  - ✓Global variables are defined outside the main() function
  - ✓Multiple functions can use them

Example 4-1:
Scope of the local and global variables



```
#include <stdio.h>
int global; //a global variable
void main()
{
    int local; //a local variable
    global = 0;//global can be used here
    local =2;
    {           //beginning a new block
        int very_local; //this is local to the block
        very_local=global+local;
    }
    //we just closed the block
    //very_local can not be use
}
```

Example 4-2



```c
#include <stdio.h>
int N = 1000;
int cal_sum();
void main(){
    int sum;
    sum = cal_sum();
    printf("sum of first %d naural numbers is %d\n",N,sum);
}

int cal_sum(){
    int i, s=0;
    for (i=0; i<=N; i++){
        s = s + i;
    }
    return(s);
}
```

Example 4-3

```
void sub1(void)
{
        int x;                  // local variable x
        x = 0;                  // OK
        {
                int y;          // local variable y
                x = 1;          // OK
                y = 2;          // OK
        }
        x = 3;
        y = 4;
}
```

```
void sub2(void)
{
        int x;                  // local variable x
        x = 0;                  // initialization for local variable x
        {
                int y;          // local variable y
                x = 1;
                y = 2;
        }
        {
                int y;          // local variable y
                x = 3;
                y = 3;
        }

        printf("x=%d\n",x);

}
```

Category of Function

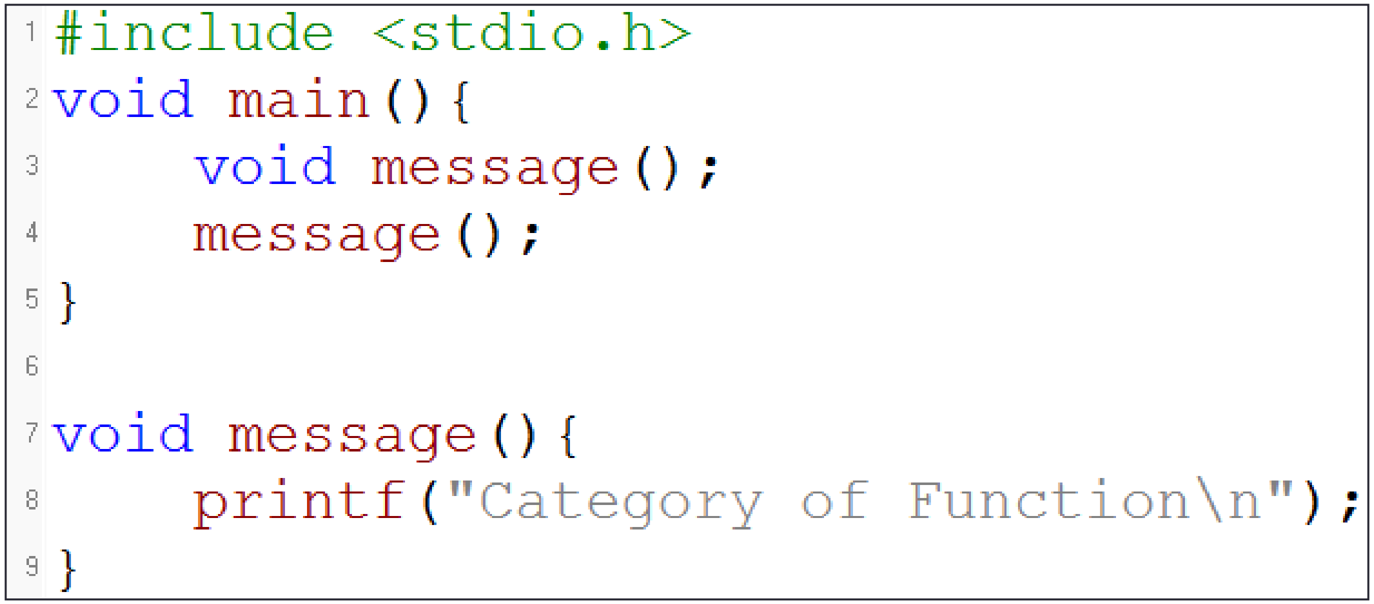
- Without arguments and no return value
- Without arguments and with return value
- With arguments and no return value
- With arguments and with return value

Category of Function

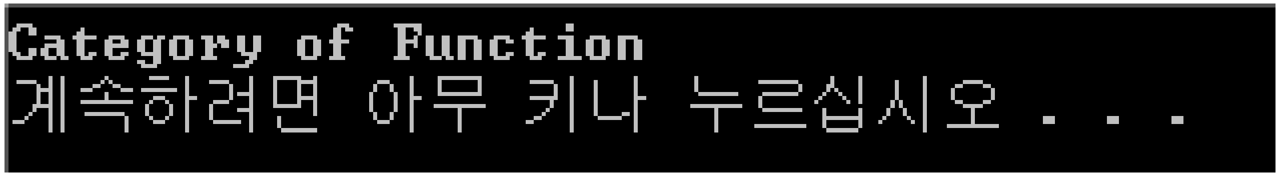
- Without arguments and no return value
  - ✓ Neither the data is passed through the calling function nor the data is sent back from the classed function
  - ✓ There is no data transfer between calling and the called function
  - ✓ If such functions are used to perform any operation, they can independently
  - ✓ Such functions may be useful to print some messages etc.

Category of Function

- Without arguments and no return value
  - ✓Example 5



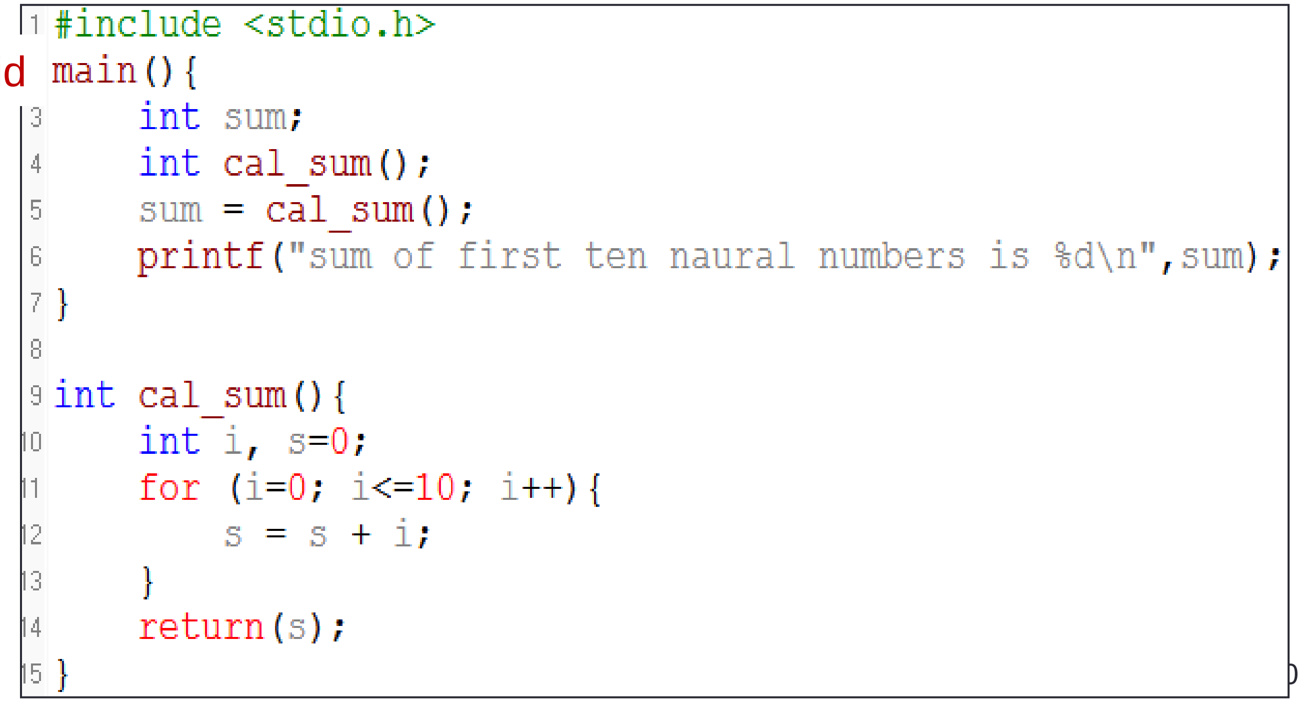
```c
#include <stdio.h>
void main(){
	void message();
	message();
}

void message(){
	printf("Category of Function\n");
}
```



```
Category of Function
계속하려면 아무 키나 누르십시오 . . .
```

Category of Function

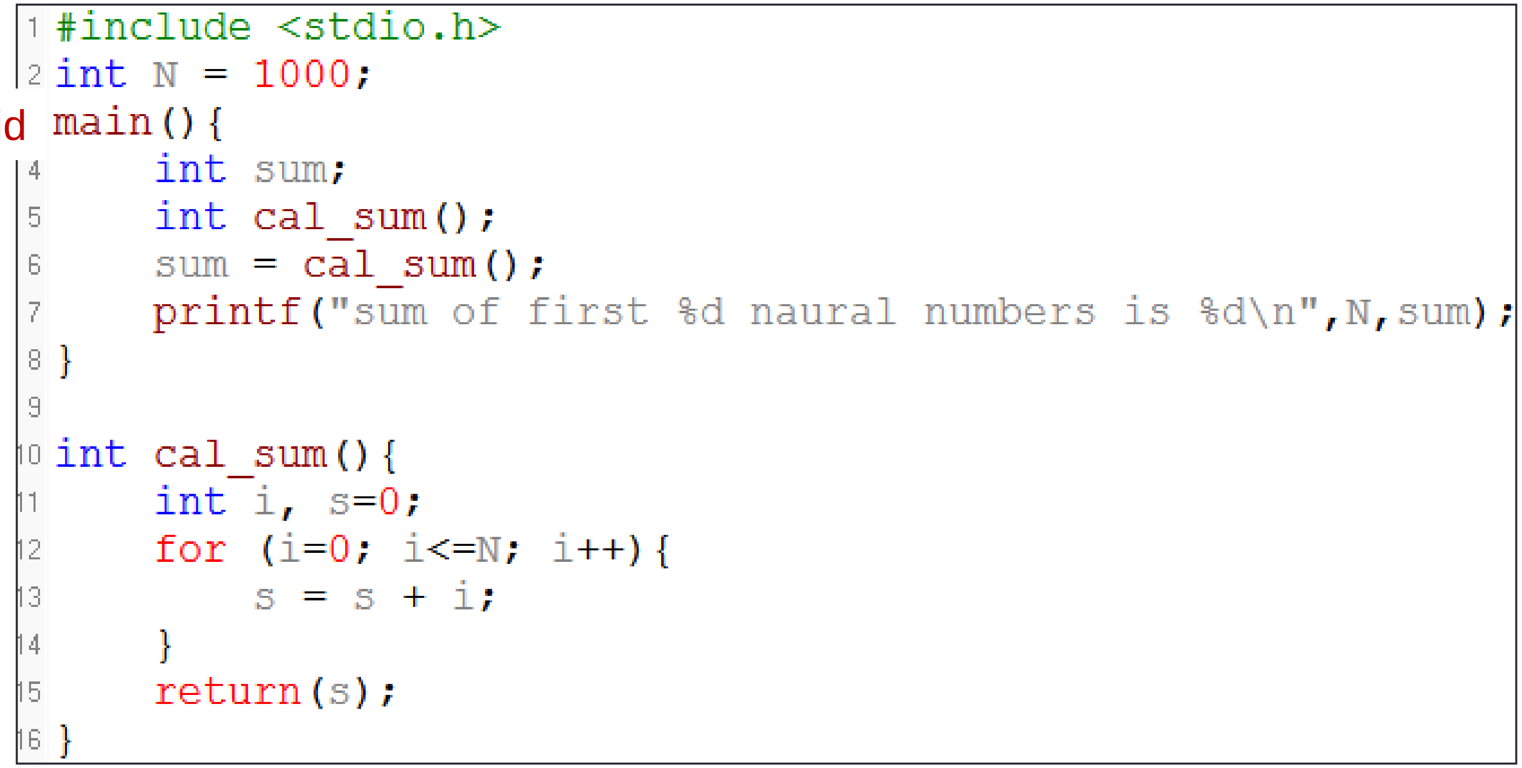
- Without arguments and with return value
  - ✓ Suppose if a function dose not receive any data from calling function but does send some value to the calling function, then it falls in this category
  - ✓ Example 6-1



```
#include <stdio.h>
void main(){
    int sum;
    int cal_sum();
    sum = cal_sum();
    printf("sum of first ten naural numbers is %d\n",sum);
}

int cal_sum(){
    int i, s=0;
    for (i=0; i<=10; i++){
        s = s + i;
    }
    return(s);
}
```

