

5. 함수

🔄 함수란?

함수란 특정한 명령을 수행하는 하나의 독립된 프로그램을 말한다. 어떠한 기능을 함수로 구현하면 원하는 시점에 해당 함수를 호출하여 기능을 사용 할 수 있게 된다. 재사용할 필요가 있는 기능은 함수로 구현해두면 편한 호출이 가능해 진다.

🔄 함수의 모양

함수는 선언부와 정의부로 나누어 진다. 함수가 호출되기 전 미리 함수의 내용을 정의해 놓는다면 함수의 선언부는 쓰지 않아도 된다. 단, 함수가 호출 되는 부분 이후에 함수를 정의하고자 한다면 반드시 함수가 호출되는 부분 이전에 함수의 선언부를 기재해야 한다.

함수의 선언(함수의 프로토 타입, 함수의 원형 이라고도 한다.)

①return type ②함수명 (③매개변수 리스트 ...);

함수의 정의

```
①return type ②함수명 ( ③매개변수 리스트 ... ) {  
    변수선언;  
    실행하려는 명령;  
    ④return 값;  
}
```

①return type : 함수의 실행이 종료된 이후 함수의 실행 결과로 나올 값의 자료형을 쓴다. 함수의 실행 결과는 return 문을 이용해 기재할 수 있으며 이 값과 return type은 항상 같은 자료형 이어야 한다. 함수의 실행 결과로 아무런 결과도 나오지 않을 경우에는 return type에 void를 쓴다.

②함수명 : 함수명 작성 규칙은 변수명 작성규칙과 동일하다. 프로그래머가 원하는 이름을 붙이 되 함수의 이름으로 함수의 기능을 예측할 수 있도록 지어주는 것이 좋다.

③매개변수 리스트 : 함수가 호출되면서 받아올 초기값을 저장할 변수를 자료형과 같이 기재한다. 매개변수의 개수는 여러 개가 될 수 있으며 , 로 구분해서 기재한다. 매개변수는 함수가 받아올 초기값이 되며 해당 값이 함수에 올바르게 넘어오지 않으면 함수가 실행되지 않는다. 함수의 선언부에는 매개변수명 없이 매개변수의 자료형만 기재하는 것도 가능하다.

④return 값 : return문은 함수를 종료시키기 위해 사용하지만 원하는 값을 함수의 실행 결과로 돌려주는 용도로도 사용 할 수 있다. return 뒤쪽에 어떠한 값을 쓰면 그 값이 함수의 실행 결과 값이 되며 함수의 실행결과는 함수가 호출된 곳으로 되돌아간다. 또한 함수의 실행 결과 값의 자료형은 함수의 return type과 반드시 일치해야 한다.

🔄 함수의 작성방법

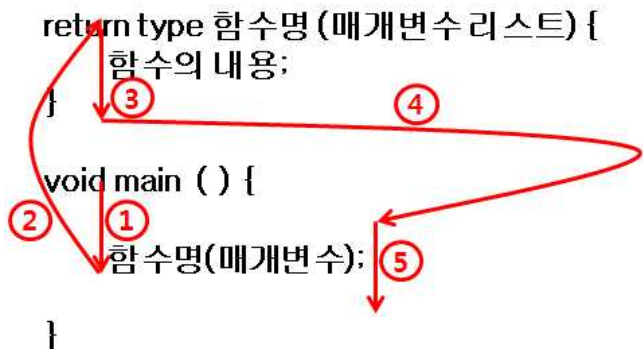
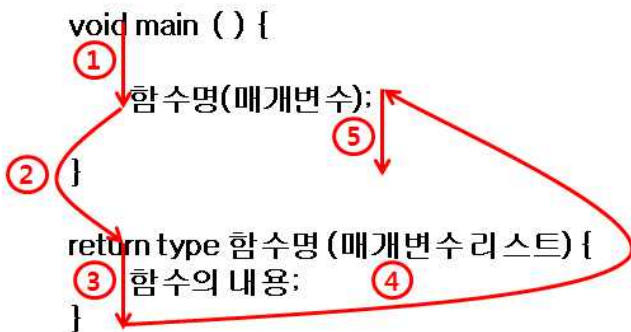
함수는 ①선언-호출-정의 와 ②정의-호출의 두 가지 방법으로 만들어 사용 할 수 있다.

방법 ①의 형태	방법 ②의 형태
<pre>return type 함수명 (매개변수 리스트); void main () { 함수명(매개변수); } return type 함수명 (매개변수 리스트) { 함수의 내용; }</pre>	<pre>return type 함수명 (매개변수 리스트) { 함수의 내용; } void main () { 함수명(매개변수); }</pre>

🔄 함수의 동작 형태

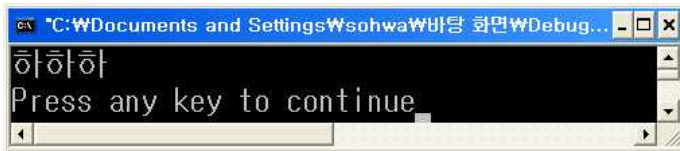
함수는 호출되면 함수의 정의부로 이동해 함수의 모든 내용을 실행 한 뒤 함수가 종료하면 호출되었던 곳으로 돌아온다. 함수를 호출할 때는 "함수명(매개변수)" 의 형태로 호출한다. 매개변수는 호출된 함수에 건네줄 초기 값을 의미하며 함수의 선언부와 정의부에 있는 매개변수의 자료형과 같은 값을 매개변수로 건네주어야만 호출이 가능해 진다.

return type 함수명 (매개변수리스트);



예제 5-1 : 함수를 만드는 방법 - 1

```
#include <stdio.h>
void func( ) {           // 함수의 정의
    printf("하하하\n");
}
void main ( ) {
    func( );             // 함수의 호출
}
```



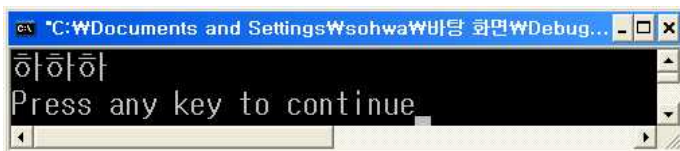
예제 5-2 : 함수를 만드는 방법 - 2

```
#include <stdio.h>

void func( );           // 함수의 선언

void main ( ) {
    func( );           // 함수의 호출
}

void func( ) {          // 함수의 정의
    printf("하하하\n");
}
```

