

CHAPTER

08

함수

- 함수의 개념, 함수의 장점을 이해한다.
- 함수를 작성할 수 있고 함수를 호출할 수 있다.
- 함수와 함수 사이에서 정보를 전달하고 받는 매커니즘을 이해한다.

Contents

3

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

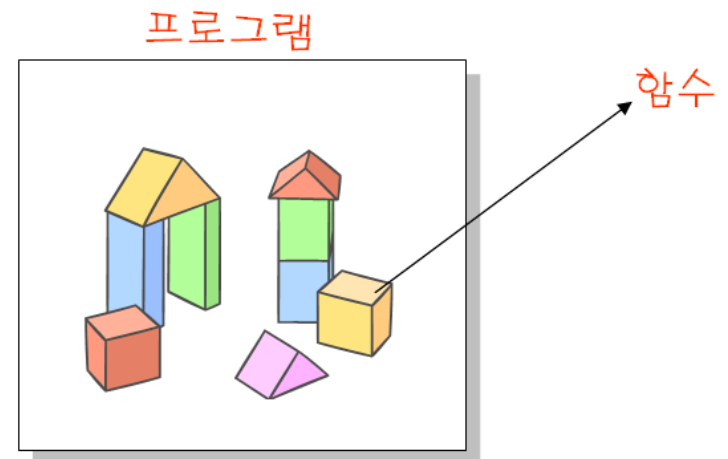
8.7

함수를 사용하는 이유

모듈의 개념

4

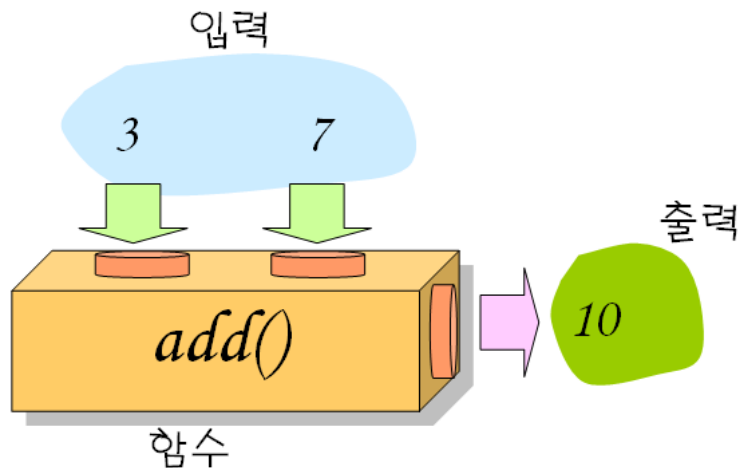
- 모듈(module)
 - ▣ 독립되어 있는 프로그램의 일부분
- 모듈러 프로그래밍
 - ▣ 모듈 개념을 사용하는 프로그래밍 기법
- 모듈러 프로그래밍의 장점
 - ▣ 각 모듈들은 독자적으로 개발 가능
 - ▣ 다른 모듈과 독립적으로 변경 가능
 - ▣ 유지 보수가 쉬워진다.
 - ▣ 모듈의 재사용 가능
- C에서는 모듈==함수



함수의 개념

5

- 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.



함수의 필요성

6

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

```
int main(void)
```

```
{
```

```
    int i;
```

```
    for(i = 0; i < 10; i++)  
        printf("*");
```

```
    ...
```

```
    for(i = 0; i < 10; i++)  
        printf("*");
```

```
    ...
```

```
    for(i = 0; i < 10; i++)  
        printf("*");
```

```
    return 0;
```

```
}
```

10개의 *을 출력하는 코드

10개의 *을 출력하는 코드

10개의 *을 출력하는 코드

함수의 필요성

7

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

```
void print_star()
```

```
{
```

```
    int i;
```

```
    for(i = 0; i < 10; i++)
```

```
        printf("*");
```

```
}
```

```
int main(void)
```

```
{
```

```
    print_star();
```

```
    ...
```

```
    print_star();
```

```
    ...
```

```
    print_star();
```

```
    return 0;
```

```
}
```

함수를 정의하였다. 함수는 한번 정의되면 여러 번 호출하여서 실행이 가능하다.

