

제8장 함수


김 강 일 (KANGIL KIM)

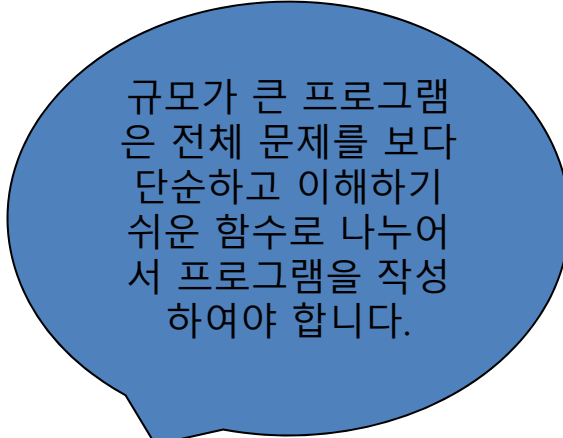
kikim01@konkuk.ac.kr

Ver. 2.0

※ 본 강의자료는 생능출판사의 "PPT 강의자료"를 기반으로 제작되었습니다.

이번 장에서 학습할 내용

- 
- 모듈화
 - 함수의 개념, 역할
 - 함수 작성 방법
 - 반환값
 - 인수 전달

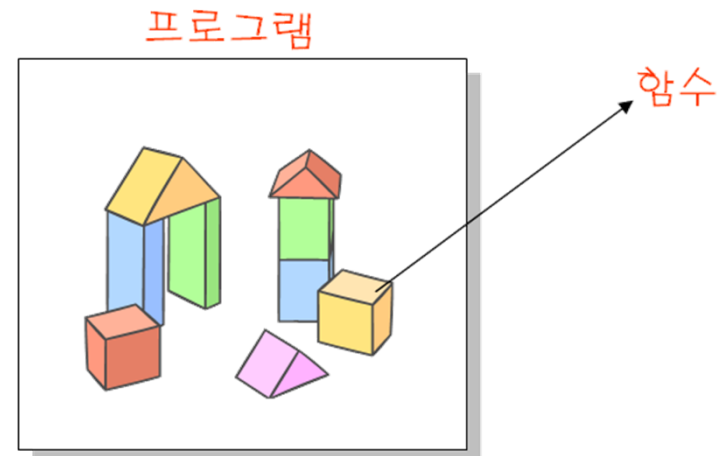


규모가 큰 프로그램은 전체 문제를 보다 단순하고 이해하기 쉬운 함수로 나누어서 프로그램을 작성하여야 합니다.



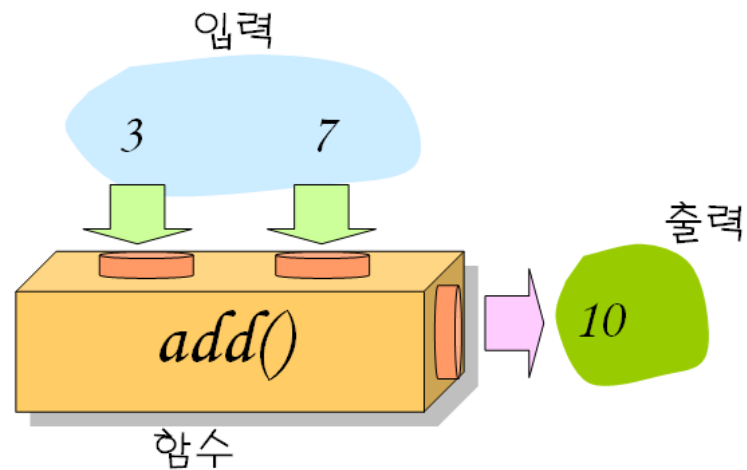
모듈의 개념

- 모듈(module)
 - 독립되어 있는 프로그램의 일부분
- Modular Programming
 - 모듈 개념을 사용하는 프로그래밍 기법
- Modular Programming의 장점
 - 각 모듈들은 독자적으로 개발 가능
 - 다른 모듈과 독립적으로 변경 가능
 - 유지 보수가 쉬워진다.
 - 모듈의 재사용 가능
- C에서는 모듈==함수



함수의 개념

- 함수(function): 특정한 작업을 수행하는 독립적인 부분
- 함수 호출(function call): 함수를 호출하여 사용하는 것
- 함수는 입력을 받으며 출력을 생성한다.

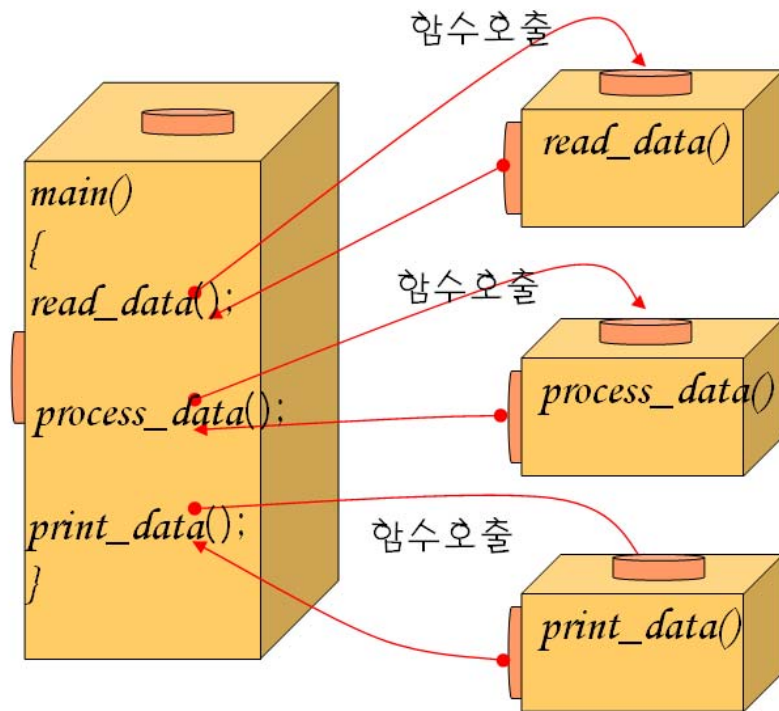


함수는 이름을 가지며 입력을 받아서 특정한 작업을 실행하고 결과를 반환합니다.