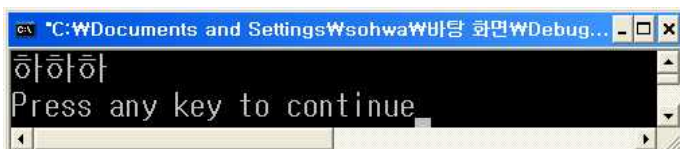


예제 5-3 : 함수의 선언부가 없어서 에러가 발생하는 예제

```
#include <stdio.h>

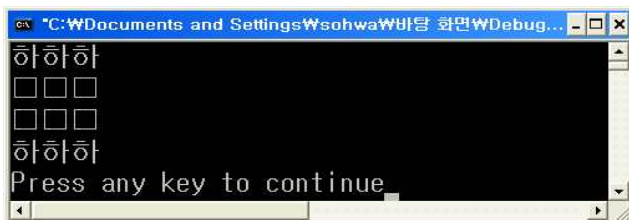
void main ( ) {
    func( );           // 함수의 호출
}

void func( ) {          // 함수의 정의
    printf("하하하\n");
}
```



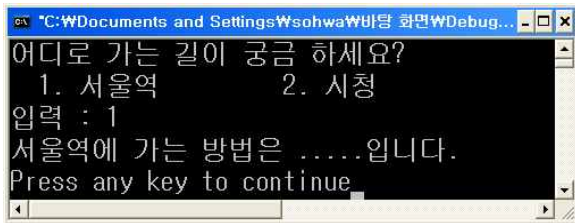
예제 5-4 : 함수의 호출 방법과 동작 형태

```
#include <stdio.h>
void haha( ) {           // 함수의 정의
    printf("하하하\n");
}
void nemo( ) {
    int x;
    for ( x = 1; x <= 3; x++ ) {
        printf("□");
    }
    printf("\n");
}
void main ( ) {
    haha( );              // 함수의 호출
    nemo( );              // 함수의 호출
    nemo( );              // 함수의 호출
    haha( );              // 함수의 호출
}
```



예제 5-5 : 입력에 따른 함수의 호출

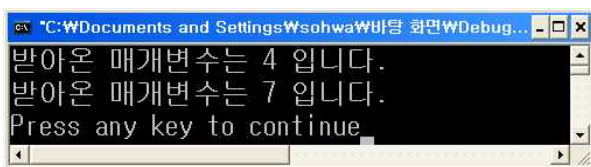
```
#include <stdio.h>
void ss( ) {             // 함수의 정의
    printf("서울역에 가는 방법은 .....입니다.\n");
}
void sh( ) {             // 함수의 정의
    printf("시청역에 가는 방법은 .....입니다.\n");
}
void main ( ) {
    int a;
    printf("어디로 가는 길이 궁금 하세요?\n");
    printf(" 1. 서울역          2. 시청  \n");
    printf("입력 : ");
    scanf("%d", &a);
    if(a == 1) {
        ss( );
    }
    else if( a == 2 ) {
        sh( );
    }
}
```



예제 5-6 : 매개변수를 받아오는 함수

```
#include <stdio.h>
void func( int x ) {          // 매개변수로 정수를 받아와 x에 보관시킨 뒤 동작하는 함수
    printf("받은 매개변수는 %d 입니다.\n", x);
}

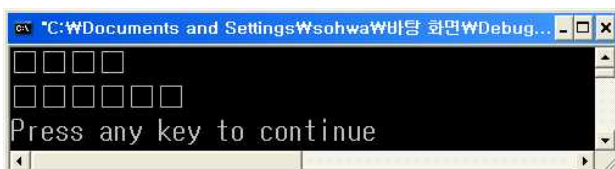
void main ( ) {
    func(4);    // func함수는 int를 매개변수로 받아와야 함으로 함수를 호출하면서
    func(7);    // int값을 넣어 호출될 함수에 전달해 준다.
}
```



예제 5-7 : 매개변수를 받아와 받아온 매개변수에 따라 동작이 달라지는 함수

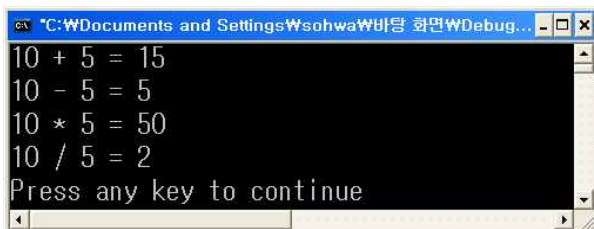
```
#include <stdio.h>
void prn( int x ) {          // 매개변수로 정수를 받아와 x에 보관시킨 뒤 동작하는 함수
    int a;
    for( a = 0; a <= x; a++ ) {
        printf("□");
    }
    printf("\n");
}

void main ( ) {
    prn(3);    // prn 함수는 int를 매개변수로 받아와야 함으로 함수를 호출하면서
    prn(5);    // int값을 넣어 호출될 함수에 전달해 준다.
}
```



예제 5-8 : 두 개의 매개변수를 받아오는 함수

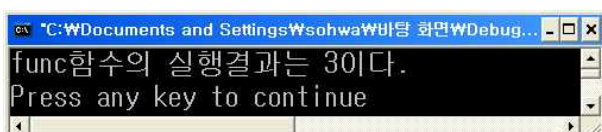
```
#include <stdio.h>
void plus( int x, int y ) {          // 두 개의 매개변수를 받아와 x와 y에 보관한 뒤 시작
    printf("%d + %d = %d\n", x, y, x + y);
}
void minus( int x, int y ) {         // 두 개의 매개변수를 받아와 x와 y에 보관한 뒤 시작
    printf("%d - %d = %d\n", x, y, x - y);
}
void mul( int x, int y ) {           // 두 개의 매개변수를 받아와 x와 y에 보관한 뒤 시작
    printf("%d * %d = %d\n", x, y, x * y);
}
void div( int x, int y ) {           // 두 개의 매개변수를 받아와 x와 y에 보관한 뒤 시작
    printf("%d / %d = %d\n", x, y, x / y);
}
void main ( ) {
    int a, b;
    a = 10;
    b = 5;
    plus(a, b);
    minus(a, b);
    mul(a, b);
    div(a, b);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
Press any key to continue
```

예제 5-9 : 실행결과로 특정한 값을 return 하는 함수 - 1

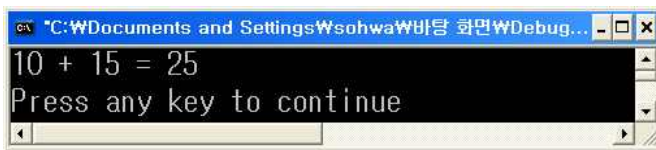
```
#include <stdio.h>
int func( ) {
    return 3;
}
void main ( ) {
    int a;
    a = func( );
    printf("func함수의 실행결과는 %d이다.\n", a);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
func함수의 실행결과는 3이다.
Press any key to continue
```