Category of Function

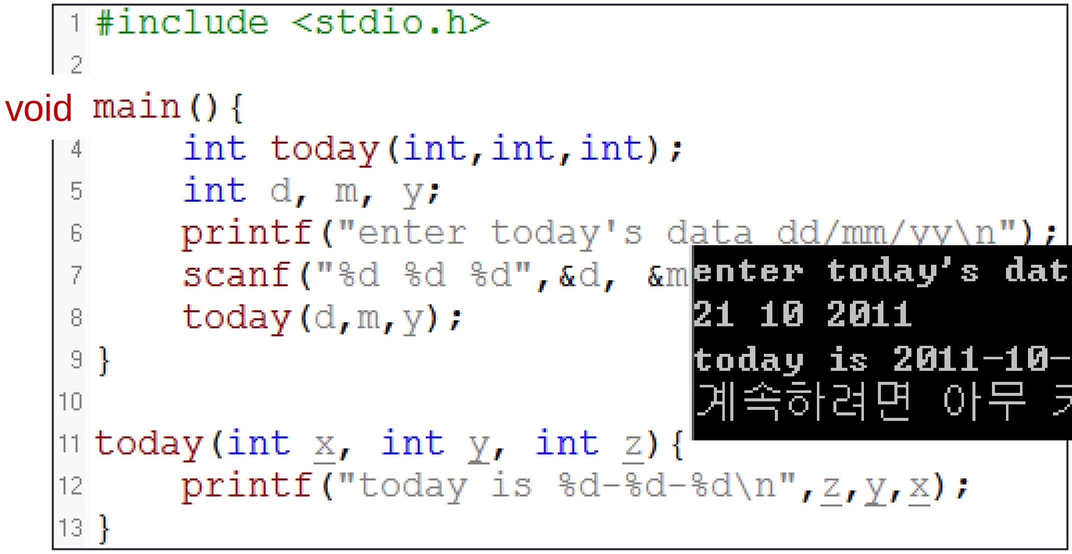
- Without arguments and with return value
  - ✓Example 6-2: Global variable



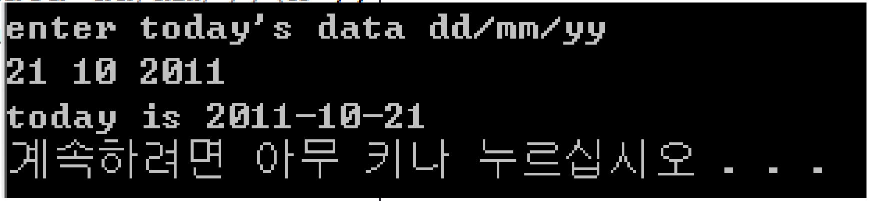
```c
#include <stdio.h>
int N = 1000;
void main(){
    int sum;
    int cal_sum();
    sum = cal_sum();
    printf("sum of first %d naural numbers is %d\n",N,sum);
}

int cal_sum(){
    int i, s=0;
    for (i=0; i<=N; i++){
        s = s + i;
    }
    return(s);
}
```

Category of Function

- With arguments and no return value
  - ✓ Arguments are passed to the calling function
  - ✓ The called function operates on the values
  - ✓ But no results is sent back
  - ✓ Example 6-3



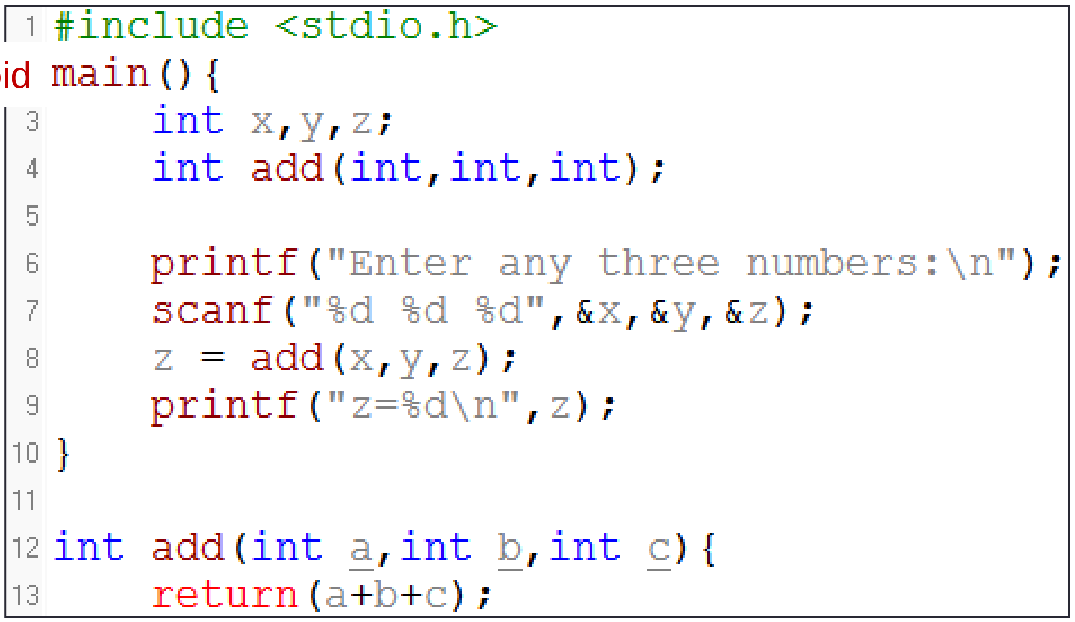
```
#include <stdio.h>

void main(){
    int today(int,int,int);
    int d, m, y;
    printf("enter today's data dd/mm/yy\n");
    scanf("%d %d %d",&d, &m
    today(d,m,y);
}

today(int x, int y, int z){
    printf("today is %d-%d-%d\n",z,y,x);
}
```



```
enter today's data dd/mm/yy
21 10 2011
today is 2011-10-21
계속하려면 아무 키나 누르십시오 . . .
```

Category of Function

- With arguments and with return value
  - ✓ Arguments are passed to the calling function
  - ✓ After necessary processing the return value will be sent back
  - ✓ Example 7



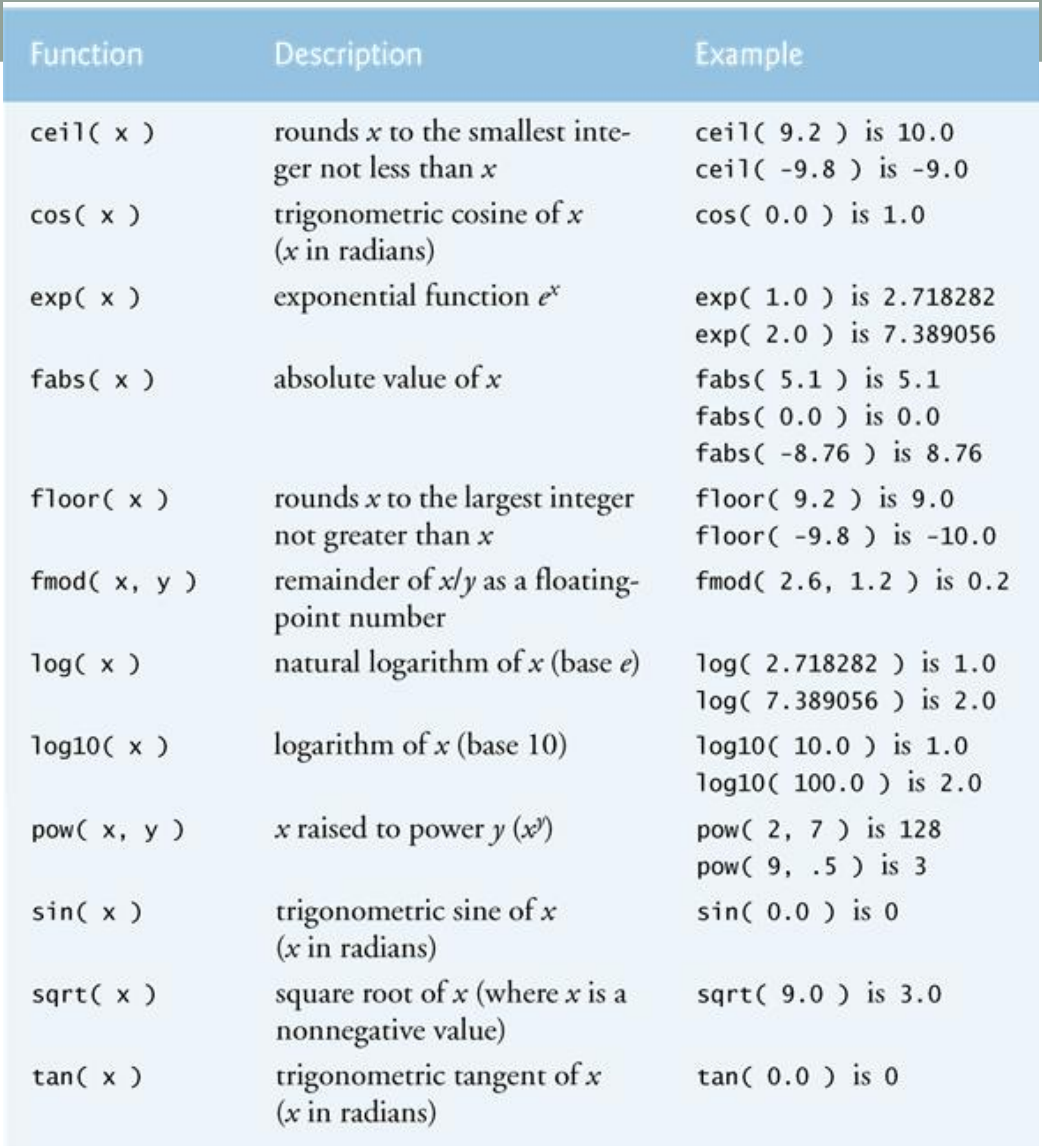
```c
#include <stdio.h>
void main(){
    int x,y,z;
    int add(int,int,int);

    printf("Enter any three numbers:\n");
    scanf("%d %d %d",&x,&y,&z);
    z = add(x,y,z);
    printf("z=%d\n",z);
}

int add(int a,int b,int c){
    return(a+b+c);
```

Library functions

- Powerful library functions are provided by C compiler
  - ✓ stdio.lib, math.lib, time.lib, stdlib.lib
  - ✓ …
- How to use the library function?
  - ✓ Include the header files

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

#include <stdlib.h>       #include <time.h>
```

Math library



Function	Description	Example

`ceil( x )`	rounds x to the smallest integer not less than x	`ceil( 9.2 )` is `10.0`
		`ceil( -9.8 )` is `-9.0`

`cos( x )`	trigonometric cosine of x (x in radians)	`cos( 0.0 )` is `1.0`

`exp( x )`	exponential function e^x	`exp( 1.0 )` is `2.718282`
		`exp( 2.0 )` is `7.389056`

`fabs( x )`	absolute value of x	`fabs( 5.1 )` is `5.1`
		`fabs( 0.0 )` is `0.0`
		`fabs( -8.76 )` is `8.76`

`floor( x )`	rounds x to the largest integer not greater than x	`floor( 9.2 )` is `9.0`
		`floor( -9.8 )` is `-10.0`

`fmod( x, y )`	remainder of x/y as a floating-point number	`fmod( 2.6, 1.2 )` is `0.2`

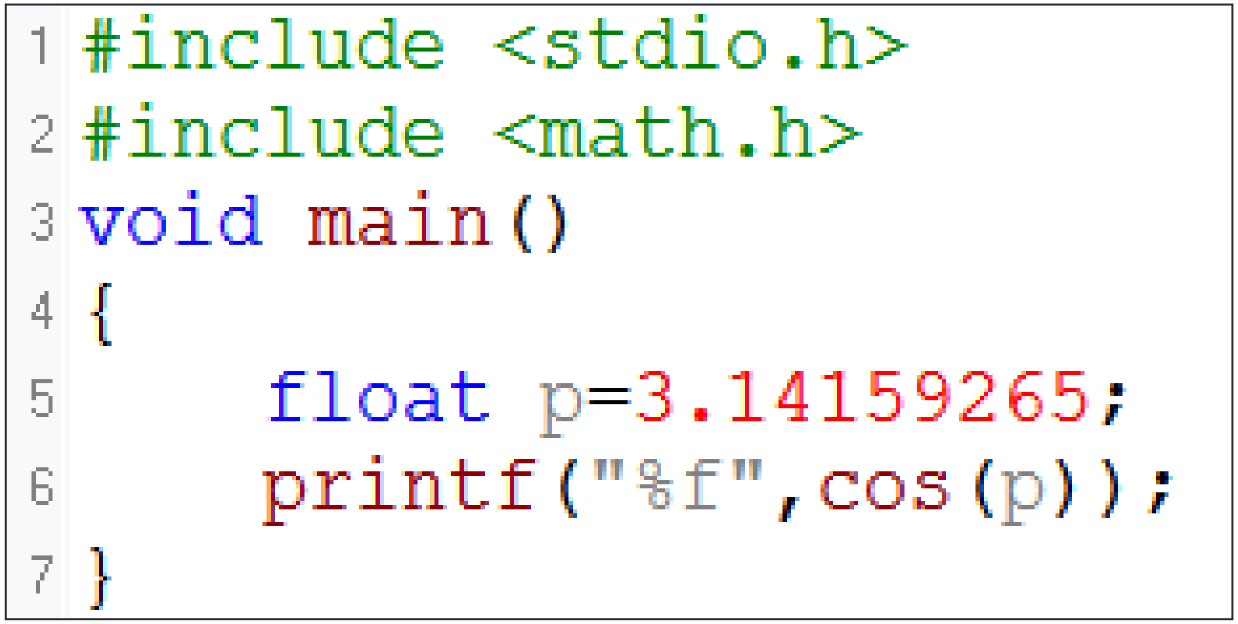
`log( x )`	natural logarithm of x (base e)	`log( 2.718282 )` is `1.0`
		`log( 7.389056 )` is `2.0`

`log10( x )`	logarithm of x (base 10)	`log10( 10.0 )` is `1.0`
		`log10( 100.0 )` is `2.0`

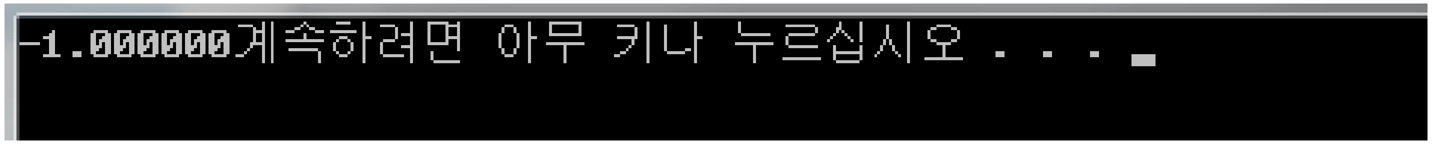
`pow( x, y )`	x raised to power y (x^y)	`pow( 2, 7 )` is `128`
		`pow( 9, .5 )` is `3`

`sin( x )`	trigonometric sine of x (x in radians)	`sin( 0.0 )` is `0`
`sqrt( x )`	square root of x (where x is a nonnegative value)	`sqrt( 9.0 )` is `3.0`
`tan( x )`	trigonometric tangent of x (x in radians)	`tan( 0.0 )` is `0`

Example 8-1: math library