함수들의 연결

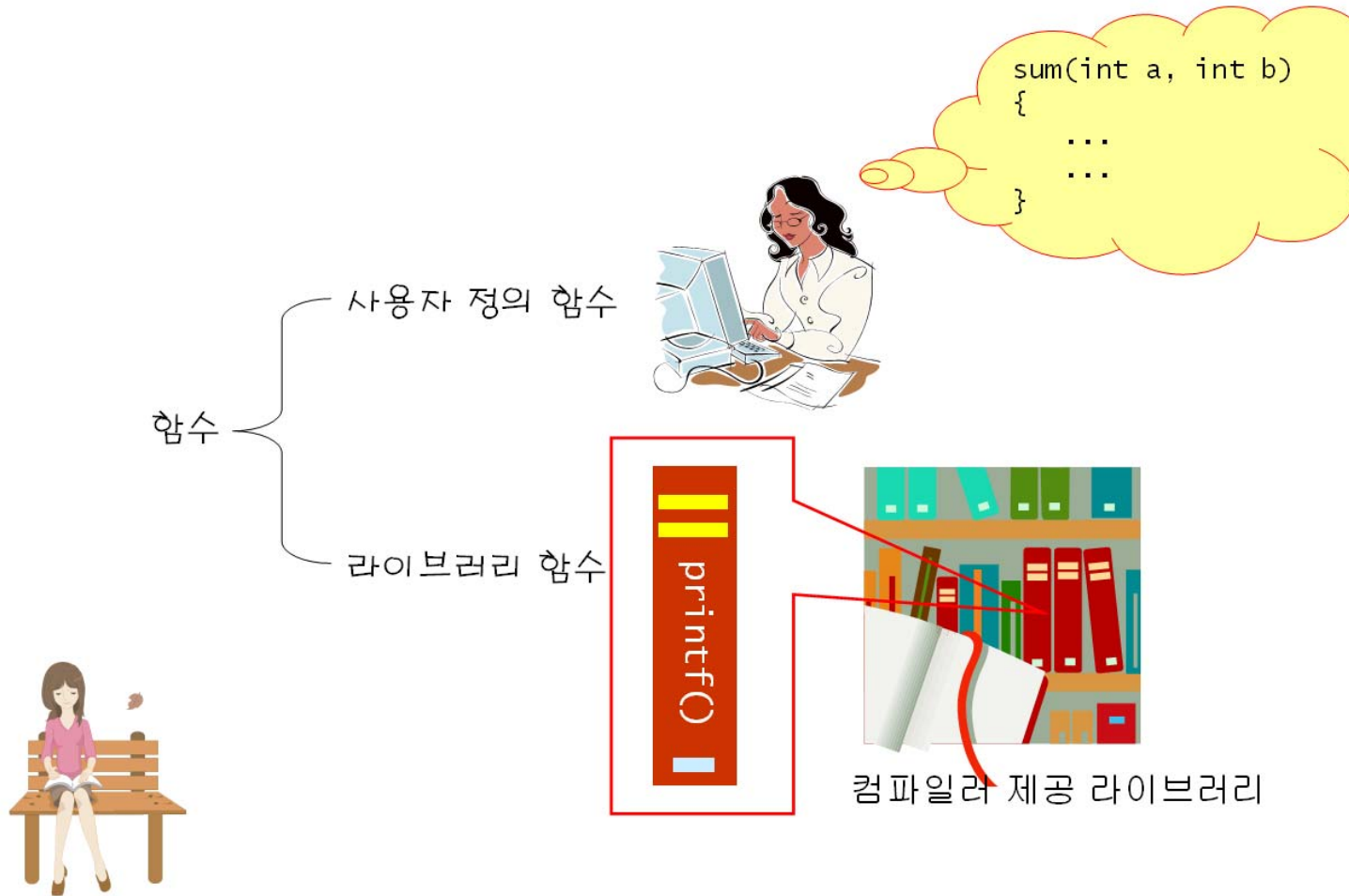
- 프로그램은 여러 개의 함수들로 이루어진다.
- 함수 호출을 통하여 서로 서로 연결된다.
- 제일 먼저 호출되는 함수는 main()이다.



각 함수들은 함수 호출을 통하여 서로 결합되어 프로그램을 구성합니다.