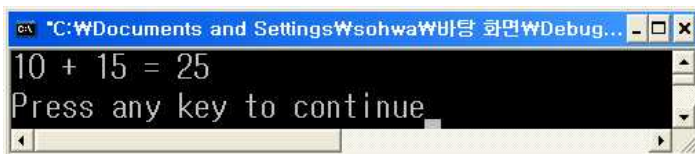
예제 5-10 : 실행결과로 특정한 값을 return 하는 함수 - 2

```
#include <stdio.h>
int plus( ) {           // 실행결과로 int가 나오는 plus함수
    int x = 10, y = 15;
    printf("%d + %d = ", x, y);
    return x + y;
}
void main ( ) {
    int a;
    a = plus( );
    printf("%d\n", a);
}
```



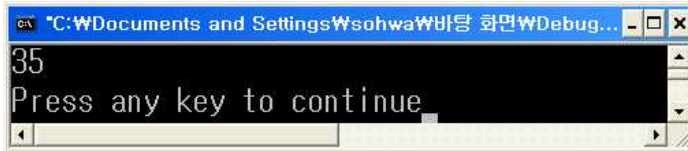
예제 5-11: 매개변수를 받아가고 return도 하는 함수 - 1

```
#include <stdio.h>
int plus( int x, int y) {           // 실행결과로 int가 나오는 plus함수
    printf("%d + %d = ", x, y);
    return x + y;
}
void main ( ) {
    int a;
    a = plus( 10 , 15 );
    printf("%d\n", a);
}
```



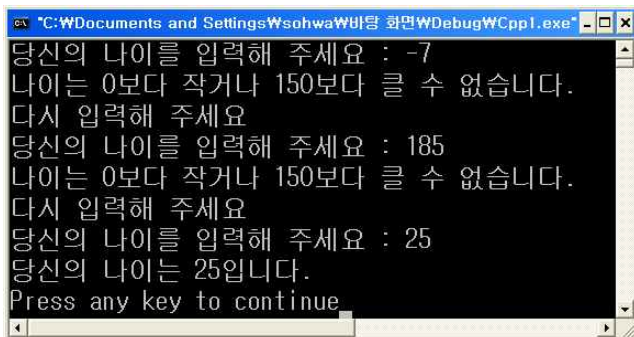
예제 5-12: 매개변수로 받아간 값의 절대값을 구해주는 함수

```
#include <stdio.h>
int abs( int x ) {
    if(x < 0) {
        x = -x;
    }
    return x;
}
void main ( ) {
    int a = -35;
    a = abs(a);
    printf("%d\n", a);
}
```



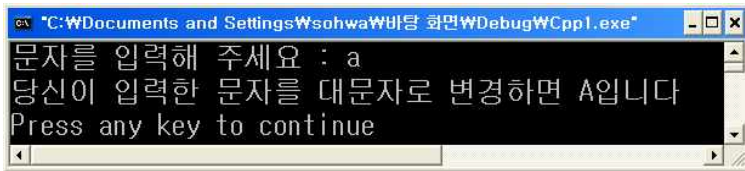
예제 5-13: 입력받은 값을 함수를 이용해 검사하고 함수의 return값으로 재입력을 결정하는 예제

```
#include <stdio.h>
int check( int x ) {
    if( x < 0 || x >= 150 ) {
        printf("나이는 0보다 작거나 150보다 클 수 없습니다. \n");
        printf("다시 입력해 주세요\n");
        return 1;
    }
    return 0;
}
void main ( ) {
    int a;
    do {
        printf("당신의 나이를 입력해 주세요 : ");
        scanf("%d", &a);
    }while( check(a) );
    printf("당신의 나이는 %d입니다. \n", a);
}
```



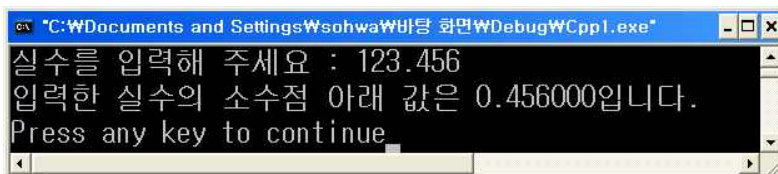
예제 5-14: 입력받은 문자를 대문자로 변경해주는 함수

```
#include <stdio.h>
char dae( char x ) {
    if( x >= 97 ) {
        x = x - 32;
    }
    return x;
}
void main ( ) {
    char a;
    printf("문자를 입력해 주세요 : ");
    scanf("%c", &a);
    printf("당신이 입력한 문자를 대문자로 변경하면 %c입니다\n", dae(a) );
}
```

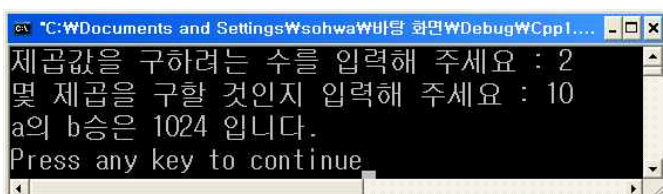
예제 5-15: 매개변수로 받아간 실수의 소수점 이하 값을 구해 리턴해주는 함수

```
#include <stdio.h>
double under( double x ) {
    int y;
    y = x;
    x = x - y;
    return x;
}
void main ( ) {
    double a;
    printf("실수를 입력해 주세요 : ");
    scanf("%lf", &a);
    a = under(a);
    printf("입력한 실수의 소수점 아래 값은 %lf입니다. \n", a);
}
```



예제 5-16: 매개변수 x와 y를 받아가 x의 y승 값을 구해 리턴하는 함수

```
#include <stdio.h>
int pow( int x, int y ) {
    int z = 1, dap = 1;
    for( z = 1; z <= y ; z++ ) {
        dap = dap * x;
    }
    return dap;
}
void main ( ) {
    int a, b;
    printf("제곱값을 구하려는 수를 입력해 주세요 : ");
    scanf("%d", &a);
    printf("몇 제곱을 구할 것인지 입력해 주세요 : ");
    scanf("%d", &b);
    printf("a의 b승은 %d 입니다.\n", pow(a, b)) ;
}
```