함수의 장점

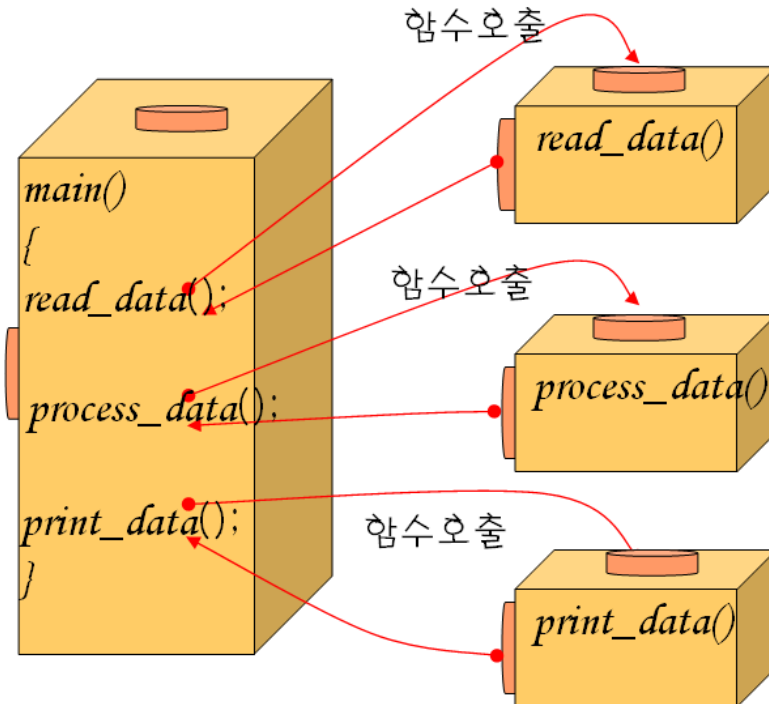
8

- 함수를 사용하면 코드가 중복되는 것을 막을 수 있다.
- 한번 작성된 함수는 여러 번 재사용할 수 있다.
- 함수를 사용하면 전체 프로그램을 모듈로 나눌 수 있어서 개발 과정이 쉬워지고 보다 체계적이 되면서 유지보수도 쉬워진다.