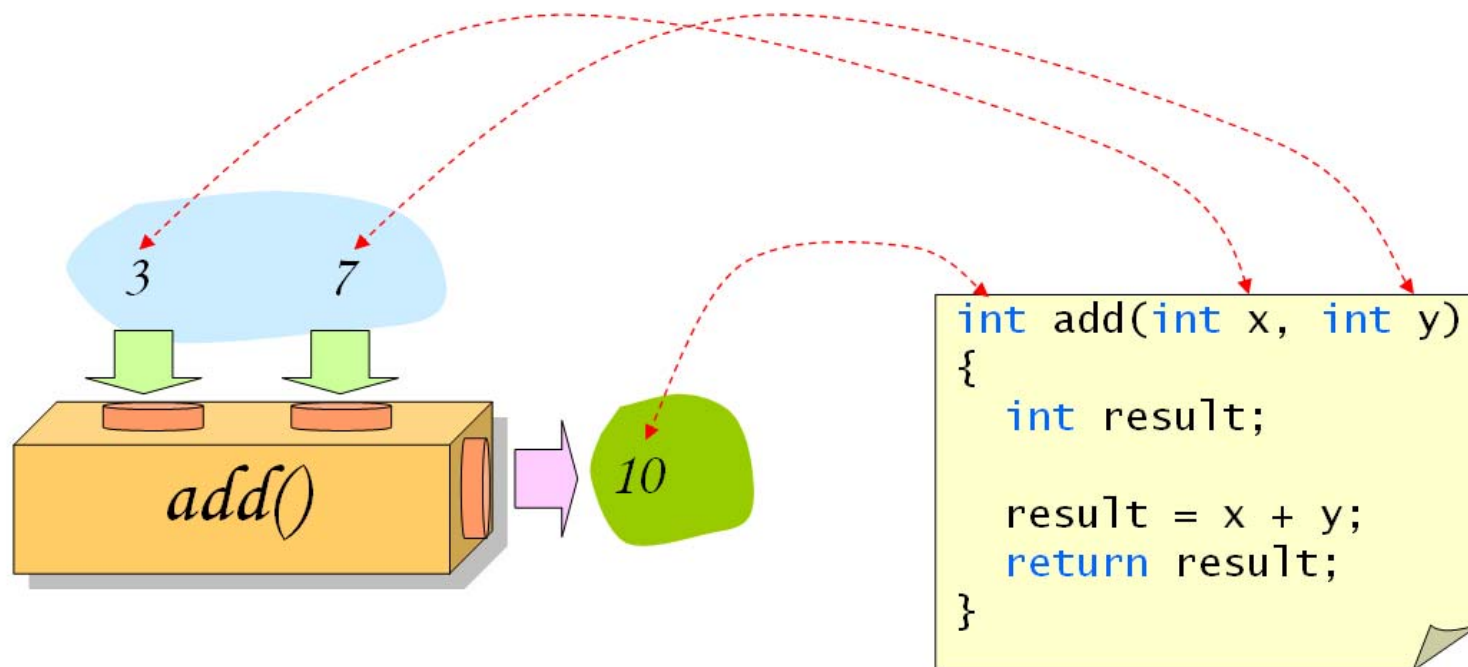


함수의 종류

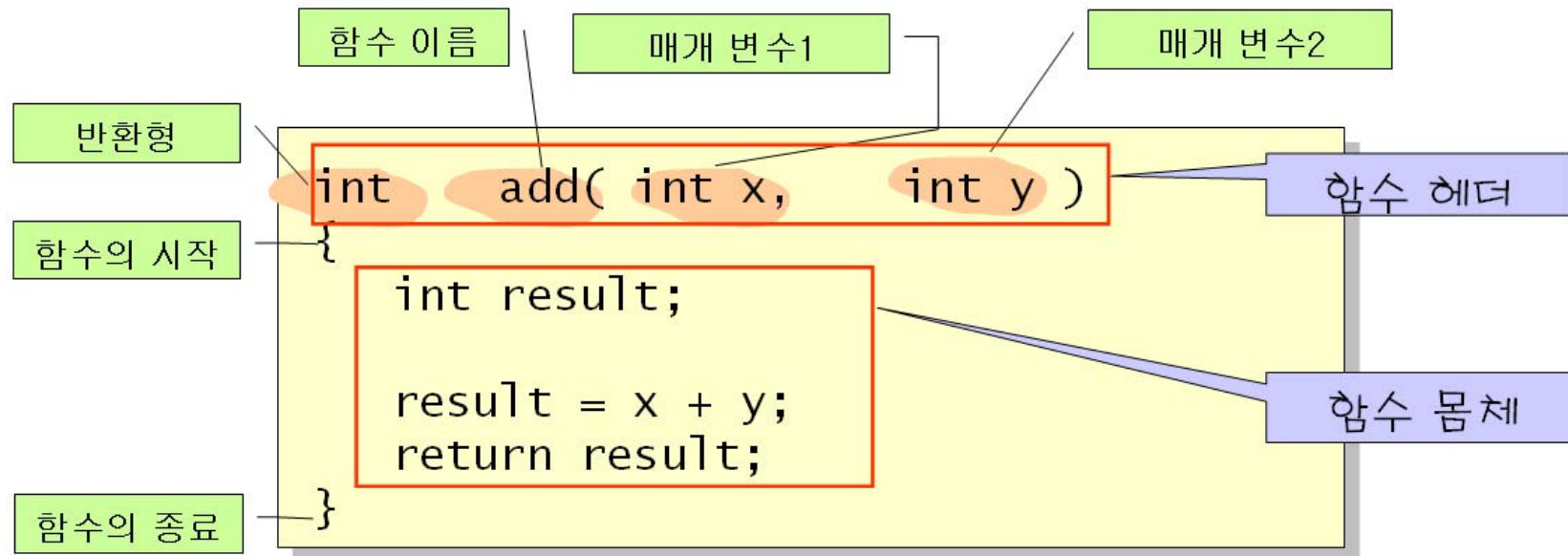


함수의 정의

- 반환형(return type)
- 함수 헤더(function header)
- 함수 몸체(function body)