```c
#include <stdio.h>
#include <math.h>
void main()
{
	float p=3.14159265;
	printf("%f",cos(p));
}
```

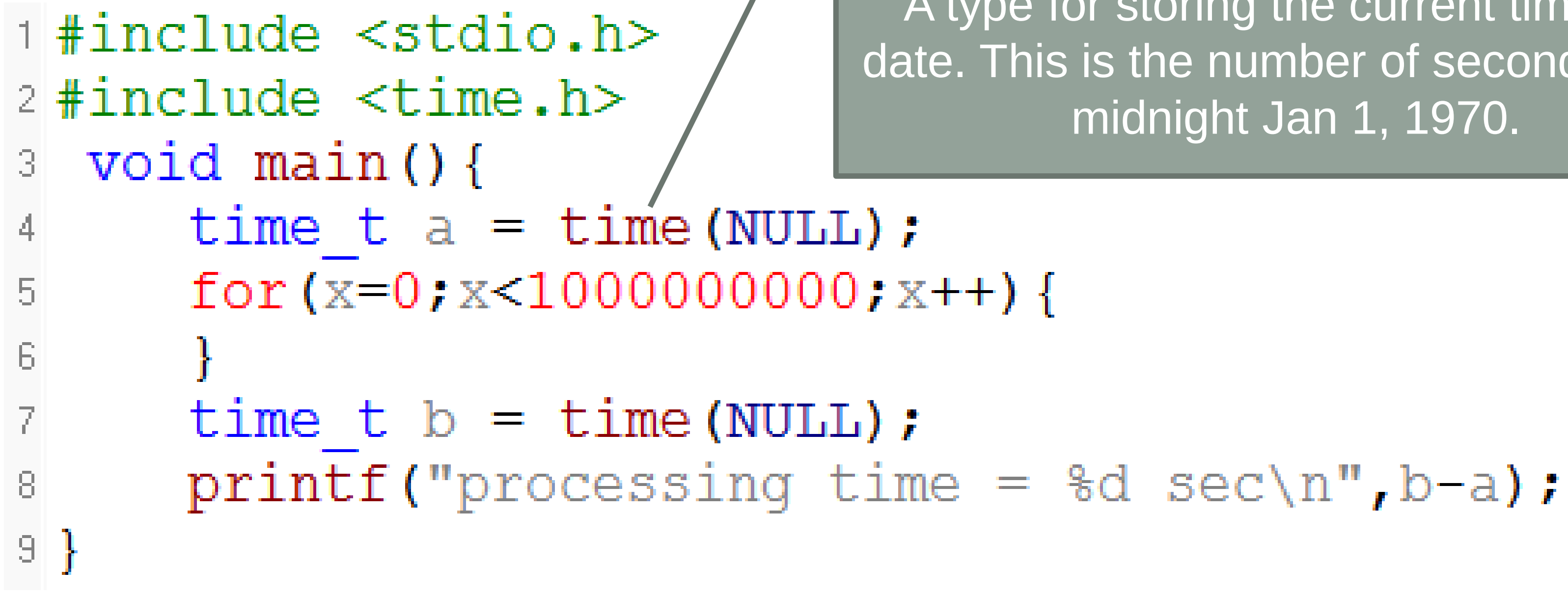


```
-1.000000계속하려면 아무 키나 누르십시오 . . .
```

Example 8-2: stdlib library random number generator

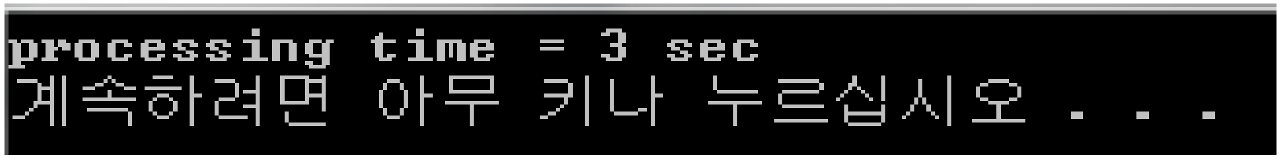
```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

// print out the number of n random number
void get_random( int n )
{
        int i;
        for( i = 0; i < n; i++ )
                printf( "  %6d\n", rand() );
}

int main( void )
{
        // the seed point is the current time.
        // will get the different results
        srand( (unsigned)time( NULL ) );
        get_random( 10 );
        return 0;
}
```

Example 8-3: time library



```c
#include <stdio.h>
#include <time.h>
 void main(){
    time_t a = time(NULL);
    for(x=0;x<1000000000;x++){
    }
    time_t b = time(NULL);
    printf("processing time = %d sec\n",b-a);
}
```

A type for storing the current time and date. This is the number of seconds since midnight Jan 1, 1970.



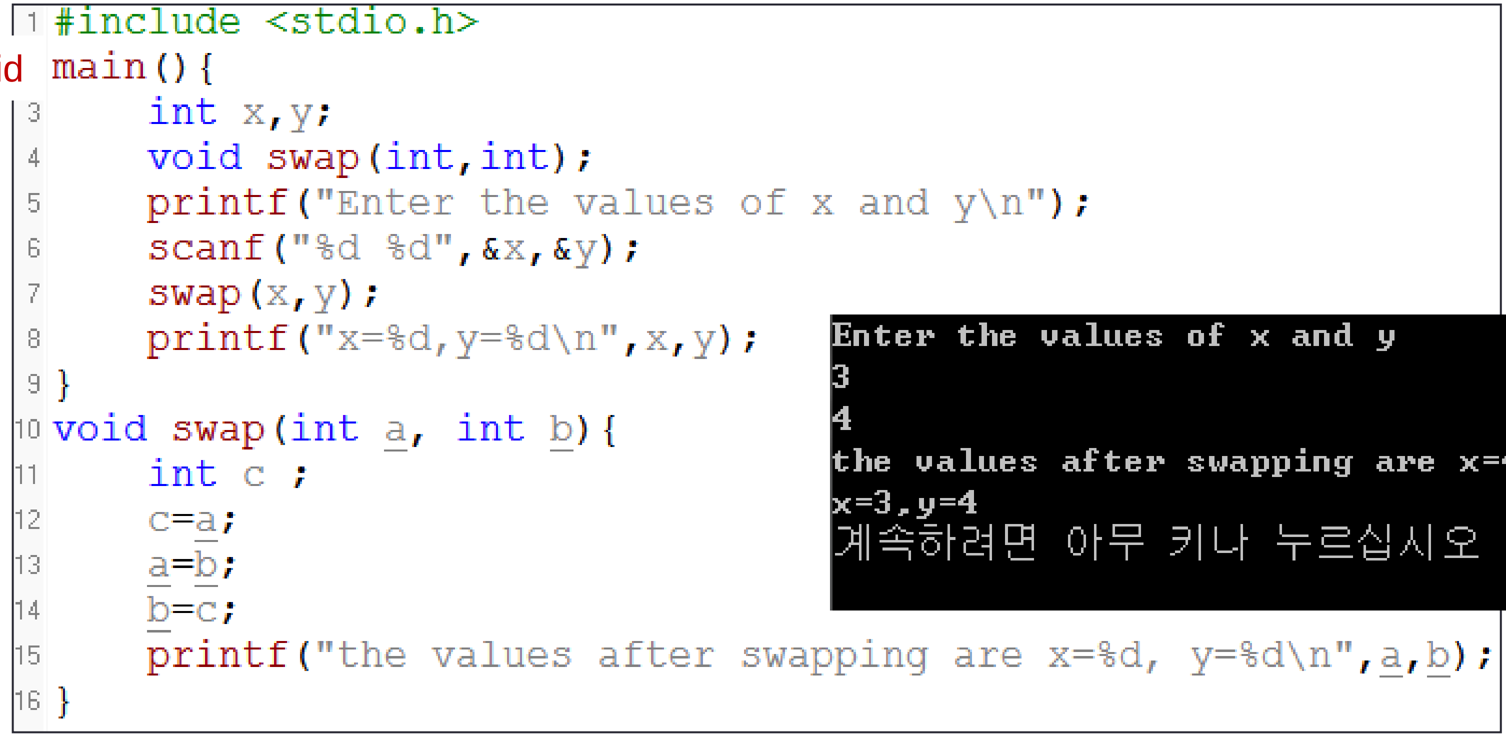
```
processing time = 3 sec
계속하려면 아무 키나 누르십시오 . . .
```

Call by Value and Reference

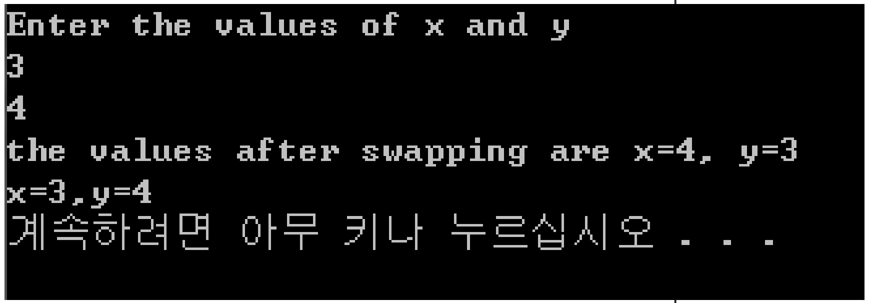
- There are two ways in which we can pass arguments to the function
  - ✓ Call by value
  - ✓ Call by reference

Call by Value and Reference

- Example 9-1 : Call by value



```c
#include <stdio.h>
void main(){
    int x,y;
    void swap(int,int);
    printf("Enter the values of x and y\n");
    scanf("%d %d",&x,&y);
    swap(x,y);
    printf("x=%d,y=%d\n",x,y);
}
void swap(int a, int b){
    int c ;
    c=a;
    a=b;
    b=c;
    printf("the values after swapping are x=%d, y=%d\n",a,b);
}
```



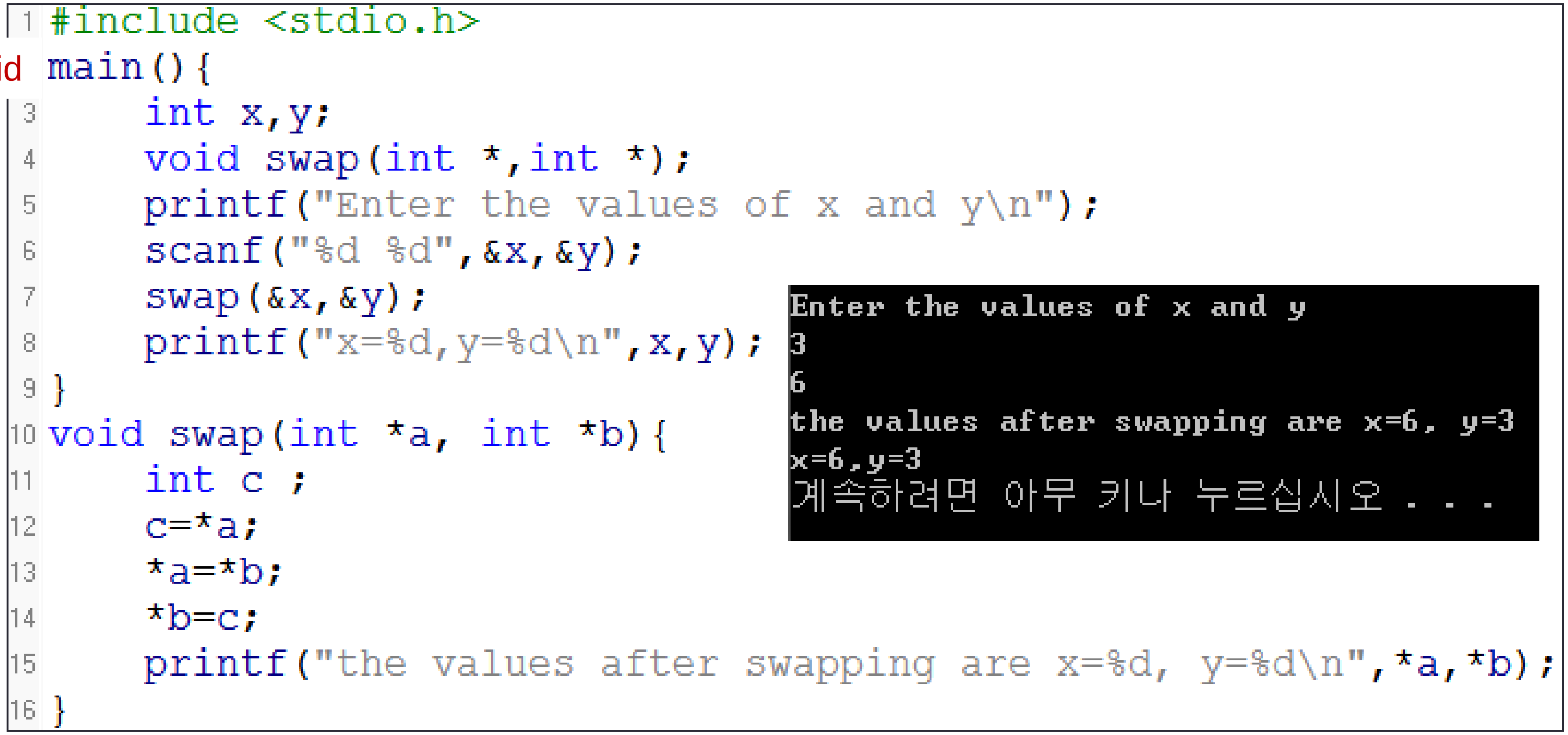
```
Enter the values of x and y
3
4
the values after swapping are x=4, y=3
x=3,y=4
계속하려면 아무 키나 누르십시오 . . .
```

Call by Value and Reference

- Call by reference
  - ✓ In this type, instead of passing values, **addresses/references are passed**
  - ✓ Function operates on **addresses rather than values**
  - ✓ The **formal arguments(parameters)** are pointers to the **actual arguments**
  - ✓ In this type, formal arguments point to the actual argument. Hence changes made in the arguments are permanent

Call by Value and Reference

- Example 9-2: Call by reference