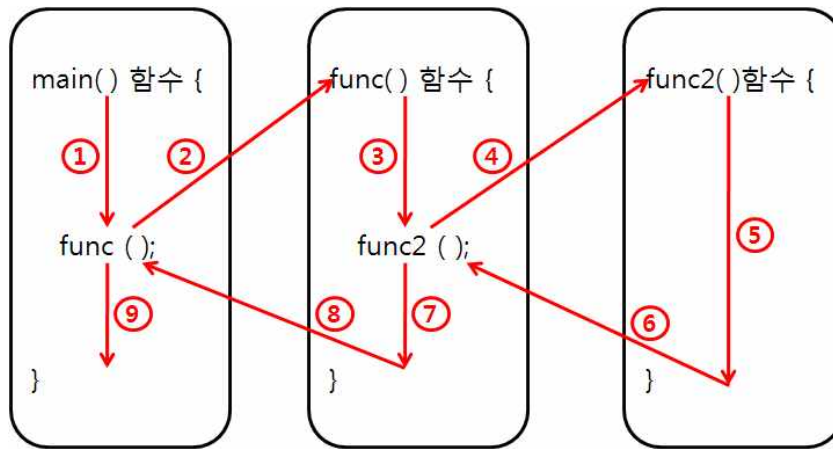
🔄 함수의 작동원리

함수는 현재 수행중인 프로그램과는 별도의 위치에 있는 명령어들의 집합이기 때문에 호출되면 함수 코드가 있는 부분으로 이동하여 프로그램이 실행된다. 따라서 함수가 종료하면 원위치로 복귀 하여야 하며 이런 복귀 주소들은 전부 메모리에 보관된다.

함수의 동작을 위한 정보가 보관되는 장소를 "시스템스택" 이라 한다. 시스템스택에는 함수의 동작을 위해 필요한 모든 정보가 보관된다. 함수의 실행을 위해 필요한 정보에는 함수의 매개변수, 함수의 지역변수, 함수의 복귀주소 등이 있다.

※ stack이란 나중에 들어간 정보가 먼저 나오는 자료보관형태를 말한다.

함수의 호출과 복귀에 따른 프로그램 실행 순서 (함수의 호출 및 종료 순서)

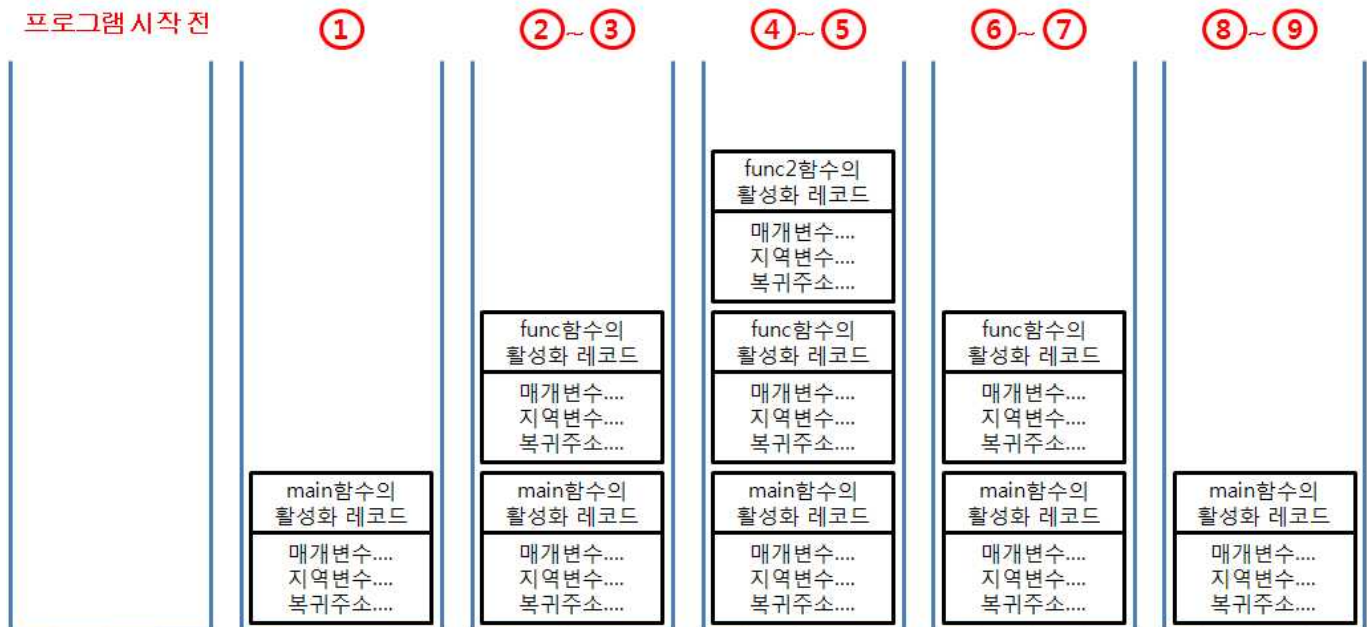


위의 그림에서 함수의 호출 순서는 main ⇨ func ⇨ func2 순이지만

함수의 종료 순서는 func2 ⇨ func ⇨ main 순서로 종료되게 된다.

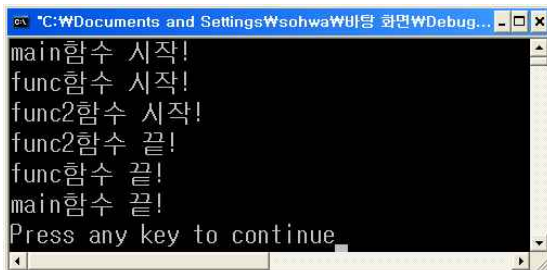
함수가 호출될 경우 시스템 스택의 동작 모양

프로그램 시작 전



예제 5-17: 함수의 호출과 종료 순서

```
#include <stdio.h>
void func( );
void func2( );
void main ( ) {
    printf("main함수 시작!\n");
    func( );
    printf("main함수 끝!\n");
}
void func( ) {
    printf("func함수 시작!\n");
    func2( );
    printf("func함수 끝!\n");
}
void func2( ) {
    printf("func2함수 시작!\n");
    printf("func2함수 끝!\n");
}
```



예제 5-18: 타 함수에 있는 변수는 사용하거나 호출 할 수 없다.

```
#include <stdio.h>
void func( ) {
    printf("a = %d\n", a);    // a는 main함수에서 만들었으므로 호출 할 수 없다.
}
void main ( ) {
    int a = 10;
    func( );
}
```

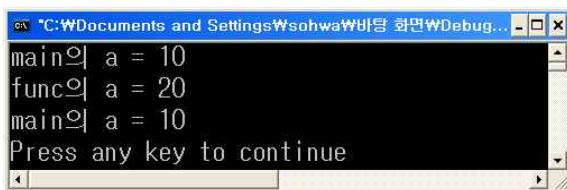
```
#include <stdio.h>
void func( ) {
    int a = 10;
}
void main ( ) {
    func( );
    printf("a = %d\n", a);    // func 함수가 종료 되었으므로 a가 사라지고 없다.
}
```

C:\Documents and Settings\Wsohwa\바탕 화면\Cpp1.cpp(3) : error C2065: 'a' : undeclared identifier
Error executing cl.exe.