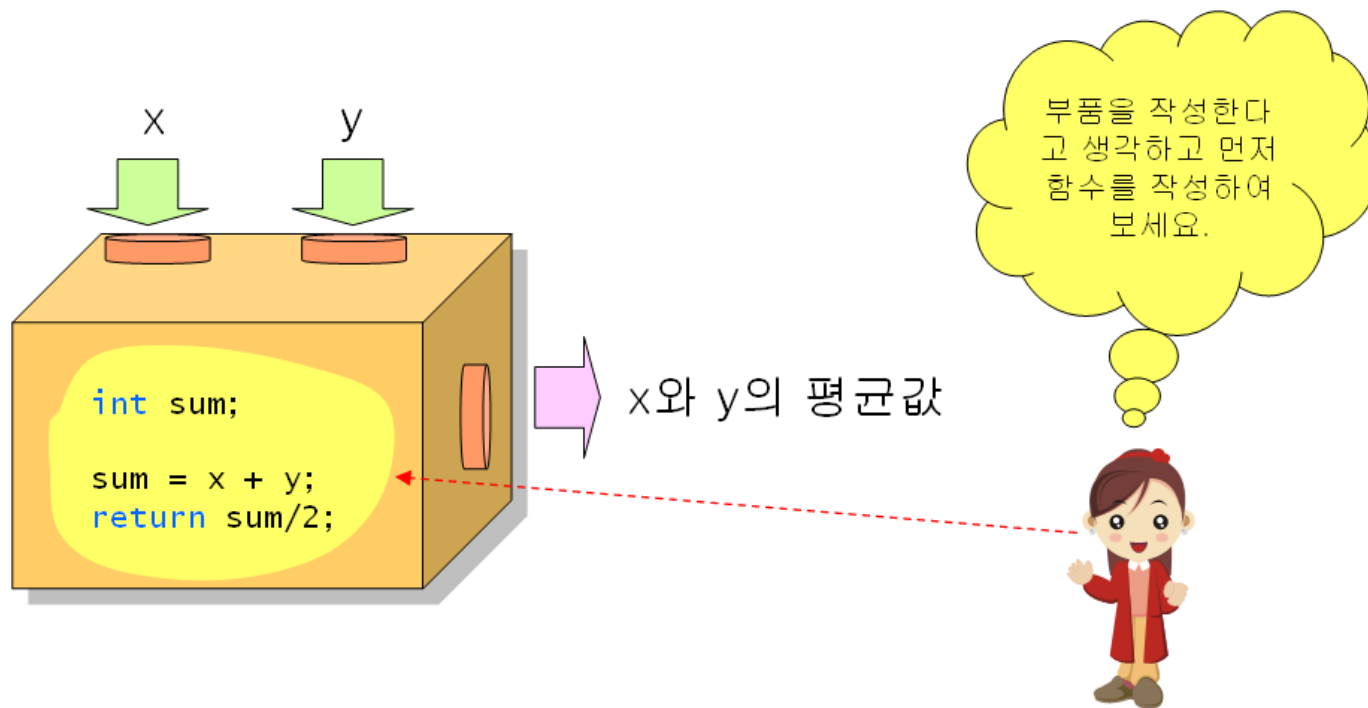


함수의 구조



함수 정의 예제

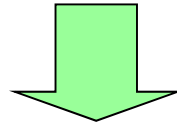
- 함수를 프로그램을 이루는 부품이라고 가정하자.
- 입력을 받아서 작업한 후에 결과를 생성한다.



예제 #1

- 정수의 제곱값을 계산하는 함수

반환값: `int`
함수 이름: `square`
매개 변수: `int n`

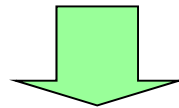


```
int square(int n)
{
    return(n*n);
}
```

예제 #2

- 두개의 정수중에서 큰 수를 계산하는 함수

반환값: `int`
함수 이름: `get_max`
매개 변수: `int x, int y`

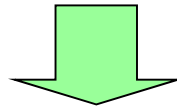


```
int get_max(int x, int y)
{
    if( x > y ) return(x);
    else return(y);
}
```

예제 #3

- 정수의 절대값을 계산하는 함수

반환값: `int`
함수 이름: `absolute`
매개 변수: `int x`

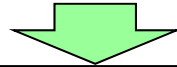


```
int absolute(int x)
{
    if( x > 0 )
        return x;
    else
        return -x;
}
```

예제 #4

- 별표 기호를 이용하여 정사각형을 그리는 함수

반환값: `void`
함수 이름: `draw_rect`
매개 변수: `int side`



```
void draw_rect(int side)
{
    int x, y;

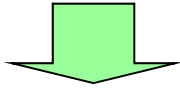
    for(y = 0; y < side; y++)
    {
        for(x = 0; x < side; x++)
            printf("*");

        printf("\n");
    }
    return;
}
```

예제 #5

- 사용자로부터 한 개의 정수를 받아서 반환하는 함수

반환값: `int`
함수 이름: `get_integer`
매개 변수: `void`



```
int get_integer(void)
{
    int n;

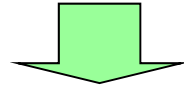
    printf("정수를 입력하시오: ");
    scanf("%d", &n);

    return n;
}
```

예제 #6

- 정수의 거듭 제곱값(x^y)을 계산하는 함수

반환값: `int`
함수 이름: `power`
매개 변수: `int x, int y`



```
int power(int x, int y)
{
    int i;
    long result = 1;

    for(i = 0; i < y; i++)
        result *= x;

    return result;
}
```

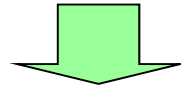
예제 #7

- 팩토리얼값($n!$)을 계산하는 함수

반환값: `int`

함수 이름: `factorial`

매개 변수: `int n`



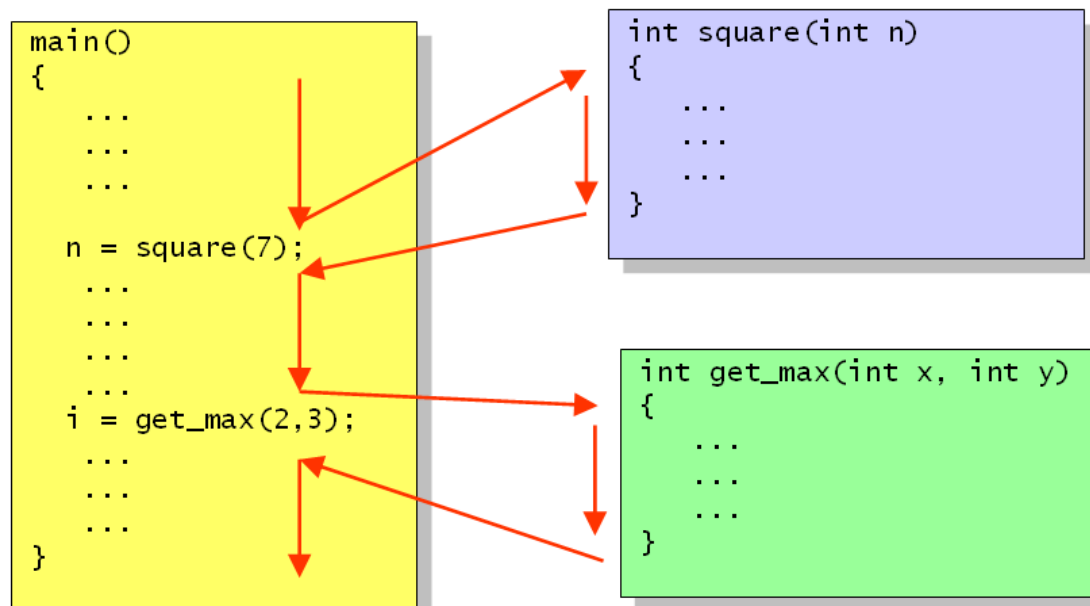
```
int factorial(int n)
{
    int i;
    long result = 1;

    for(i = 1; i <= n; i++)
        result = result * i;

    return result;
}
```