```c
#include <stdio.h>
void main(){
    int x,y;
    void swap(int *,int *);
    printf("Enter the values of x and y\n");
    scanf("%d %d",&x,&y);
    swap(&x,&y);
    printf("x=%d,y=%d\n",x,y);
}
void swap(int *a, int *b){
    int c ;
    c=*a;
    *a=*b;
    *b=c;
    printf("the values after swapping are x=%d, y=%d\n",*a,*b);
}
```

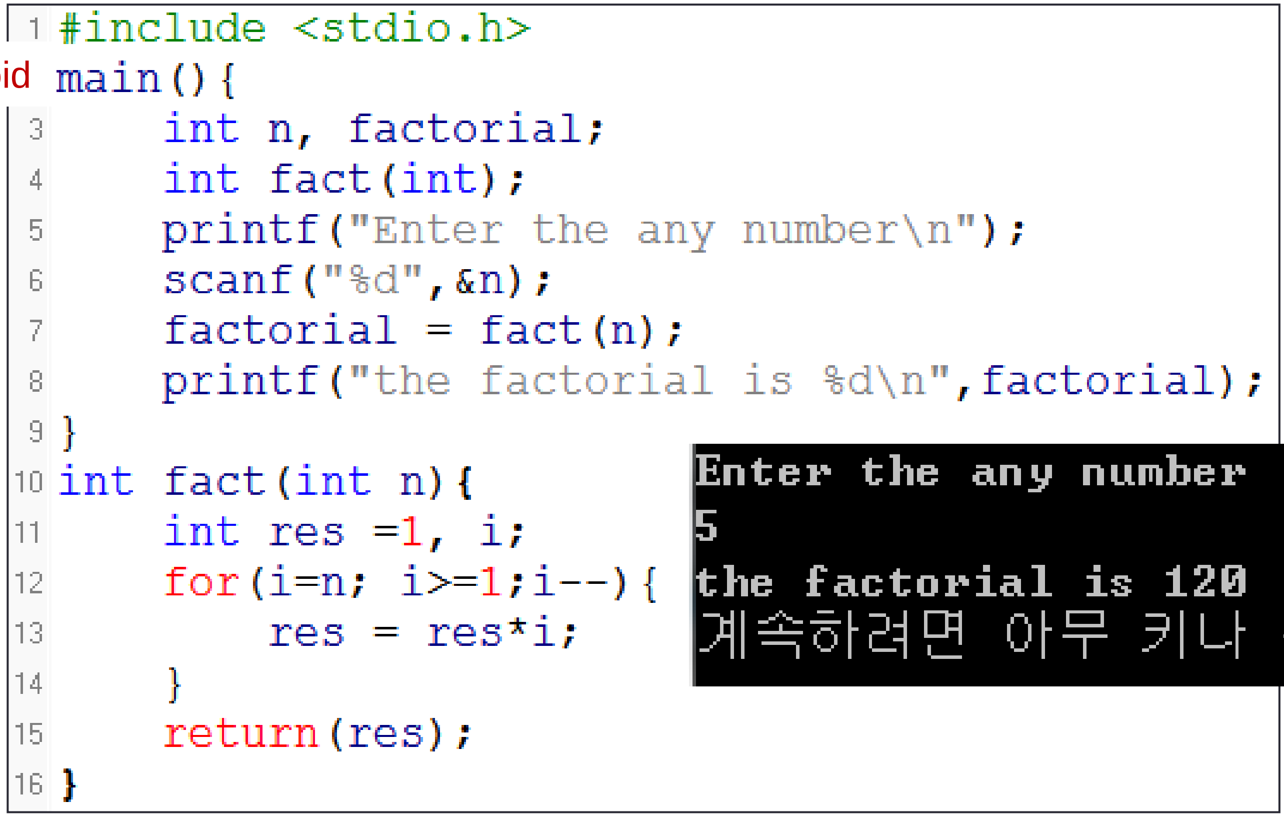
```
Enter the values of x and y
3
6
the values after swapping are x=6, y=3
x=6,y=3
계속하려면 아무 키나 누르십시오 . . .
```

Recursion

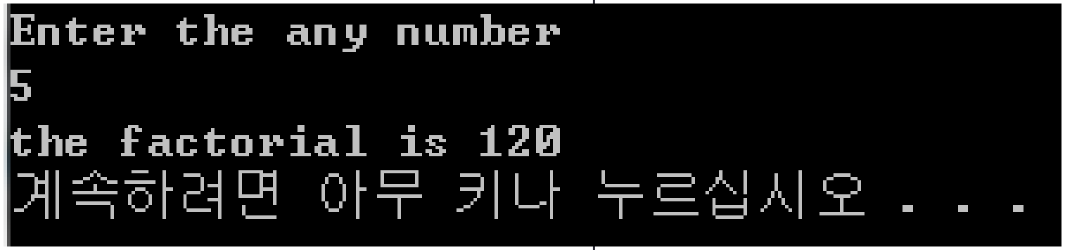
- Within a function body, if the function **calls itself**, the mechanism is known as "**recursion**" and the function is known as "**recursive function**"

Recursion

➢Example 10-1 : Non-recursive way to find factorial



```c
#include <stdio.h>
void main(){
     int n, factorial;
     int fact(int);
     printf("Enter the any number\n");
     scanf("%d",&n);
     factorial = fact(n);
     printf("the factorial is %d\n",factorial);
}
int fact(int n){
     int res =1, i;
     for(i=n; i>=1;i--){
          res = res*i;
     }
     return(res);
}
```

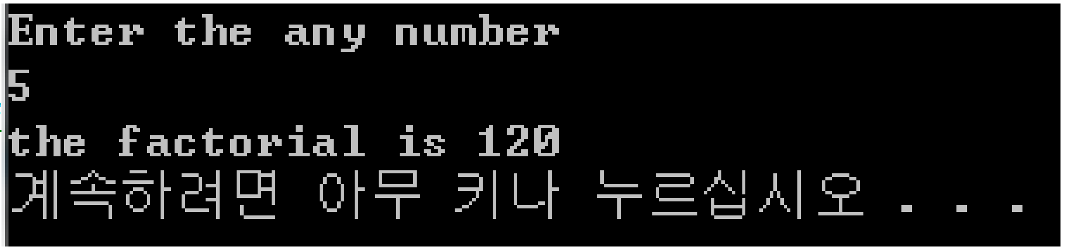


```
Enter the any number
5
the factorial is 120
계속하려면 아무 키나 누르십시오 . . .
```

Recursion

- Example 10-2 : Recursive way to find factorial

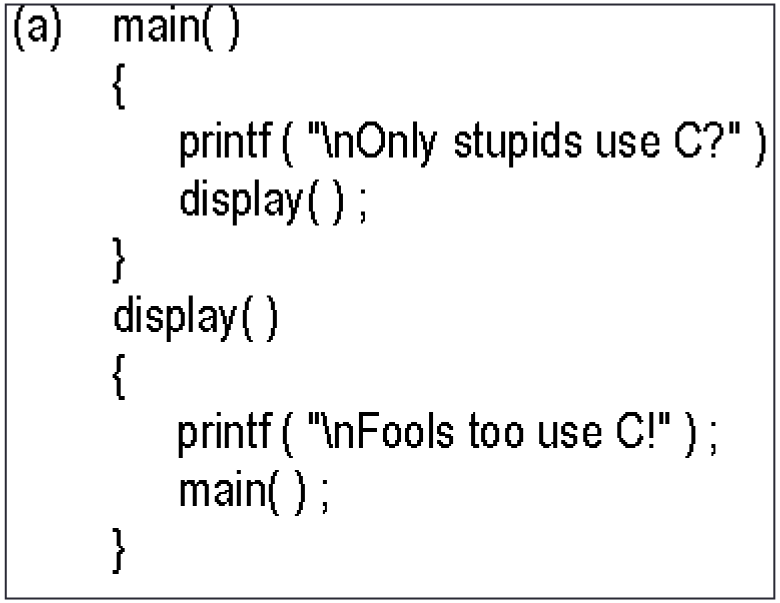
```c
#include <stdio.h>
void main(){
     int n, factorial;
     int fact(int);
     printf("Enter the any number\n");
     scanf("%d",&n);
     factorial = fact(n);
     printf("the factorial is %d\n",factorial);
}
int fact(int n){
     int res =1;
     if(n==1){        // termi
          return(res);
     }
     else{
          res = n*fact(n-1);   //recursive call
     }
     return(res);
}
```



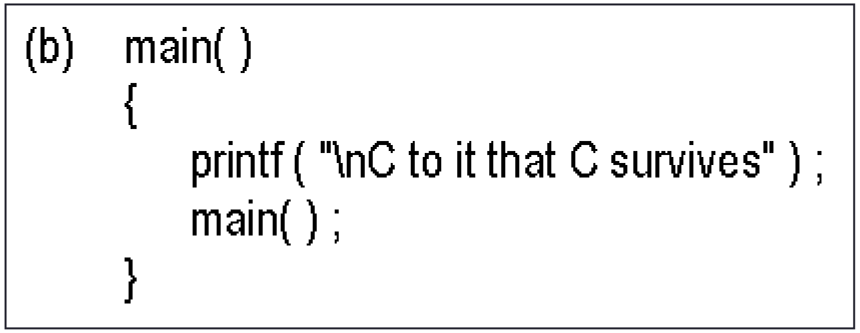
```
Enter the any number
5
the factorial is 120
계속하려면 아무 키나 누르십시오 . . .
```

Exercise 1.

➢What will be the output of the following programs



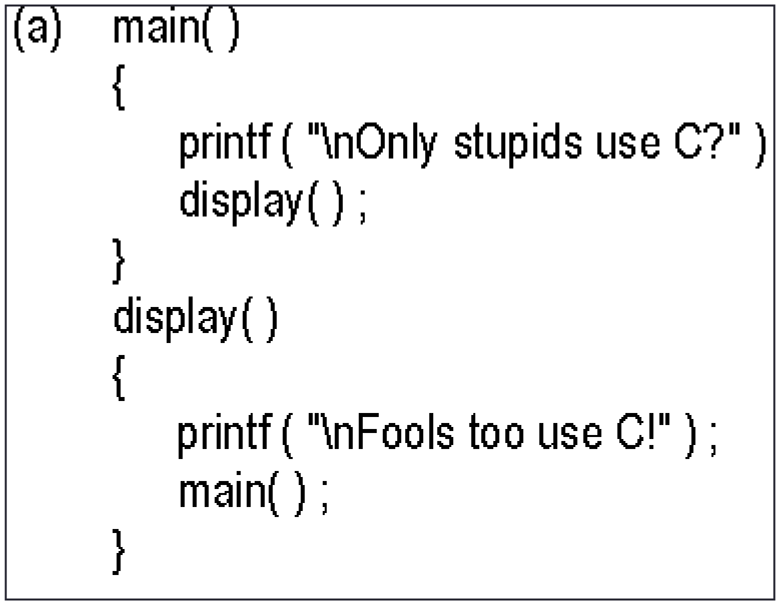
```
(a)   main( )
      {
            printf ( "\nOnly stupids use C?" )
            display( ) ;
      }
      display( )
      {
            printf ( "\nFools too use C!" ) ;
            main( ) ;
      }
```



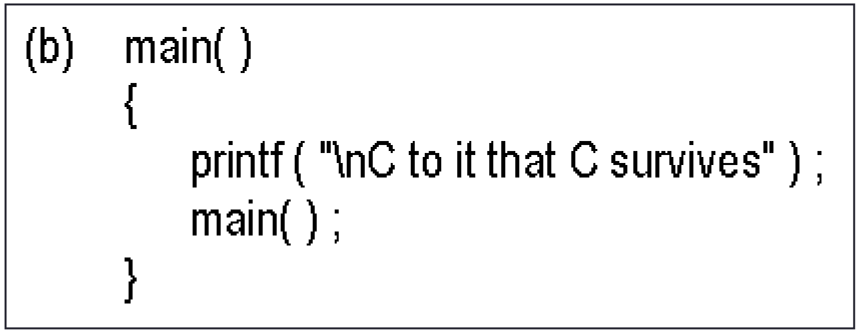
```
(b)   main( )
      {
            printf ( "\nC to it that C survives" ) ;
            main( ) ;
      }
```

Exercise 1.

- What will be the output of the following programs



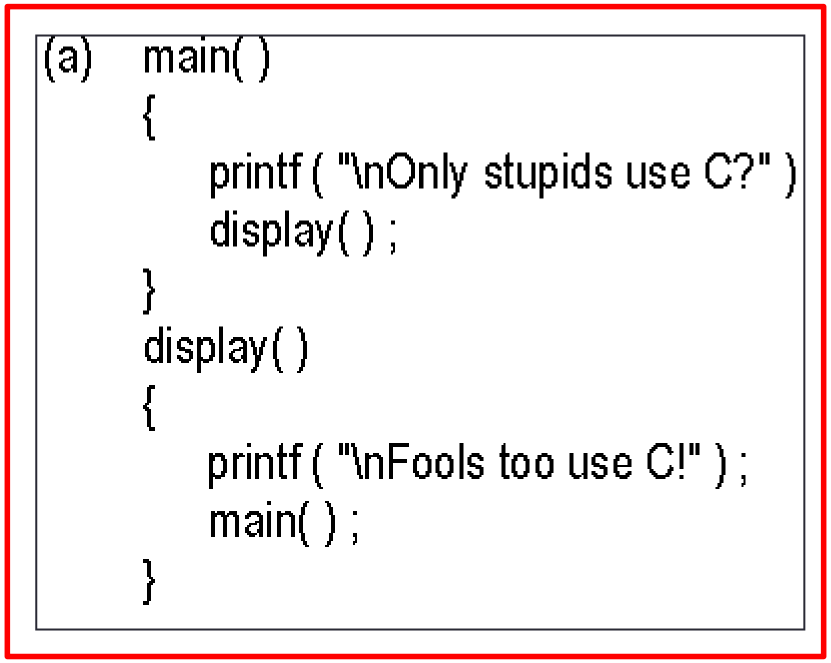
```
(a)   main( )
      {
           printf ( "\nOnly stupids use C?" )
           display( ) ;
      }
      display( )
      {
           printf ( "\nFools too use C!" ) ;
           main( ) ;
      }
```



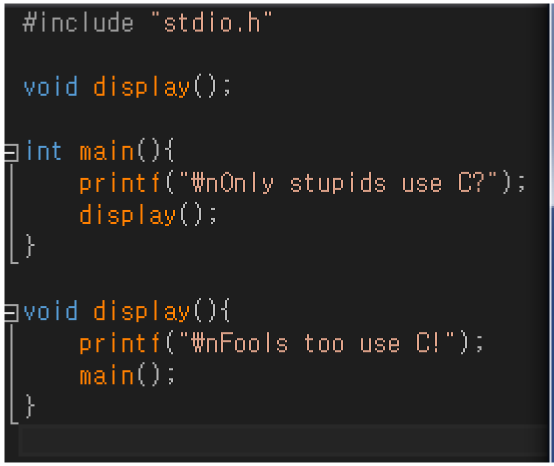
```
(b)   main( )
      {
           printf ( "\nC to it that C survives" ) ;
           main( ) ;
      }
```

Exercise 2.

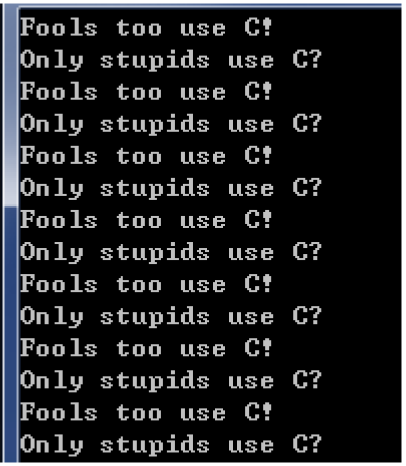
- What will be the output of the following programs