Cpp1.exe - 1 error(s), 0 warning(s)

예제 5-19: 각 함수의 변수는 서로 다른 변수임을 확인하는 예제

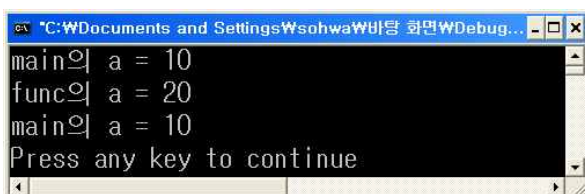
```
#include <stdio.h>
void func( ) {
    int a = 20;
    printf("func의 a = %d\n", a);
}
void main ( ) {
    int a = 10;
    printf("main의 a = %d\n", a);
    func( );
    printf("main의 a = %d\n", a);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
main의 a = 10
func의 a = 20
main의 a = 10
Press any key to continue
```

예제 5-20: 매개변수도 호출된 함수에서 별도로 만들어 지는 변수임을 확인하는 예제

```
#include <stdio.h>
void func( int a ) {
    a = 20;
    printf("func의 a = %d\n", a);
}
void main ( ) {
    int a = 10;
    printf("main의 a = %d\n", a);
    func( a );
    printf("main의 a = %d\n", a);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
main의 a = 10
func의 a = 20
main의 a = 10
Press any key to continue
```

예제 5-21: 호출된 함수에서 main함수의 변수값을 변경하는 방법에 대한 예제

```
#include <stdio.h>
int func( int a ) {
    a = 20;
    printf("func의 a = %d\n", a);
    return a;
}
void main ( ) {
    int a = 10;
    printf("main의 a = %d\n", a);
    a = func( a );
    printf("main의 a = %d\n", a);
}
```

```

C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
main의 a = 10
func의 a = 20
main의 a = 20
Press any key to continue

```

매개변수에 따른 함수 호출의 세 가지 형태

Call by value : 매개변수로 값만 전달하는 방식, 호출된 함수에서 원 함수의 변수 값을 변경 할 수 없다.

예제 5-22: 매개변수를 값으로 전달한 예제

```

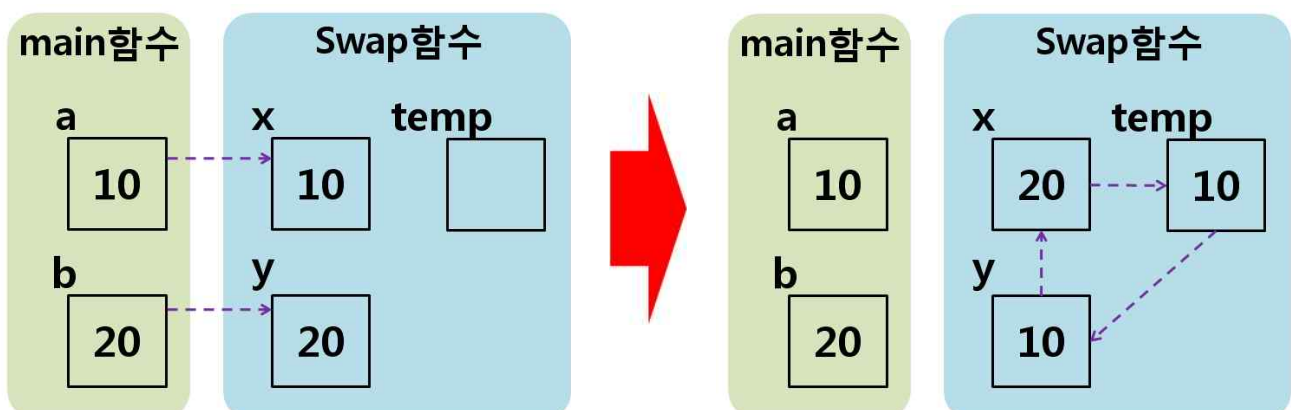
#include <stdio.h>
void swap( int x, int y ) {
    int temp = x;
    x = y;
    y = temp;
    printf("swap함수 내에서의 x = %d, y = %d \n", x, y);
}
void main ( ) {
    int a = 10, b = 20;
    printf("함수 호출 전 a = %d, b = %d \n", a, b);
    swap( a , b );
    printf("함수 호출 후 a = %d, b = %d \n", a, b);
}

```

```

C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
함수 호출 전 a = 10, b = 20
swap함수 내에서의 x = 20, y = 10
함수 호출 후 a = 10, b = 20
Press any key to continue

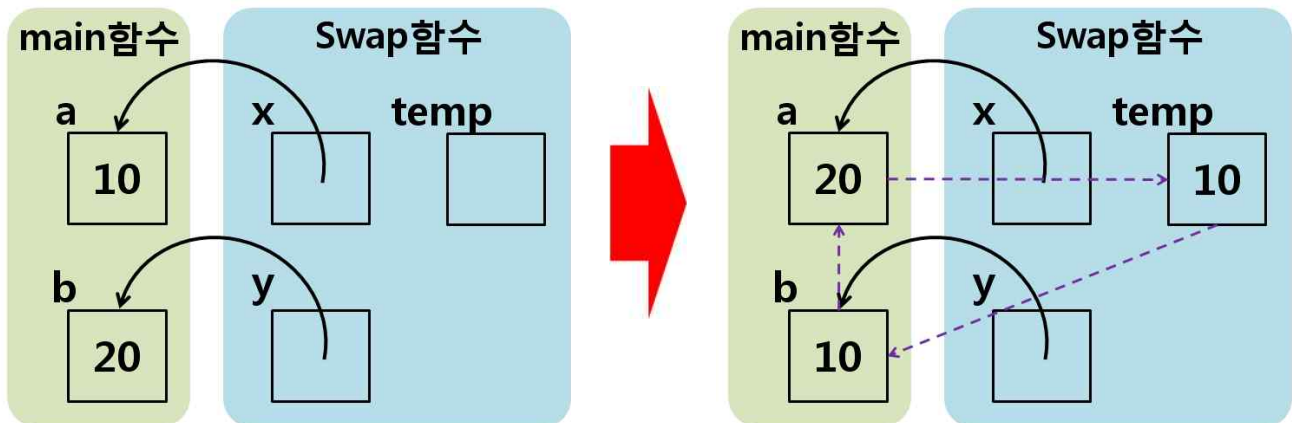
```