함수 호출과 반환

- 함수 호출(function call):
 - 함수를 사용하기 위하여 함수의 이름을 적어주는 것
 - 함수 안의 문장들이 순차적으로 실행된다.
 - 문장의 실행이 끝나면 호출한 위치로 되돌아 간다.
 - 결과값을 전달할 수 있다.



인수와 매개 변수

- *인수(argument)*: 실인수, 실매개 변수라고도 한다.
- *매개 변수(parameter)*: 형식 인수, 형식 매개 변수라고도 한다.

```
int main(void)
{
    ...
    i = get_max(2, 3);
    ...
}
```

인수

```
int get_max(int x, int y)
{
    ...
    ...
    ...
}
```

매개변수

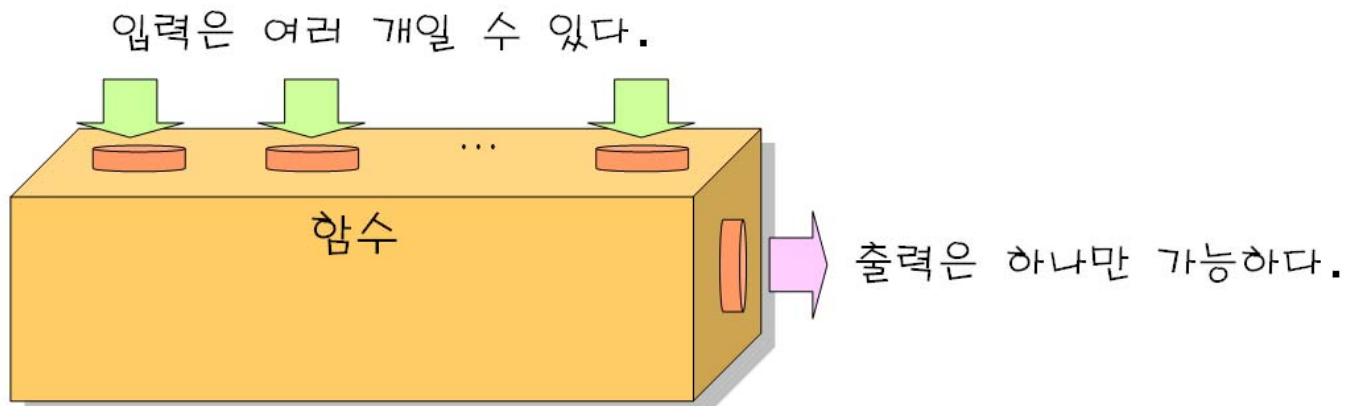
```
#include <stdio.h>
int add(int x, int y)
{
    return (x + y);
}

int main(void)
{
    // 2와 3이 add()의 인수가 된다.
    add(2, 3);

    // 5와 6이 add()의 인수가 된다.
    add(5, 6);
    return 0;
}
```

반환값

- *반환값(return value)*: 호출된 함수가 호출한 곳으로 작업의 결과값을 전달하는 것
- 인수는 여러 개가 가능하나 반환값은 하나만 가능



```
return 0;  
return(0);  
return x;  
return x+y;  
return x*x+2*x+1;
```

함수 원형

- 함수 원형(*function prototyping*): 컴파일러에게 함수에 대하여 미리 알리는 것

```
// 정수의 제곱을 계산하는 함수 예제
#include <stdio.h>
int square(int n);           // 함수 원형

int main(void)
{
    int i, result;

    for(i = 0; i < 5; i++)
    {
        result = square(i);   // 함수 호출
        printf("%d \n", result);
    }
    return 0;
}

int square(int n)            // 함수 정의
{
    return(n * n);
}
```