```
(a)   main( )
      {
          printf ( "\nOnly stupids use C?" ) ;
          display( ) ;
      }
      display( )
      {
          printf ( "\nFools too use C!" ) ;
          main( ) ;
      }
```



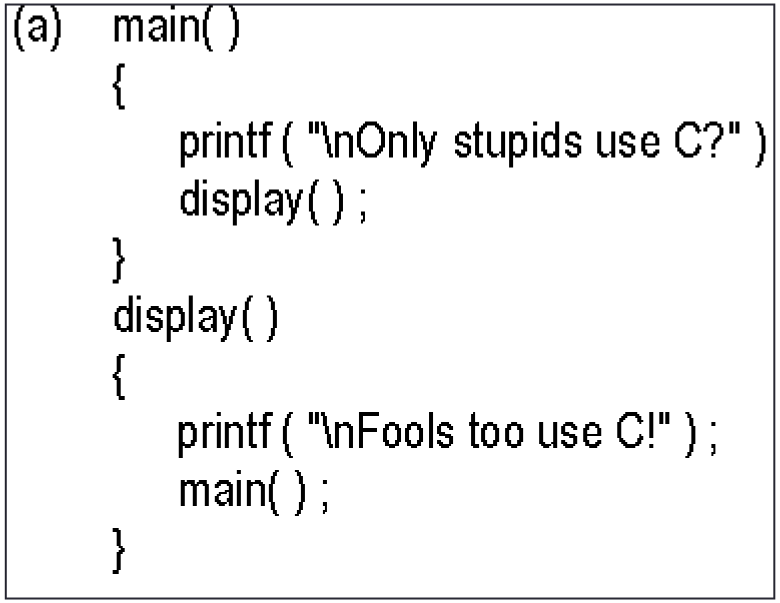
```
#include "stdio.h"

void display();

int main(){
    printf("₩nOnly stupids use C?");
    display();
}

void display(){
    printf("₩nFools too use C!");
    main();
}
```



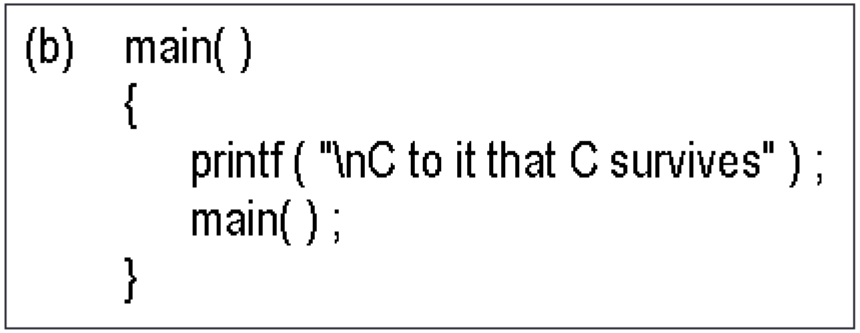
```
Fools too use C!
Only stupids use C?
Fools too use C!
Only stupids use C?
Fools too use C!
Only stupids use C?
Fools too use C!
Only stupids use C?
Fools too use C!
Only stupids use C?
Fools too use C!
Only stupids use C?
Fools too use C!
Only stupids use C?
```

Exercise 2.

➢What will be the output of the following programs



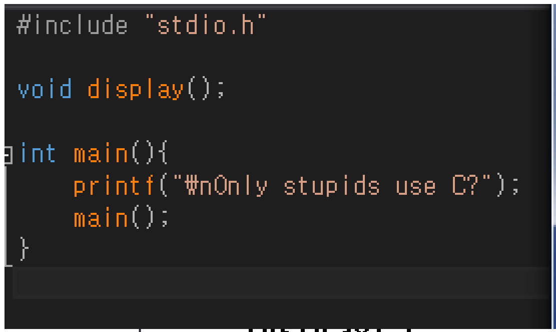
```
(a)   main( )
      {
          printf ( "\nOnly stupids use C?" )
          display( ) ;
      }
      display( )
      {
          printf ( "\nFools too use C!" ) ;
          main( ) ;
      }
```



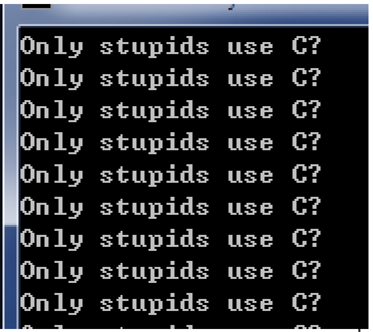
```
(b)   main( )
      {
          printf ( "\nC to it that C survives" ) ;
          main( ) ;
      }
```

Exercise 2.

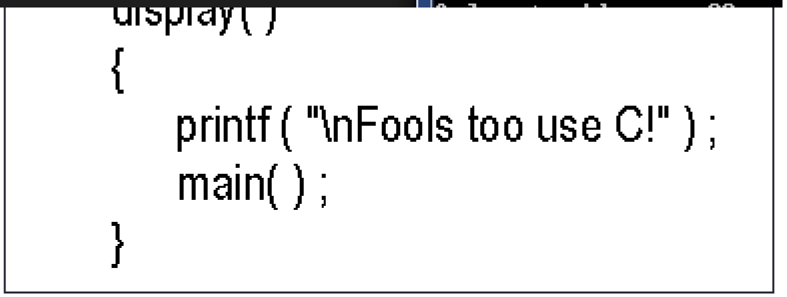
- What will be the output of the following programs



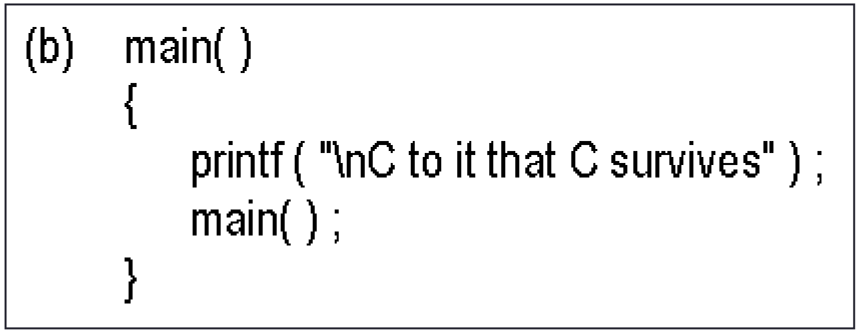
```
#include "stdio.h"

void display();

int main(){
    printf("#nOnly stupids use C?");
    main();
}
```



```
Only stupids use C?
Only stupids use C?
Only stupids use C?
Only stupids use C?
Only stupids use C?
Only stupids use C?
Only stupids use C?
Only stupids use C?
Only stupids use C?
```



```
display( )
{
    printf ( "\nFools too use C!" ) ;
    main( ) ;
}
```



```
(b)  main( )
     {
         printf ( "\nC to it that C survives" ) ;
         main( ) ;
     }
```

Exercise 3.

- What will be the output of the following programs

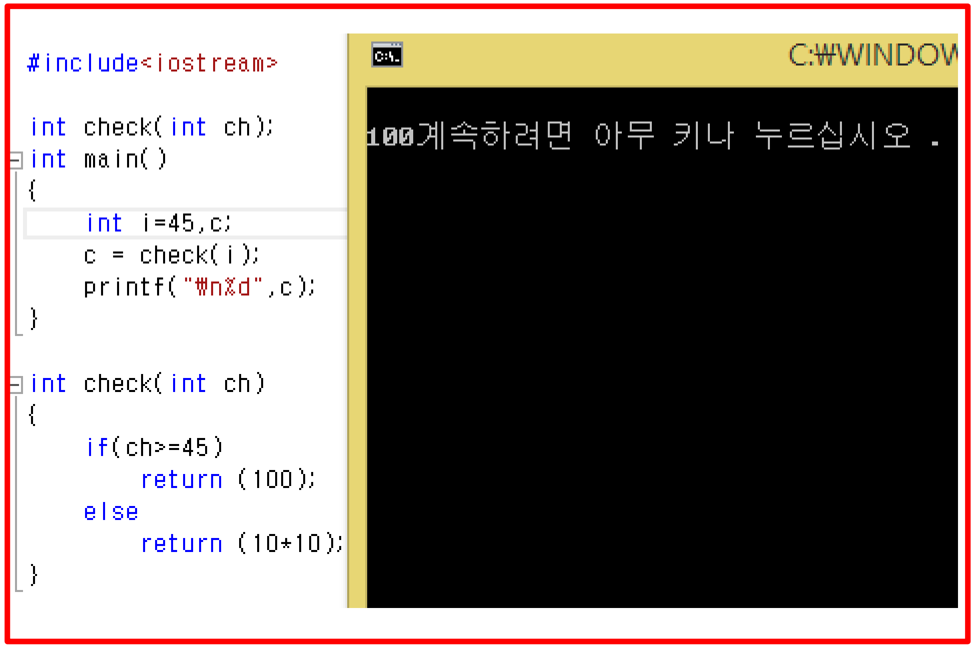
```
(c)   main( )
      {
          int  i = 45, c ;
          c = check ( i ) ;
          printf ( "\n%d", c ) ;
      }
      check ( int  ch )
      {
          if ( ch >= 45 )
              return ( 100 ) ;
          else
              return ( 10 * 10 ) ;
      }
```