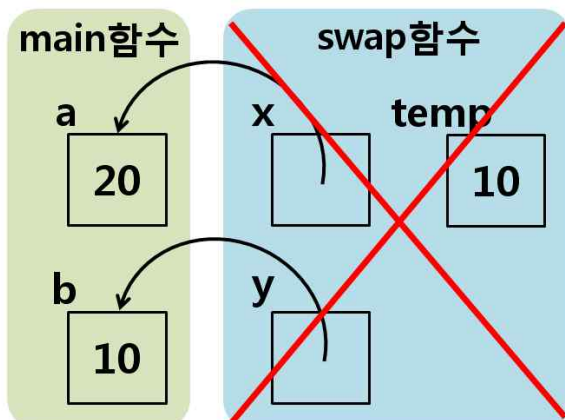
Call by pointer (Call by adress) : 매개변수로 변수의 주소값을 전달하는 방식. 호출된 함수에서 원 함수의 변수 값을 역참조를 통해 변경 할 수 있다.

예제 5-23: 매개변수를 주소값으로 전달해 main에 있는 a와b가 변경된 예제

```
#include <stdio.h>
void swap( int *x, int *y ) {
    int temp = *x;
    *x = *y;
    *y = temp;
    printf("swap함수 내에서의 x = %d, y = %d Wn", *x, *y);
}
void main ( ) {
    int a = 10, b = 20;
    printf("함수 호출 전 a = %d, b = %d Wn", a, b);
    swap( &a , &b );
    printf("함수 호출 후 a = %d, b = %d Wn", a, b);
}
```



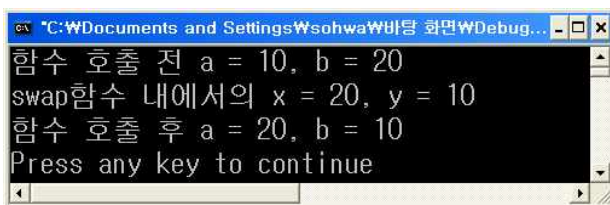
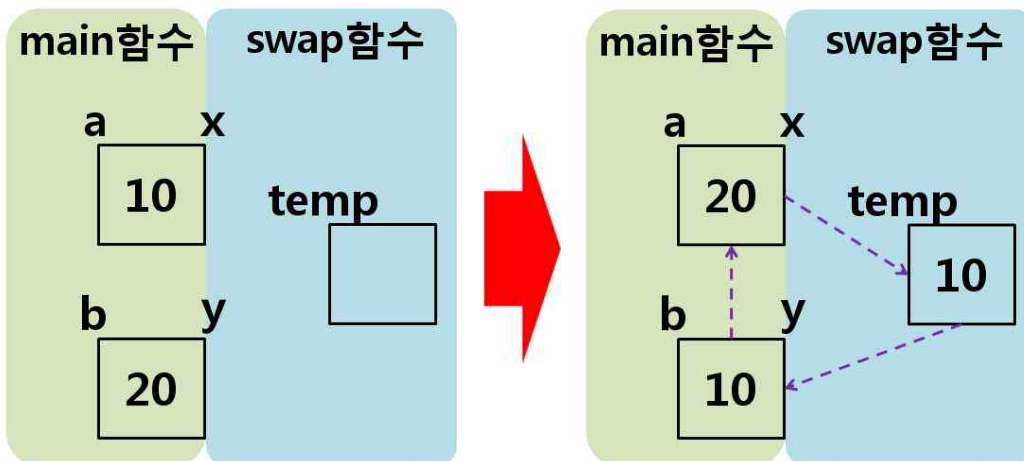
swap 함수가 종료된 뒤에는 swap함수내의 모든 변수가 사라진다.



Call by reference : 매개변수에 변수에 대한 별명을 만들어 값을 받아오는 방식. 호출된 함수에서 원 함수의 변수 값을 별명을 통해 변경 할 수 있다.
(C++에서만 사용이 가능하다.)

예제 5-24: 별명(참조자)에 의한 함수 호출로 main에 있는 a와b가 변경된 예제

```
#include <stdio.h>
void swap( int &x, int &y ) {
    int temp = x;
    x = y;
    y = temp;
    printf("swap함수 내에서의 x = %d, y = %d Wn", x, y);
}
void main ( ) {
    int a = 10, b = 20;
    printf("함수 호출 전 a = %d, b = %d Wn", a, b);
    swap( a , b );
    printf("함수 호출 후 a = %d, b = %d Wn", a, b);
}
```

변수를 만들거나 매개변수 정의 부분의 변수명 앞에 &를 붙이면 별명(참조자)을 만들라는 의미가 된다. 별명이란 이미 만들어져 있는 변수에 또 다른 이름을 부여하는 것이고 이러한 별명은 C++에서 새롭게 추가된 기능임으로 C언어에서는 사용 할 수 없다.

🔄 함수의 재귀호출(Recursive Call)

함수가 실행도중 직접 또는 간접적으로 자기 자신을 다시 호출하는 것을 재귀호출이라 한다.

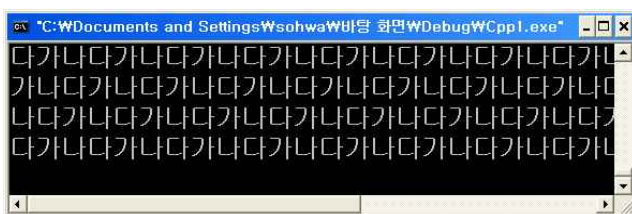
직접호출	간접 호출
<pre>void func() { func() ; }</pre>	<pre>void func() { func2() ; } void func2() { func() ; }</pre>

재귀호출 된 함수는 별도로 활성화 레코드가 만들어 지는 새로운 함수가 된다. 따라서 스택의 크기 만큼 재귀호출이 가능하다. 재귀호출은 분할정복이라는 문제 해결방법을 위해 사용한다.

※ 분할정복이란 크고 복잡한 문제를 쉽게 해결 할 수 있는 간단한 문제의 반복으로 해결하는 문제 해결 방법이다.