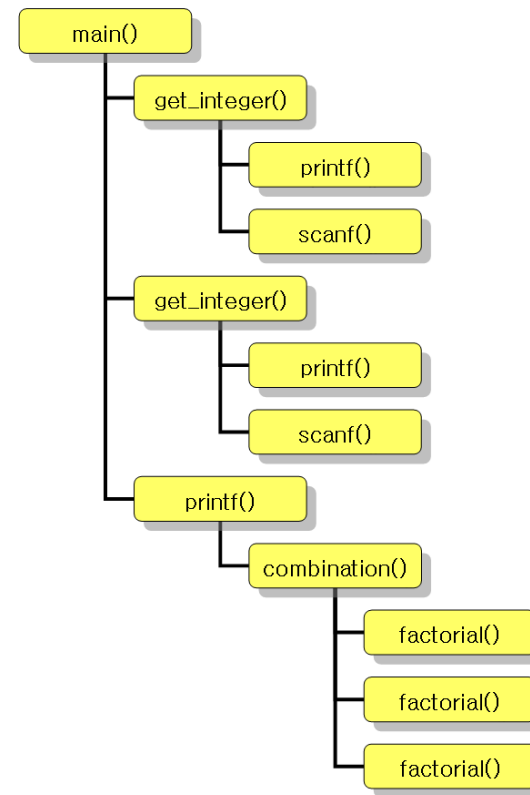
함수 원형

조합(combination) 계산 함수

- 팩토리얼 계산 함수와 get_integer() 함수를 호출하여 조합을 계산한다

$$C(n, r) = \frac{n!}{(n-r)!r!}$$

$$C(3, 2) = \frac{3!}{(3-2)!2!} = \frac{6}{2} = 3$$



예제



```
#include <stdio.h>

int get_integer(void);
int combination(int n, int r);
int factorial(int n);

int main(void)
{
    int a, b;

    a = get_integer();
    b = get_integer();

    printf("C(%d, %d) = %d \n", a, b, combination(a, b));

    return 0;
}

int combination(int n, int r)
{
    return (factorial(n)/(factorial(r) * factorial(n-r)));
}
```

예제



```
int get_integer(void)
{
    int n;

    printf("정수를 입력하시오: ");
    scanf("%d", &n);

    return n;
}

int factorial(int n)
{
    int i;
    long result = 1;

    for(i = 1; i <= n; i++)
        result = result * i;    // result = result * i

    return result;
}
```



정수를 입력하시오: 10
정수를 입력하시오: 3
 $C(10, 3) = 120$

함수 원형

- 함수 원형(*function prototype*): 미리 컴파일러에게 함수에 대한 정보를 알리는 것

반환형 함수이름 (매개변수 1, 매개변수 2, ...);

```
#include <stdio.h>

int compute_sum(int n);

int main(void)
{
    ...
    ...
    ...
    sum = compute_sum(10);
    ...
    ...
}

int compute_sum(int n)
{
    ...
}
```

함수 원형

함수 호출

함수 정의

- int compute_sum(int n);
- int get_integer(void);
- int combination(int n, int r);
- void draw_rect(int side);

OR

- int compute_sum(int);
- int get_integer(void);
- int combination(int, int);
- void draw_rect(int);

함수 원형 예제



```
#include <stdio.h>
// 함수 원형
int compute_sum(int n);

int main(void)
{
    int n, sum;

    printf("정수를 입력하시오: ");
    scanf("%d", &n);

    sum = compute_sum(n);           // 함수 사용
    printf("1부터 %d까지의 합은 %d입니다. \n", n, sum);
}

int compute_sum(int n)
{
    int i;
    int result = 0;

    for(i = 1; i <= n; i++)
        result += i;
    return result;
}
```



정수를 입력하시오: 10
1부터 10까지의 합은 55입니다.

함수 원형을 사용하지 않는 예제



```
#include <stdio.h>
// 함수 정의
int compute_sum(int n)
{
    int i;
    int result = 0;

    for(i = 1; i <= n; i++)
        result += i;

    return result;
}

int main(void)
{
    int n, sum;

    printf("정수를 입력하시오: ");
    scanf("%d", &n);

    sum = compute_sum(n);           // 함수 사용
    printf("1부터 %d까지의 합은 %d입니다. \n", n, sum);
    return 0;
}
```



정수를 입력하시오: 10
1부터 10까지의 합은 55입니다.

함수 원형과 헤더 파일

- 보통은 헤더 파일에 함수 원형이 선언되어 있음

```
/* 두개의 숫자의 합을 계산하는 프로그램 */
#include <stdio.h>

int main(void)
{
    int n1;    /* 첫번째 숫자 */
    int n2;    /* 두번째 숫자 */
    int sum;   /* 두개의 숫자의 합을 저장 */

    printf("첫번째 숫자를 입력하시오:");
    scanf("%d", &n1);

    printf("두번째 숫자를 입력하시오:");
    scanf("%d", &n2);

    sum = n1 + n2;
    printf("두수의 합: %d", sum);

    return 0;
}
```

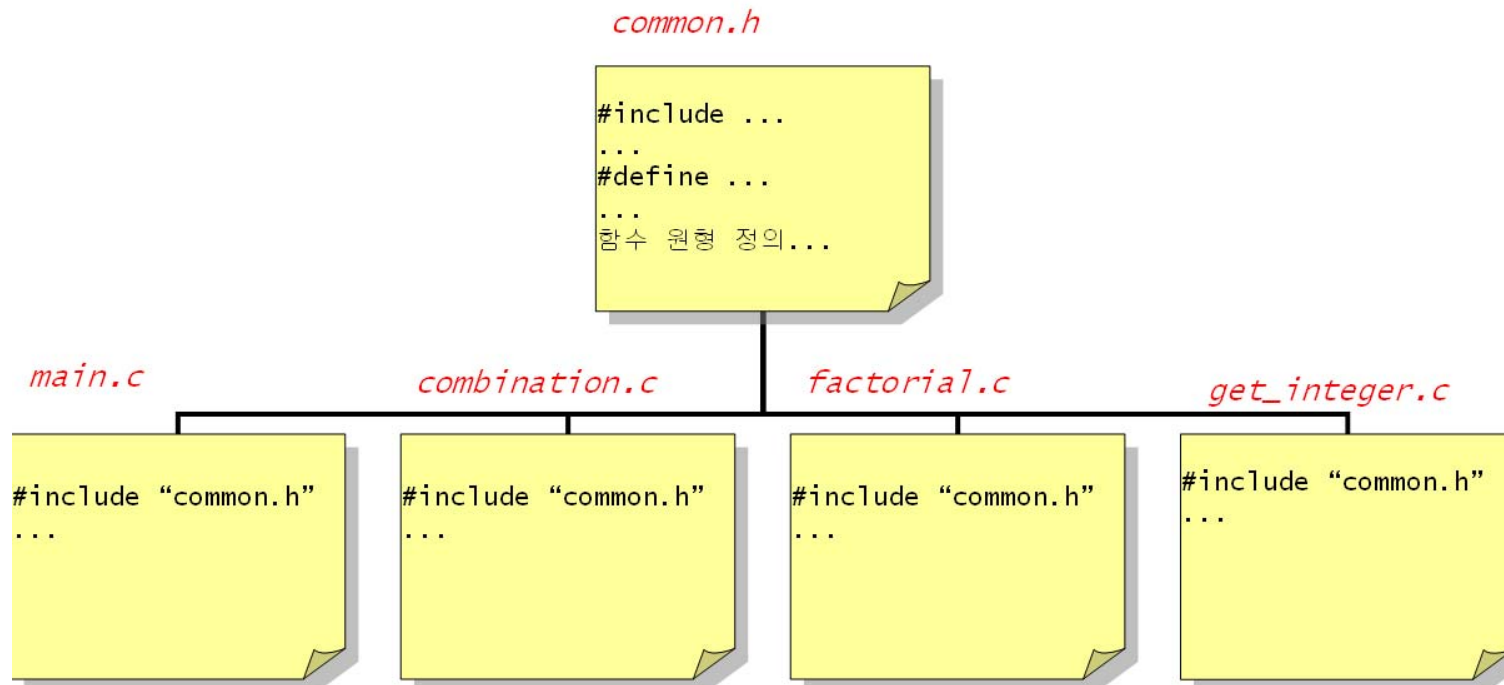
add.c

```
/**
 *stdio.h - definitions/declarations for
 *standard I/O routines
 */
...
_CRTIMP int __cdecl printf(const char
*, ...);
...
_CRTIMP int __cdecl scanf(const char
*, ...);
...
```