```
(d)   main( )
      {
          int  i = 45, c ;
          c = multiply ( i * 1000 ) ;
          printf ( "\n%d", c ) ;
      }
      check ( int  ch )
      {
          if ( ch >= 40000 )
              return ( ch / 10 ) ;
          else
              return ( 10 ) ;
      }
```

Exercise 3.

- What will be the output of the following programs

```
(c)   main( )
      {
          int  i = 45, c ;
          c = check ( i ) ;
          printf ( "\n%d", c ) ;
      }
      check ( int  ch )
      {
          if ( ch >= 45 )
               return ( 100 ) ;
          else
               return ( 10 * 10 ) ;
      }
```



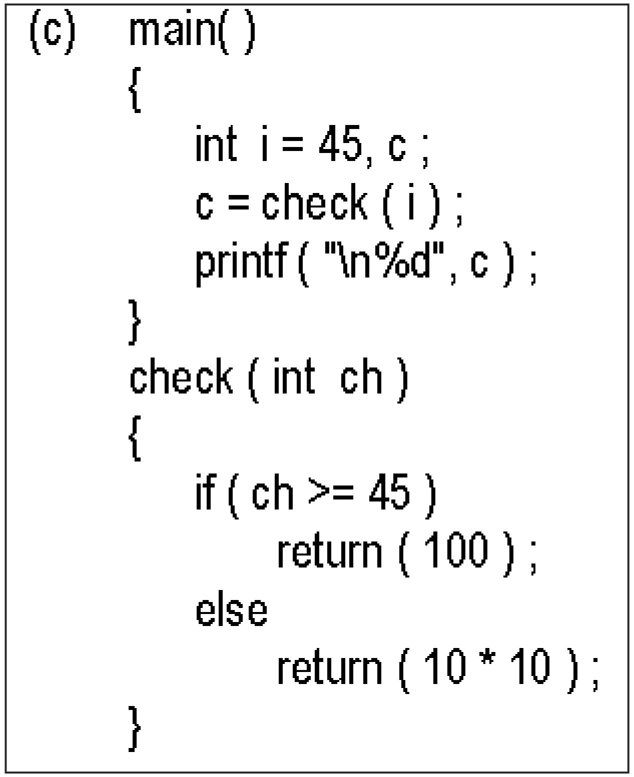
```
#include<iostream>

int check(int ch);
int main()
{
    int i=45,c;
    c = check(i);
    printf("₩n%d",c);
}

int check(int ch)
{
    if(ch>=45)
        return (100);
    else
        return (10*10);
}
```

C:₩WINDOW

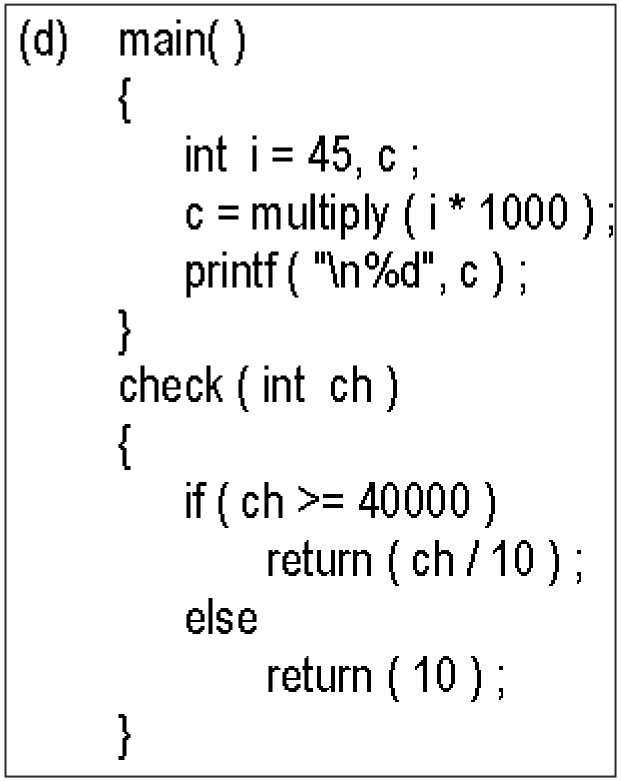
```
100계속하려면 아무 키나 누르십시오 .
```

Exercise 3.

- What will be the output of the following programs



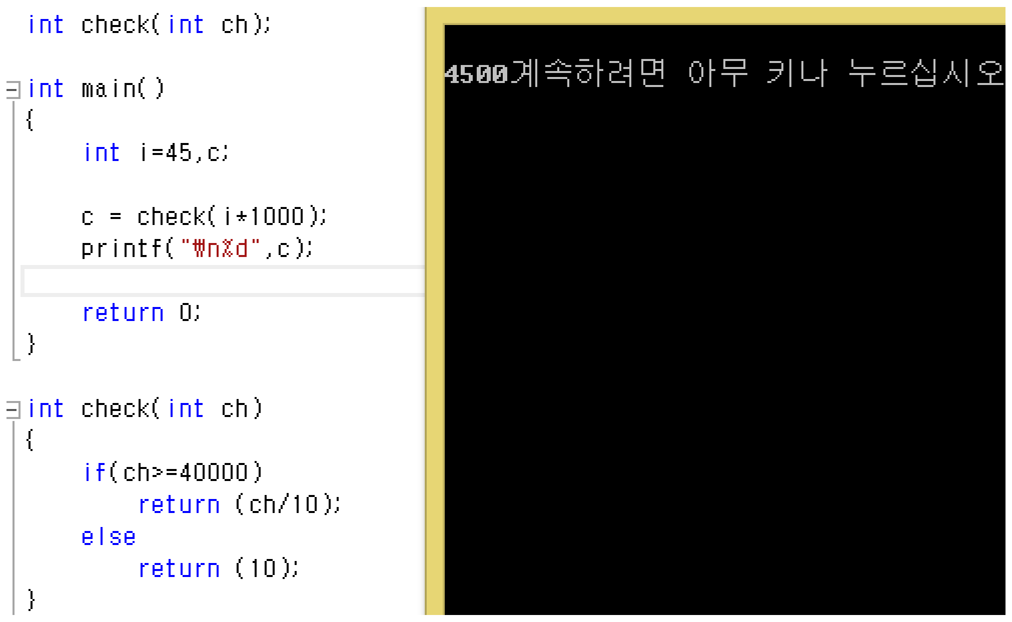
```
(c)   main( )
      {
          int  i = 45, c ;
          c = check ( i ) ;
          printf ( "\n%d", c ) ;
      }
      check ( int  ch )
      {
          if ( ch >= 45 )
              return ( 100 ) ;
          else
              return ( 10 * 10 ) ;
      }
```



```
(d)   main( )
      {
          int  i = 45, c ;
          c = multiply ( i * 1000 ) ;
          printf ( "\n%d", c ) ;
      }
      check ( int  ch )
      {
          if ( ch >= 40000 )
              return ( ch / 10 ) ;
          else
              return ( 10 ) ;
      }
```

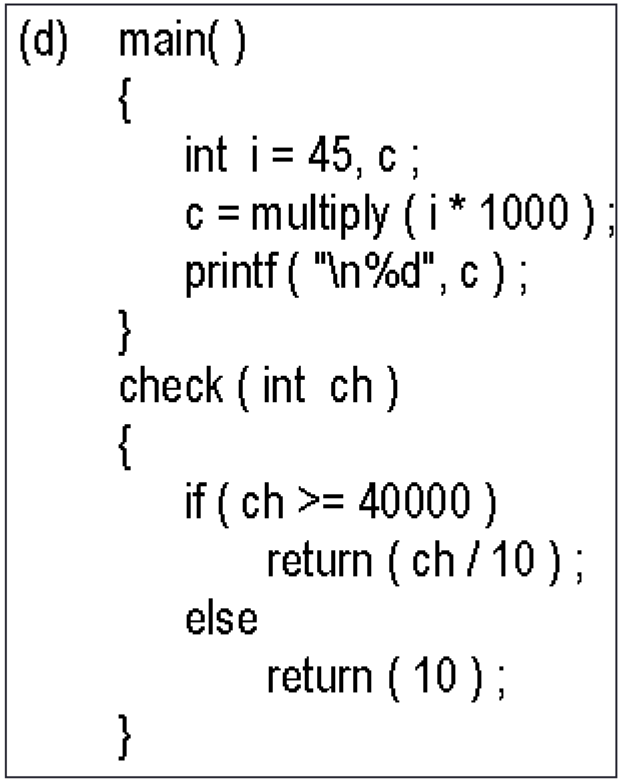
Exercise 3.

- What will be the output of the following programs



```
int check(int ch);

int main()
{
    int i=45,c;

    c = check(i*1000);
    printf("₩n%d",c);

    return 0;
}

int check(int ch)
{
    if(ch>=40000)
        return (ch/10);
    else
        return (10);
}
```

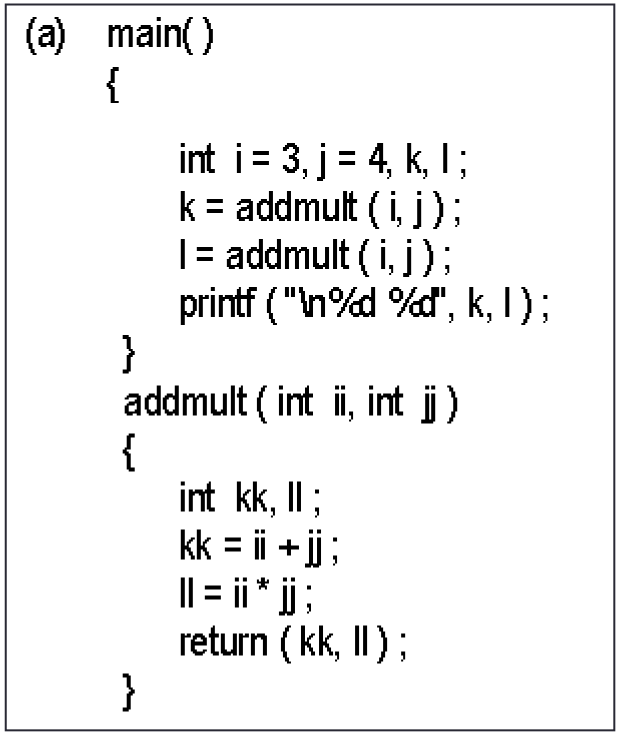
```
4500계속하려면 아무 키나 누르십시오
```



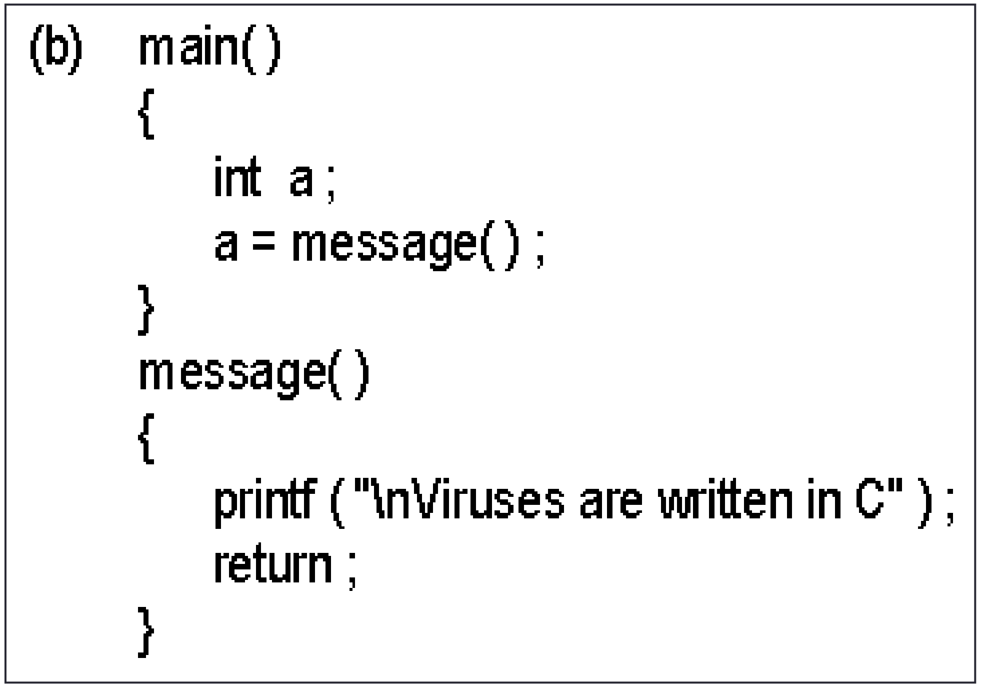
```
(d)   main( )
      {
          int  i = 45, c ;
          c = multiply ( i * 1000 ) ;
          printf ( "\n%d", c ) ;
      }
      check ( int  ch )
      {
          if ( ch >= 40000 )
              return ( ch / 10 ) ;
          else
              return ( 10 ) ;
      }
```

Exercise 4.

- Point out the errors, if any, in the following programs



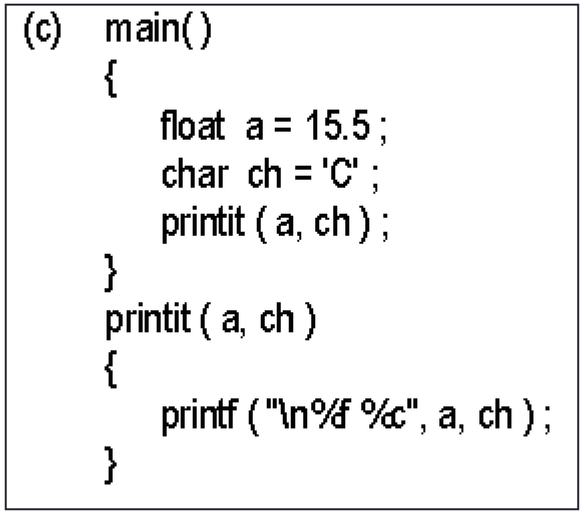
```
(a)   main( )
      {