예제 5-25: 재귀 호출의 예제

```
#include <stdio.h>
void func( ) {
    printf("가나다");
    func( );
}
void main ( ) {
    func( );
}
```

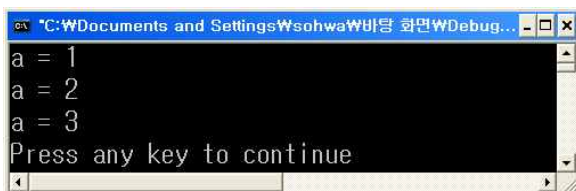


재귀호출은 무한대로 반복시켜서는 의미가 없다. 따라서 재귀호출 함수를 만들 경우에는 반드시 탈출구를 만들어야 한다.

조건을 만족했을 경우에만 재귀호출 하는 형태	조건을 만족한 경우 함수가 종료되는 형태
<pre>void func() { if (조건) { func() ; } }</pre>	<pre>void func() { if(조건) return; func2() ; }</pre>

예제 5-26: 재귀 호출을 이용해 1~3까지 출력하는 방법

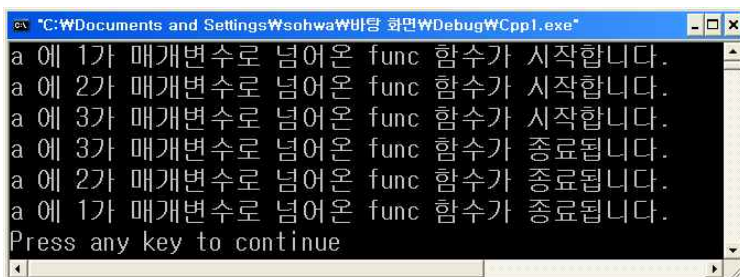
```
#include <stdio.h>
void func( int a ) {
    if ( a > 1 ) {
        func( a-1 );
    }
    printf("a = %d\n", a);
}
void main ( ) {
    func( 3 );
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
a = 1
a = 2
a = 3
Press any key to continue
```

예제 5-27: 재귀호출의 동작에 대한 이해

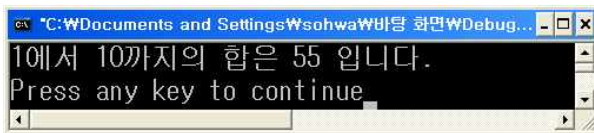
```
#include <stdio.h>
void func( int a ) {
    printf("a 에 %d가 매개변수로 넘어온 func 함수가 시작합니다.\n", a );
    if ( a < 3 ) {
        func( a + 1 );
    }
    printf("a 에 %d가 매개변수로 넘어온 func 함수가 종료됩니다.\n", a);
}
void main ( ) {
    func( 1 );
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug\WCpp1.exe
a 에 1가 매개변수로 넘어온 func 함수가 시작합니다.
a 에 2가 매개변수로 넘어온 func 함수가 시작합니다.
a 에 3가 매개변수로 넘어온 func 함수가 시작합니다.
a 에 3가 매개변수로 넘어온 func 함수가 종료됩니다.
a 에 2가 매개변수로 넘어온 func 함수가 종료됩니다.
a 에 1가 매개변수로 넘어온 func 함수가 종료됩니다.
Press any key to continue
```

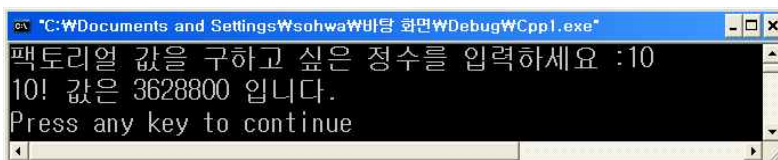
예제 5-28: 재귀호출을 이용해 1부터 10까지의 합을 구하는 예제

```
#include <stdio.h>
int func( int a ) {
    int b = 0;
    if ( a <= 10 ) {
        b = a + func( a + 1 );
    }
    return b;
}
void main ( ) {
    printf("1에서 10까지의 합은 %d 입니다.\n", func( 1 ) );
}
```



예제 5-29: 재귀호출을 이용해 1부터 입력받은 수까지의 곱을 구하는 예제

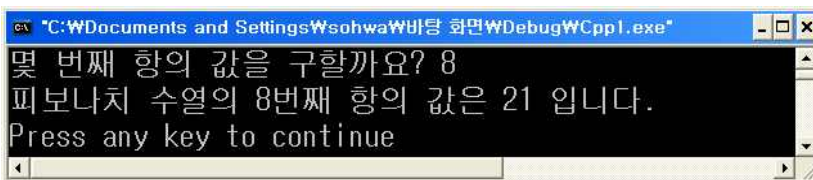
```
#include <stdio.h>
int fac( int x ) {
    if ( x <= 1 ) {
        return 1;
    }
    return x * fac(x-1);
}
void main ( ) {
    int a;
    printf("팩토리얼 값을 구하고 싶은 정수를 입력하세요 :");
    scanf("%d", &a);
    printf("%d! 값은 %d 입니다.\n", a, fac( a ) );
}
```



예제 5-30: 재귀호출을 이용해 피보나치 수열의 n번째 항을 구하는 프로그램

※ 피보나치 수열이란 1항=1, 2항=2로 시작해 n번째 항이 n-1항 + n-2항 이 되는 수열이다.

```
#include <stdio.h>
int fibo( int x ) {
    if (x == 1 || x == 2) {
        return 1;
    }
    return fibo(x - 1) + fibo(x - 2);
}
void main ( ) {
    int a;
    printf("몇 번째 항의 값을 구할까요? ");
    scanf("%d", &a);
    printf("피보나치 수열의 %d번째 항의 값은 %d 입니다.\n", a, fibo( a ) );
}
```



♻ 매크로

매크로란 프로그램에서 자주 사용하는 상수, 수식, 함수 등에 대한 별명을 정의하여 정의한 별명으로 프로그램을 작성하기 위해 사용한다. 복잡하거나 자주 사용하는 문장을 간단한 별명으로 만들어 사용 할 수 있음으로 프로그램의 작성이 편리해 지지만 너무 많이 사용하면 오히려 프로그램을 이해하기 어렵게 된다는 단점이 있다. 단, 매크로를 이용하여 프로그램의 수정을 하게 되면 보다 간단히 프로그램을 수정 할 수 있게 된다. (프로그램의 변경이 예측되는 부분이나 중요 동작을 매크로로 지정하여 향후 수정 시 소스코드를 직접 수정하지 않고 매크로만 수정하면 프로그램이 수정되도록 한다.) 이러한 형태의 매크로 사용은 많이 할수록 프로그램 수정이 용이함으로 좋은 프로그램이 된다. 매크로는 프로그램의 전처리부에 기록한다.

※ 전처리부 : 프로그램에서 가장 먼저 compile 되어지는 부분. #으로 시작하는 부분이다.