stdio.h

다중 소스 프로그램

- 함수 원형 정의는 헤더 파일에 들어 있고 여러 파일에서 헤더 파일을 포함



다중 소스 프로그램 예제

common.h



```
// 헤더 파일
#include <stdio.h>

#define MAX_INPUT 30

int get_integer(void);
int combination(int n, int r);
int factorial(int n);
```

main.c



```
// 수학적 조합값을 구하는 예제
#include "common.h"
int main(void)
{
    int a, b;

    a = get_integer();
    b = get_integer();

    printf("C(%d, %d) = %d \n", a, b, combination(a, b));
    return 0;
}
```

다중 소스 프로그램 예제

combination.c



```
// 수학적 조합값을 계산
#include "common.h"

int combination(int n, int r)
{
    return (factorial(n)/(factorial(r) * factorial(n-r)));
}
```

factorial.c



```
// 팩토리얼 계산
#include "common.h"

int factorial(int n)
{
    int i;
    long result = 1;

    for(i = 1; i <= n; i++)
        result = result * i;    // result = result * i

    return result;
}
```

다중 소스 프로그램 예제



get_integer.c

// 사용자로부터 정수를 입력받는 함수 정의

`#include "common.h"`

`int get_integer(void)`

`{`

`int n;`

`printf("정수를 입력하시오: ");`

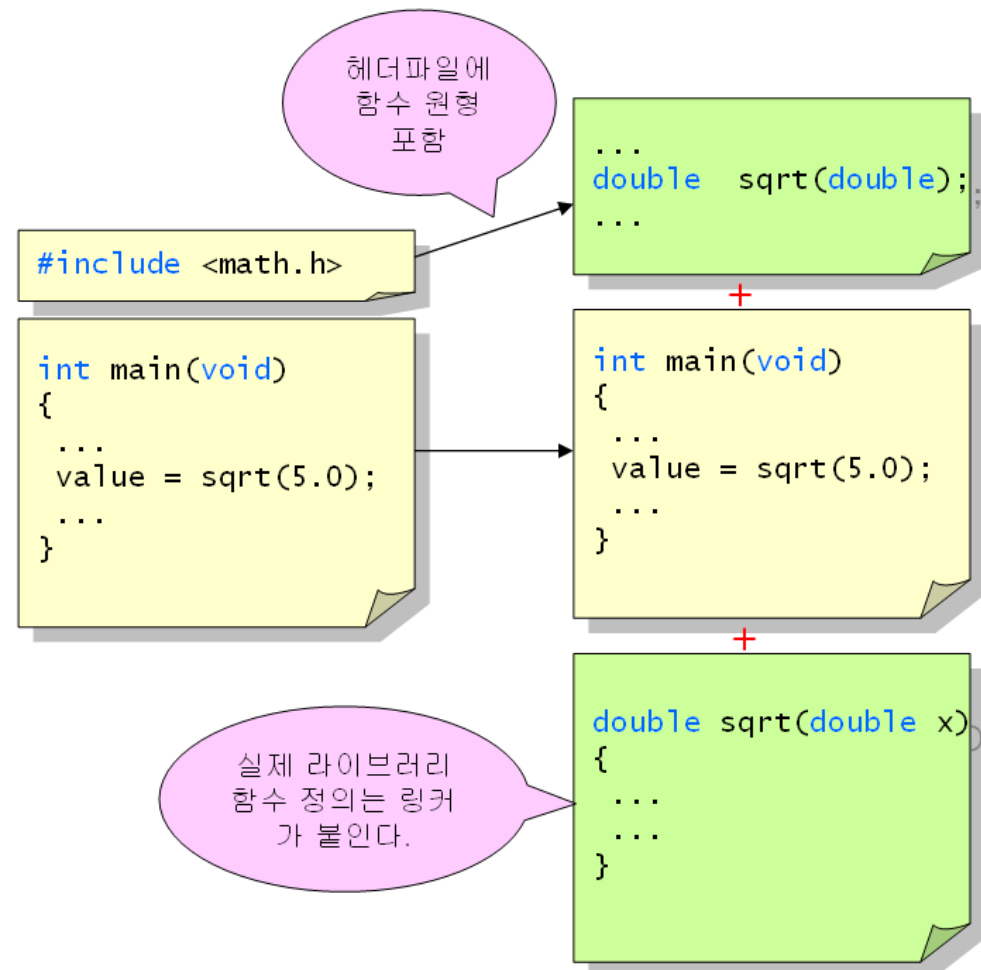
`scanf("%d", &n);`

`return n;`

`}`

라이브러리 함수

- 라이브러리 함수(library function): 컴파일러에서 제공하는 함수
 - 표준 입출력
 - 수학 연산
 - 문자열 처리
 - 시간 처리
 - 오류 처리
 - 데이터 검색과 정렬