          int  i = 3, j = 4, k, l ;
          k = addmult ( i, j ) ;
          l = addmult ( i, j ) ;
          printf ( "\n%d %d", k, l ) ;
       }
       addmult ( int  ii, int  jj )
       {
          int  kk, ll ;
          kk = ii + jj ;
          ll = ii * jj ;
          return ( kk, ll ) ;
       }
```



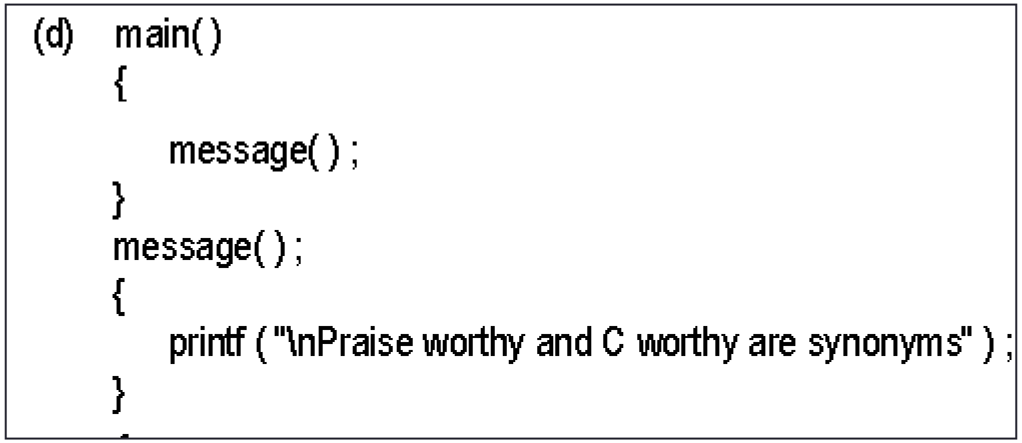
```
(b)   main( )
      {
          int  a ;
          a = message( ) ;
      }
      message( )
      {
          printf ( "\nViruses are written in C" ) ;
          return ;
      }
```

Exercise 4.

- Point out the errors, if any, in the following programs



```
(c)   main( )
      {
          float  a = 15.5 ;
          char  ch = 'C' ;
          printit ( a, ch ) ;
      }
      printit ( a, ch )
      {
          printf ( "\n%f %c", a, ch ) ;
      }
```

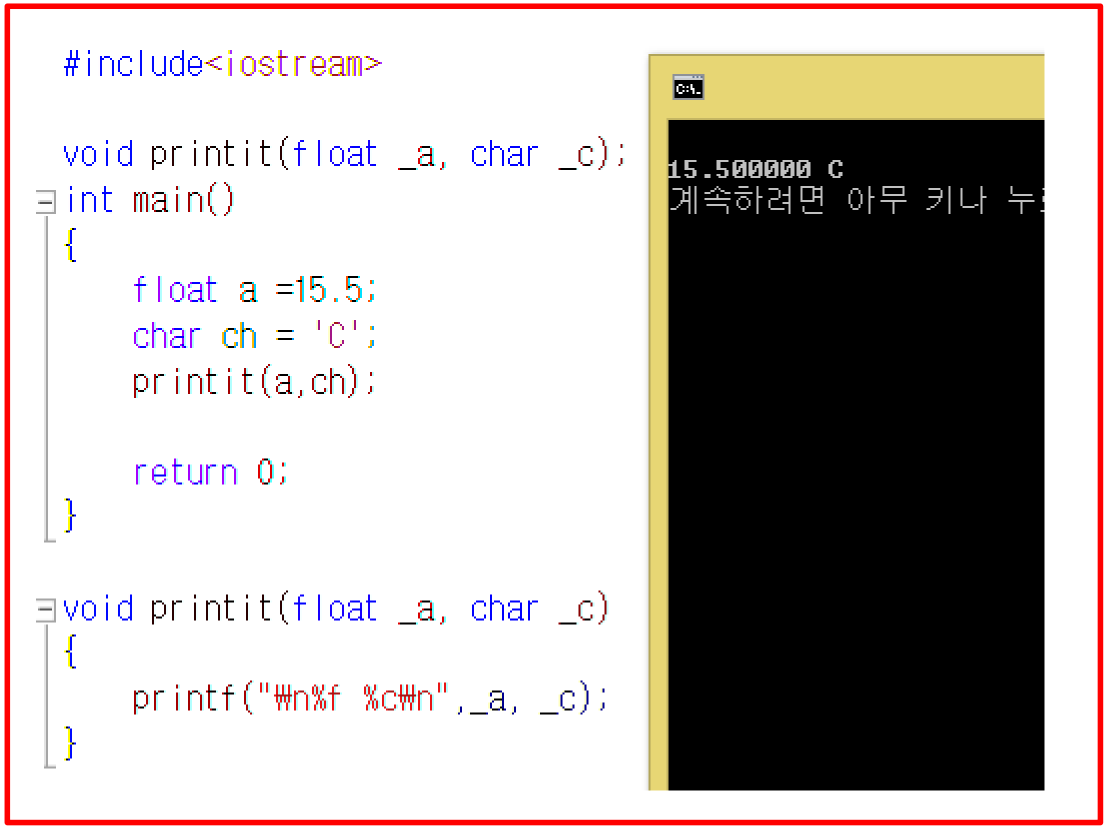


```
(d)   main( )
      {
          message( ) ;
      }
      message( ) ;
      {
          printf ( "\nPraise worthy and C worthy are synonyms" ) ;
      }
```

Exercise 4.

- Point out the errors, if any, in the following programs

```
(c)   main( )
      {
            float  a = 15.5 ;
            char  ch = 'C' ;
            printit ( a, ch ) ;
      }
      printit ( a, ch )
      {
            printf ( "\n%f %c", a, ch ) ;
      }
```

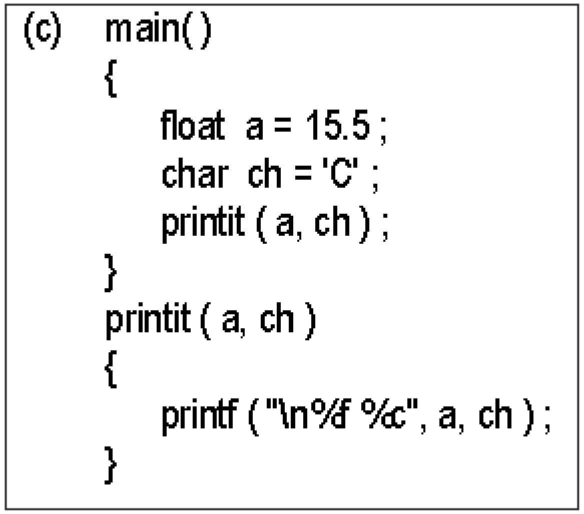


```
#include<iostream>

void printit(float _a, char _c);
int main()
{
    float a =15.5;
    char ch = 'C';
    printit(a,ch);

    return 0;
}

void printit(float _a, char _c)
{
    printf("₩n%f %c₩n",_a, _c);
}
```

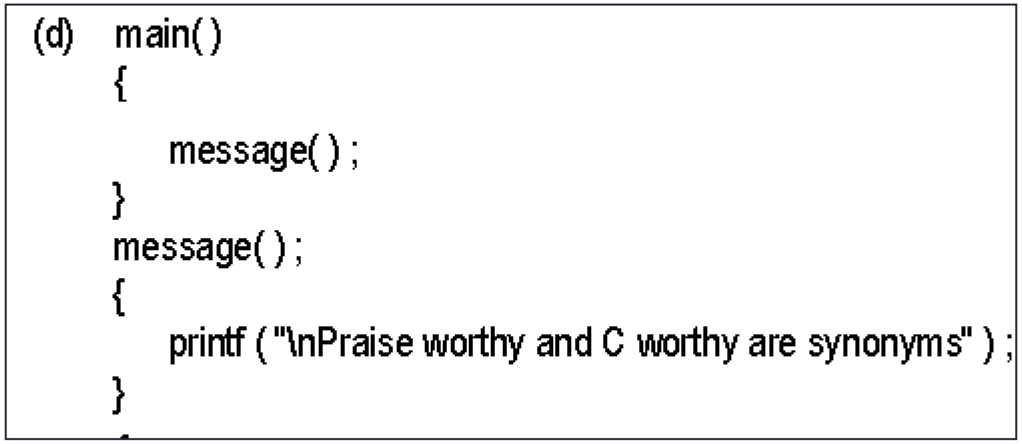
```
15.500000 C
계속하려면 아무 키나 누
```

Exercise 4.

- Point out the errors, if any, in the following programs



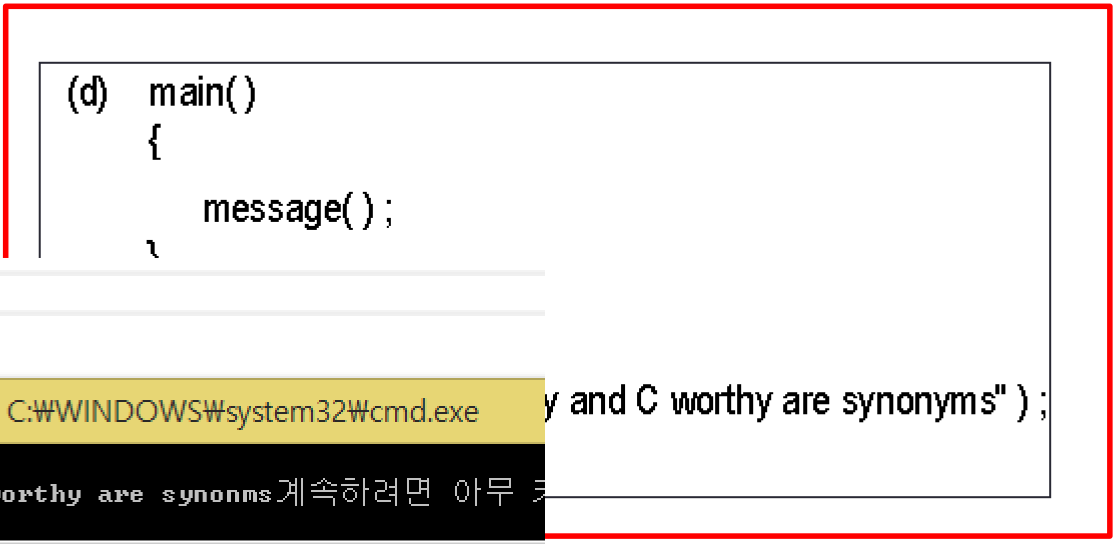
```
(c)   main( )
      {
          float  a = 15.5 ;
          char  ch = 'C' ;
          printit ( a, ch ) ;
      }
      printit ( a, ch )
      {
          printf ( "\n%f %c", a, ch ) ;
      }
```



```
(d)   main( )
      {
          message( ) ;
      }
      message( ) ;
      {
          printf ( "\nPraise worthy and C worthy are synonyms" ) ;
      }
```

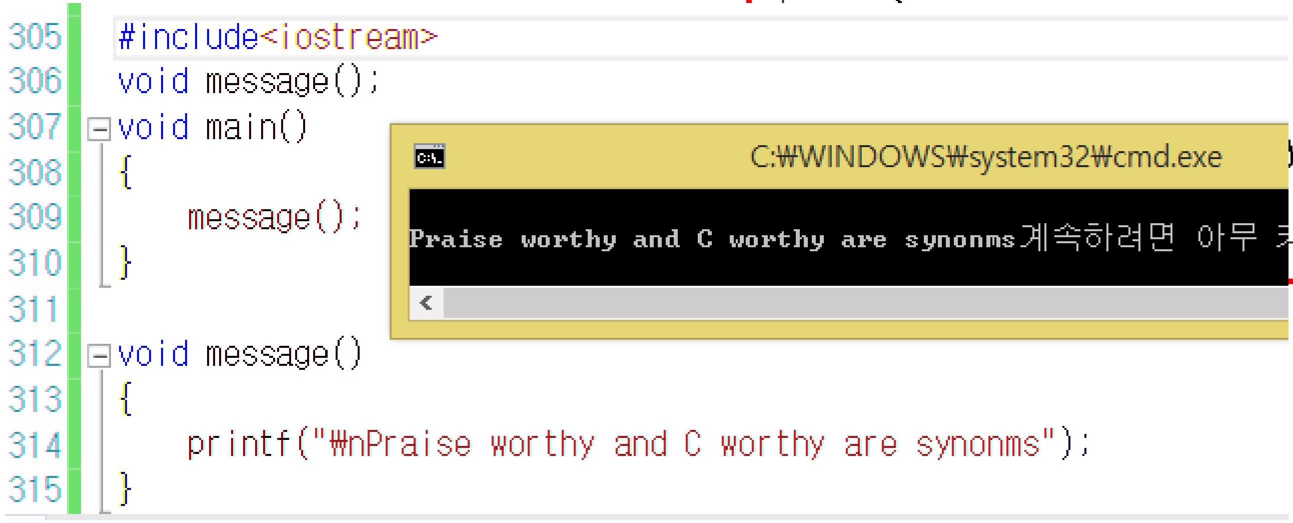
Exercise 4.

- Point out the errors, if any, in the following programs