매크로정의 형식

#define 매크로명 원하는 내용

매크로 해제 형식

#undef 매크로명

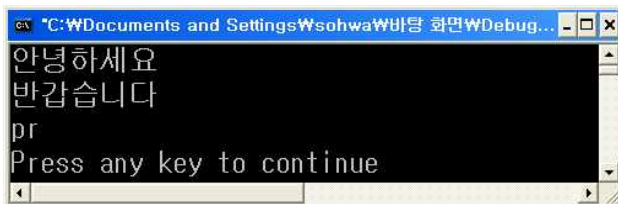
매크로명은 변수명과 동일한 규칙으로 사용자가 지정 할 수 있으며 프로그램 중간에 매크로를 해제 하고 싶다면 #undef를 이용해 매크로의 사용을 중지 할 수 있다.

매크로의 사용 예

- ① #define MAX 100
- ② #define PI 3.141592
- ③ #define plus 10+20+30
- ④ #define pa printf("a = %d\n", a);
- ⑤ #define re removeControlBox();

예제 5-31: 매크로의 사용방법

```
#include <stdio.h>
#define pr printf
void main ( ) {
    pr("안녕하세요\n");           // pr이 printf로 변경되어 컴파일 된다.
    printf("반갑습니다\n");       // 일부만 동일하다고 매크로가 동작하지는 않는다.
    pr("pr\n");                   // 문자열은 변경되지 않는다.
}
```

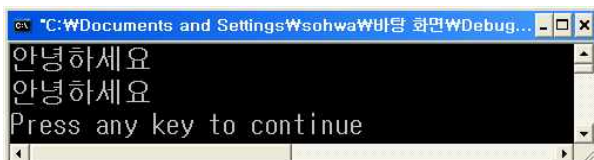


예제 5-32: 매크로의 해제 방법 (에러발생)

```
#include <stdio.h>
#define pr printf
void main ( ) {
    pr("안녕하세요\n");           // pr이 printf로 변경되어 컴파일 된다.
    #undef pr
    pr("안녕하세요\n");           // 해제한 매크로를 사용하면 에러가 발생한다.
}
```

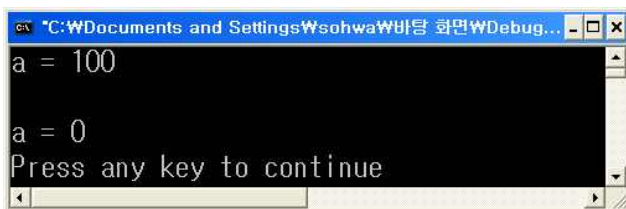
예제 5-33: 매크로의 해제 및 재지정 방법

```
#include <stdio.h>
#define pr printf
void main ( ) {
    pr("안녕하세요\n");           // pr이 printf로 변경되어 컴파일 된다.
    #undef pr                     // 매크로 pr이 해제된다.
    #define pp printf             // 새로운 매크로 pp가 지정된다.
    pp("안녕하세요\n");           // pp가 printf로 변경되어 실행된다.
}
```



예제 5-33: 매크로의 활용 - 1

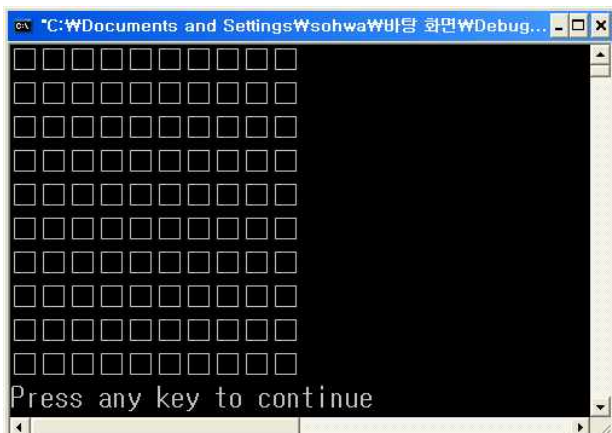
```
#include <stdio.h>
#define min 0
#define max 100+min
#define pn printf("Wn");
#define pa printf("a = %dWn", a);
void main ( ) {
    int a;
    a = max;
    pa;
    a = min;
    pn;
    pa;
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
a = 100
a = 0
Press any key to continue
```

예제 5-34: 매크로의 활용 - 2

```
#include <stdio.h>
#define garo 10
#define sero 10
void main ( ) {
    int a, b;
    for( a = 1; a <= 10; a++ ) {
        for( b = 1; b <= 10; b++ ) {
            printf("□");
        }
        printf("Wn");
    }
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
Press any key to continue
```

♻️ 매크로 함수

매크로 함수는 매크로로 하여금 매개변수를 받아오게 하여 해당 매개변수가 반영된 문장으로 치환할 수 있도록 해주는 함수를 말한다. 매크로 함수는 일반함수와는 그 의미와 동작이 다르다. 매크로 함수는 단지 문장만 치환을 해준다.

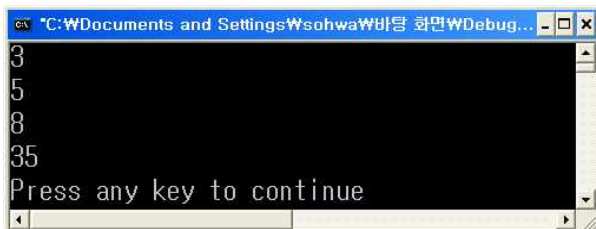
매크로정의 형식

#define 매크로함수명(매개변수...) 매개변수가사용된문장

매크로 함수의 매개변수는 자료형을 지정하지 않는다.

예제 5-35: 매크로 함수의 사용 예

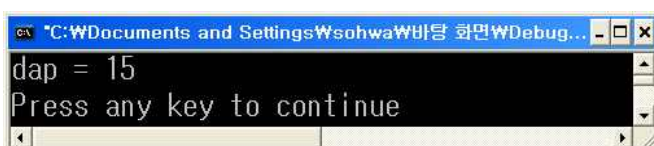
```
#include <stdio.h>
#define out(x) printf("%d\n", x);
void main ( ) {
    int a=3, b=5;
    out(a);
    out(b);
    out(a+b);
    out(35);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
3
5
8
35
Press any key to continue
```

예제 5-36: 매크로 함수는 단지 문장 치환 만을 한다는 것을 보여주는 예

```
#include <stdio.h>
#define sum(x) x + x
void main ( ) {
    int dap;
    dap = sum(3) * sum(3);
    printf("dap = %d\n", dap);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
dap = 15
Press any key to continue
```