수학 라이브러리 함수

분류	함수	설명
삼각함수	<code>double sin(double x)</code>	<u>사인값</u> 계산
	<code>double cos(double x)</code>	<u>코사인값</u> 계산
	<code>double tan(double x)</code>	<u>탄젠트값</u> 계산
역삼각함수	<code>double acos(double x)</code>	<u>역코사인값</u> 계산 <u>결과값</u> 범위 $[0, \pi]$
	<code>double asin(double x)</code>	<u>역사인값</u> 계산 <u>결과값</u> 범위 $[-\pi/2, \pi]$
	<code>double atan(double x)</code>	<u>역탄젠트값</u> 계산 <u>결과값</u> 범위 $[-\pi/2, \pi]$
쌍곡선함수	<code>double cosh(double x)</code>	쌍곡선 코사인
	<code>double sinh(double x)</code>	쌍곡선 사인
	<code>double tanh(double x)</code>	쌍곡선 탄젠트
지수함수	<code>double exp(double x)</code>	e^x
	<code>double log(double x)</code>	$\log_e x$
	<code>double log10(double x)</code>	$\log_{10} x$
기타함수	<code>double ceil(double x)</code>	x보다 작지 않은 가장 작은 정수
	<code>double floor(double x)</code>	x보다 크지 않은 가장 큰 정수
	<code>double fabs(double x)</code>	x의 <u>절대값</u>
	<code>double pow(double x, double y)</code>	x^y
	<code>double sqrt(double x)</code>	\sqrt{x}

예제



```
// 삼각 함수 라이브러리
```

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

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

```
int main( void )
```

```
{
```

```
    double pi = 3.1415926535;
```

```
    double x, y;
```

```
    x = pi / 2;
```

```
    y = sin( x );
```

```
    printf( "sin( %f ) = %f\n", x, y );
```

```
    y = sinh( x );
```

```
    printf( "sinh( %f ) = %f\n", x, y );
```

```
    y = cos( x );
```

```
    printf( "cos( %f ) = %f\n", x, y );
```

```
    y = cosh( x );
```

```
    printf( "cosh( %f ) = %f\n", x, y );
```

```
}
```



```
sin( 1.570796 ) = 1.000000
```

```
sinh( 1.570796 ) = 2.301299
```

```
cos( 1.570796 ) = 0.000000
```

```
cosh( 1.570796 ) = 2.509178
```

직각 삼각형 예제



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

#define RAD_TO_DEG (45.0/atan(1))

int main(void)
{
    double w, h, r, theta;

    printf("밑변과 높이를 입력하시오:");
    scanf("%lf %lf", &w, &h);

    r = sqrt(w * w + h * h);
    theta = RAD_TO_DEG * atan2(h, w);

    printf("빗변= %f 각도= %f\n", r, theta);
    return 0;
}
```



밑변과 높이를 입력하시오: 10.0 10.0
빗변= 14.142136 각도= 45.000000

수학 라이브러리 함수들

- `abs(int x), fabs(double x)`
 - `abs(-9)` // 9를 반환
 - `fabs(-3.67)` // 3.67을 반환
- `pow(double x, double y)`
 - 인수 `x`의 `y`- 거듭제곱인 x^y 을 계산한다.
 - `pow(2.0, 3.0);` // 8.0을 반환
- `sqrt(double x)`
 - 주어진 수의 제곱근을 구한다. 만약에 음수가 입력되면 오류가 발생한다.
 - `sqrt(9.0);` // 3.0을 반환
- `ceil(double x)`
 - `ceil`은 `x`보다 작지 않은 가장 작은 정수를 반환
 - `ceil(-2.9);` // -2.0을 반환
 - `ceil(2.9);` // 3.0을 반환
- `floor(double x)`
 - `floor()`는 `x`보다 크지 않은 가장 큰 정수를 반환한다.
 - `floor(-2.9);` // -3.0을 반환
 - `floor(2.9);` // 2.0을 반환

난수 생성 라이브러리 함수



```
// 난수 생성 프로그램
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

// n개의 난수를 화면에 출력한다.
void get_random( int n )
{
    int i;
    for( i = 0; i < n; i++ )
        printf( " %6d\n", rand() ); // 0부터 RAND_MAX까지의 난수를 생성한다.
}

int main( void )
{
    // 일반적으로 난수 발생기의 시드(seed)를 현재 시간으로 설정한다.
    // 현재 시간은 수행할 때마다 달라지기 때문이다.
    srand( (unsigned)time( NULL ) );
    get_random( 10 );
    return 0;
}
```



```
21783
14153
4693
13117
21900
19957
15212
20710
4357
16495
```

함수를 사용하는 이유

- 소스 코드의 중복을 없애준다.
 - 한번 만들어진 함수를 여러 번 호출하여 사용할 수 있다.
- 한번 작성된 함수를 다른 프로그램에서도 사용할 수 있다.
- 복잡한 문제를 단순한 부분으로 분해할 수 있다.

```
void print_heading(void)
{
    printf("*****");
    printf(" NAME ADDRESS PHONE ");
    printf("*****");
}
int main(void)
{
    // 출력이 필요한 위치 #1
    print_heading();
    ...
    // 출력이 필요한 위치 #2
    print_heading();
    ...
    ...
}
```

```
int main(void)
{
    ...
    read_list();
    sort_list();
    print_list();
    ...
}
```

Q & A