```
(d)  main( )
     {
         message( ) ;
     }
```

```
y and C worthy are synonyms" ) ;
```



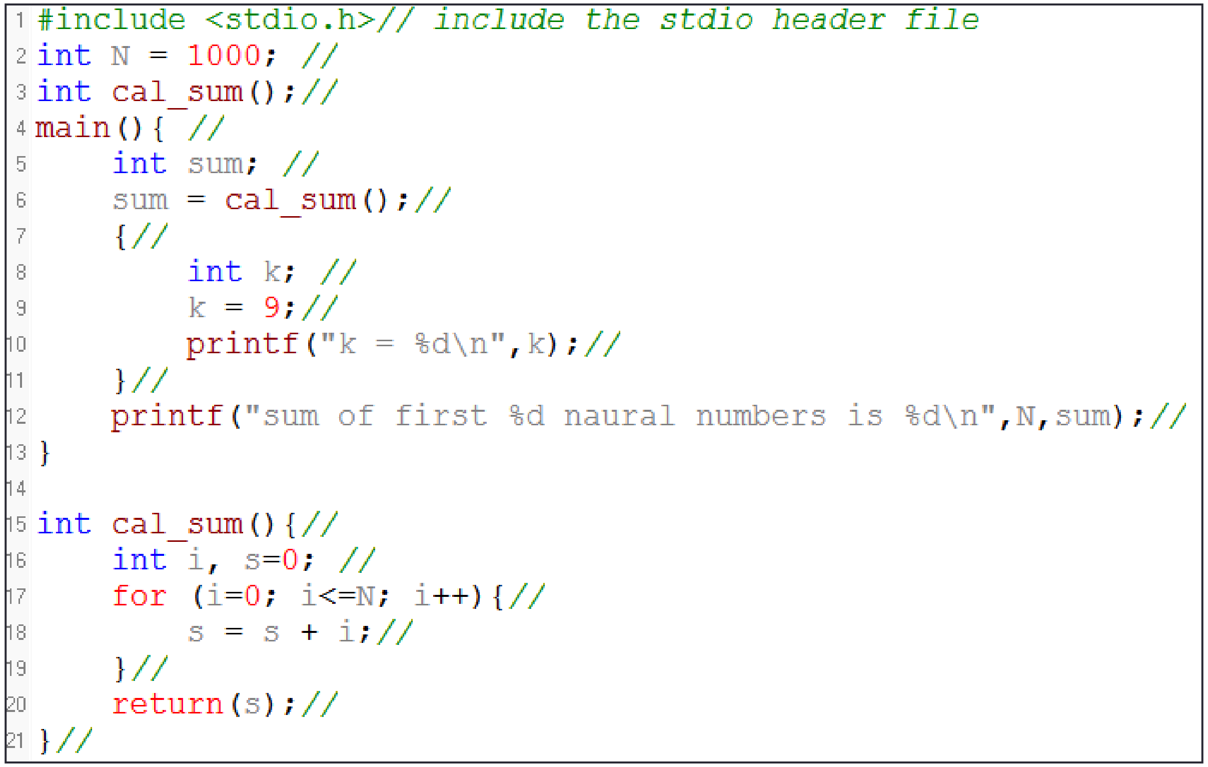
```
#include<iostream>
void message();
void main()
{
    message();
}

void message()
{
    printf("₩nPraise worthy and C worthy are synonms");
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Praise worthy and C worthy are synonms계속하려면 아무 
```

HW #6-2

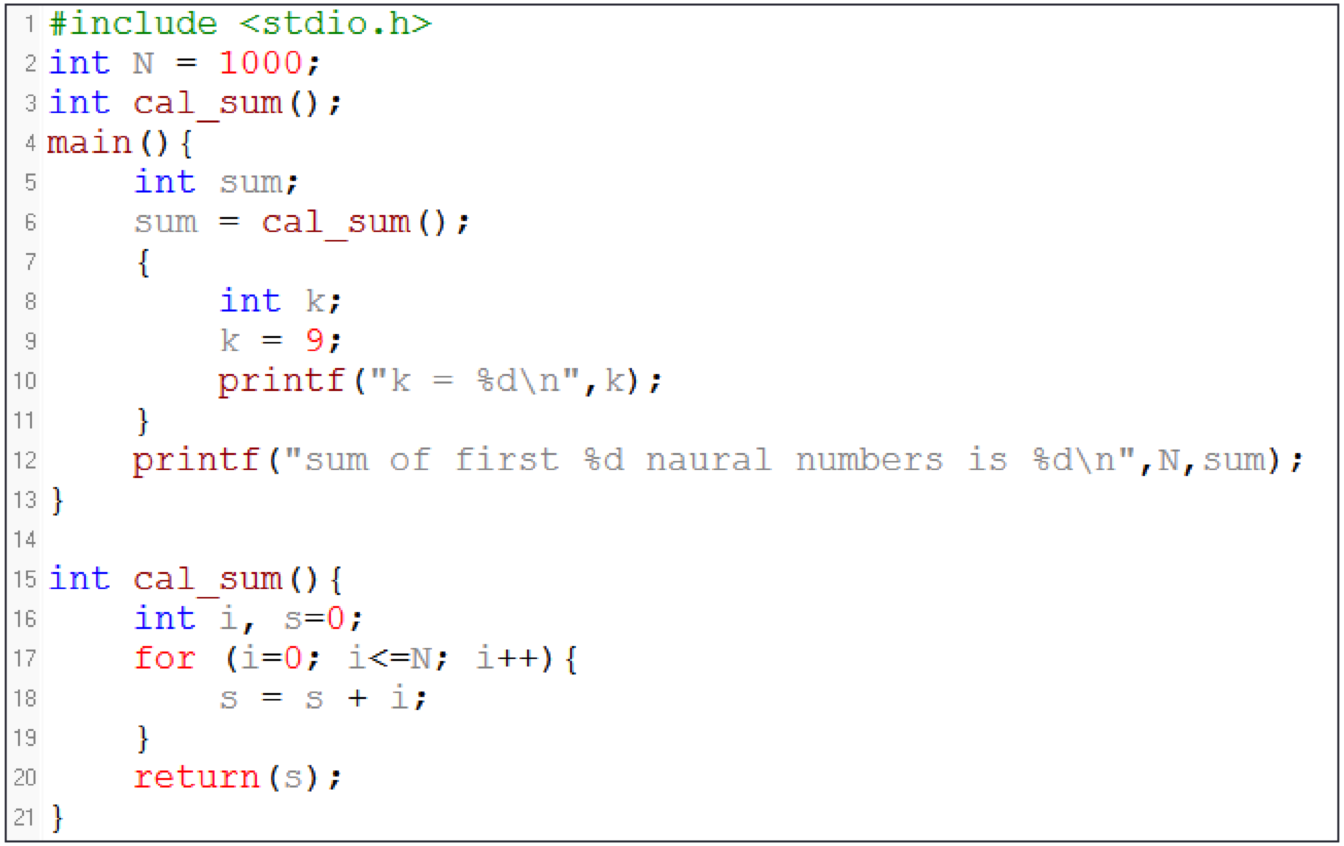
1. Please complete the comment for the following source code like first line (describe the meaning of each statement)



```c
#include <stdio.h>// include the stdio header file
int N = 1000; //
int cal_sum();//
main(){ //
    int sum; //
    sum = cal_sum();//
    {//
        int k; //
        k = 9;//
        printf("k = %d\n",k);//
    }//
    printf("sum of first %d naural numbers is %d\n",N,sum);//
}

int cal_sum(){//
    int i, s=0; //
    for (i=0; i<=N; i++){//
        s = s + i;//
    }//
    return(s);//
}//
```

HW #6-2

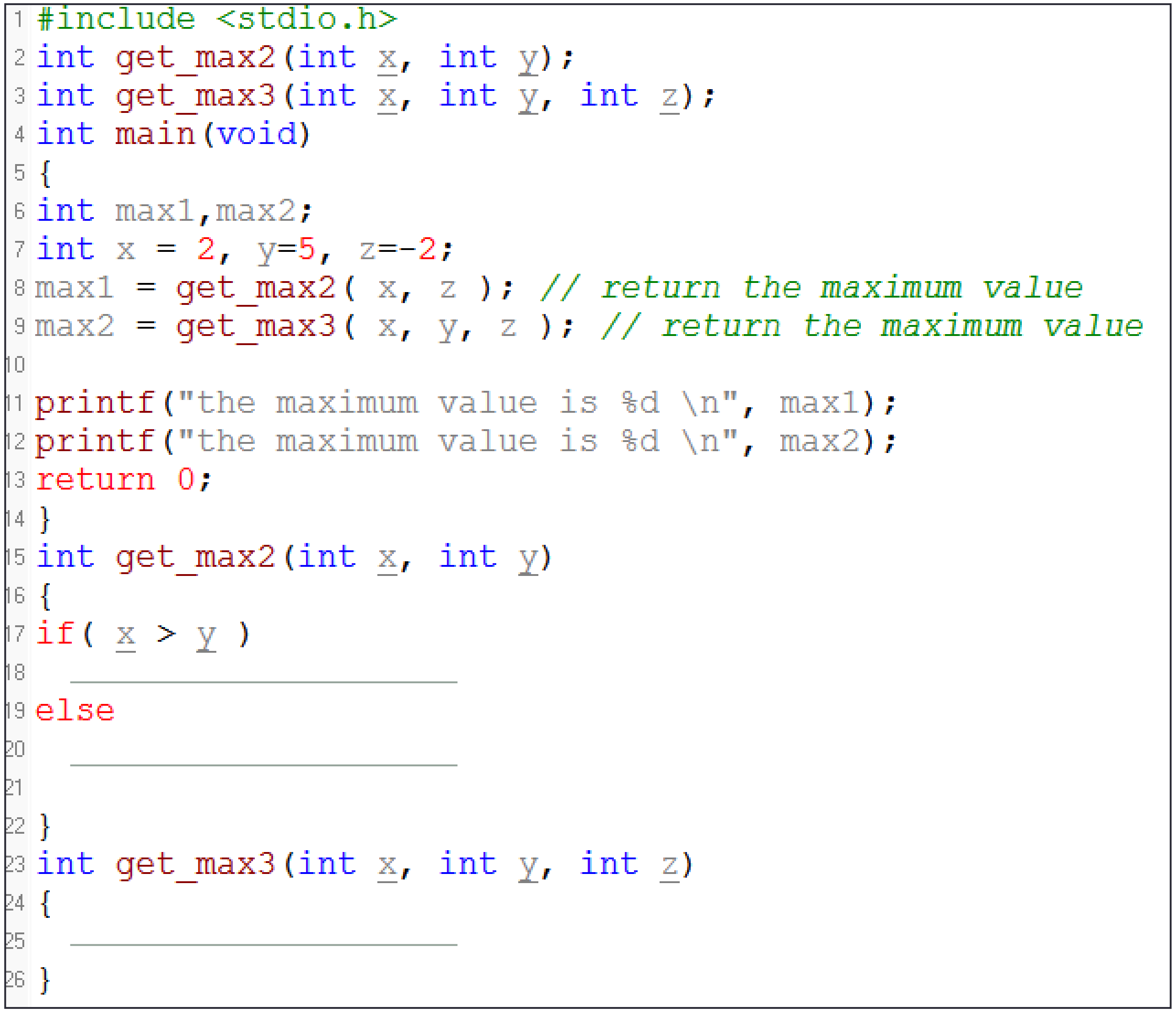
2. How many variables in the following program?
   - ✓ Draw the scope of each variable.



```c
#include <stdio.h>
int N = 1000;
int cal_sum();
main(){
    int sum;
    sum = cal_sum();
    {
        int k;
        k = 9;
        printf("k = %d\n",k);
    }
    printf("sum of first %d naural numbers is %d\n",N,sum);
}

int cal_sum(){
    int i, s=0;
    for (i=0; i<=N; i++){
        s = s + i;
    }
    return(s);
}
```

HW #6-2

3. Complete the following functions to get the maximum value.



```c
#include <stdio.h>
int get_max2(int x, int y);
int get_max3(int x, int y, int z);
int main(void)
{
int max1,max2;
int x = 2, y=5, z=-2;
max1 = get_max2( x, z ); // return the maximum value
max2 = get_max3( x, y, z ); // return the maximum value

printf("the maximum value is %d \n", max1);
printf("the maximum value is %d \n", max2);
return 0;
}
int get_max2(int x, int y)
{
if( x > y )
    ____________________
else
    ____________________

}
int get_max3(int x, int y, int z)
{
    ____________________
}
```

HW #6-2

4. 아래의 기능을 수행할 수 있는 함수 sum을 구현하시오.(하나의 함수에서 아래의 기능을 수행해야 함)
   - 1부터 n까지의 합 (n은 integer 타입의 변수)
   - 1부터 n까지의 짝수의 합
   - 1부터 n까지의 홀수의 합