함수들의 연결

9

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

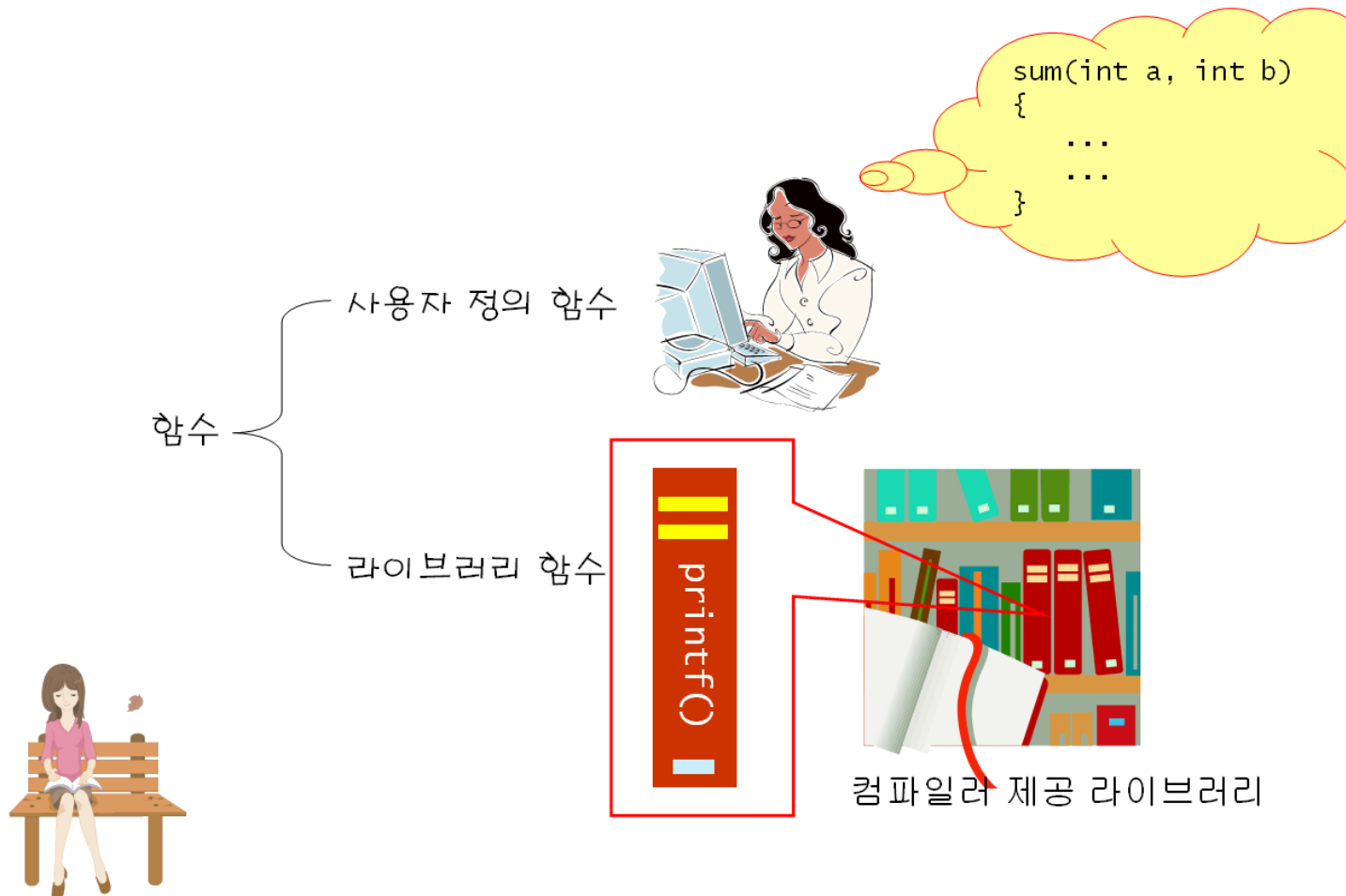


각 함수들은 함수 호출을 통하여 서로 결합되어 프로그램을 구성합니다.



함수의 종류

10



Contents

11

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

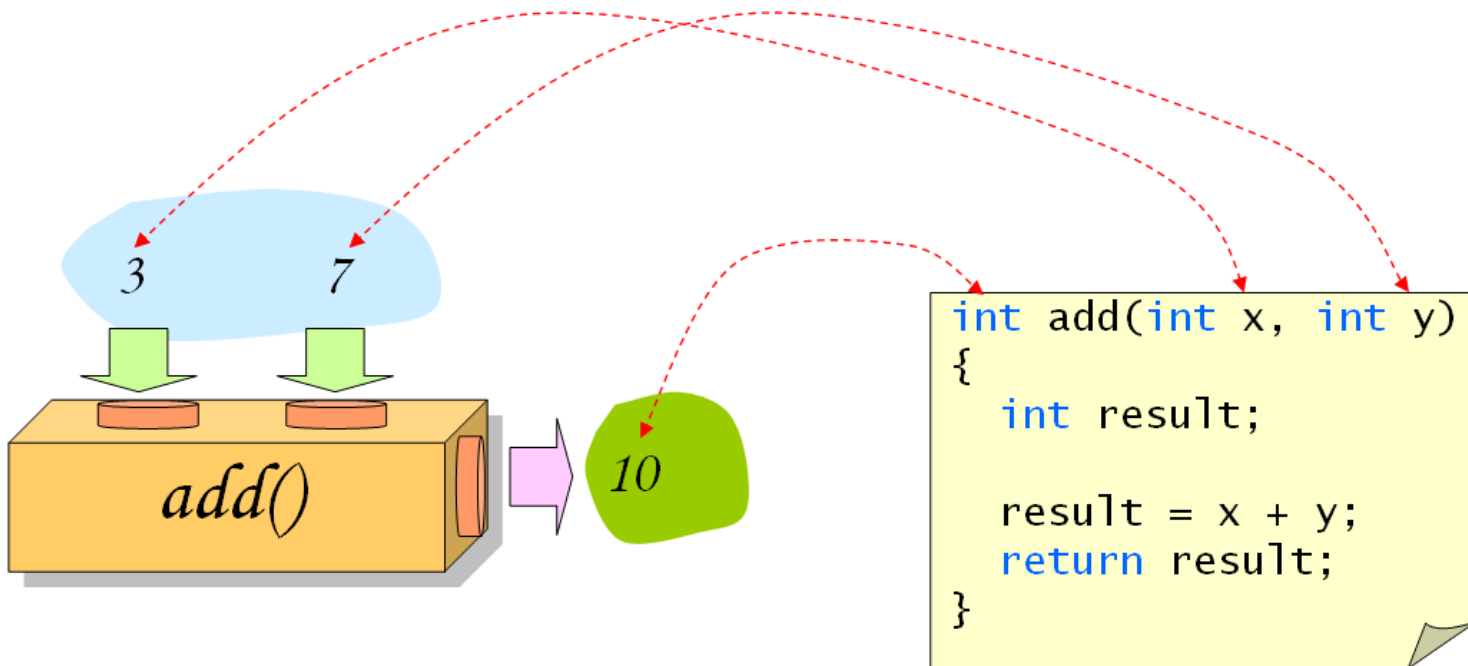
8.7

함수를 사용하는 이유

함수의 정의

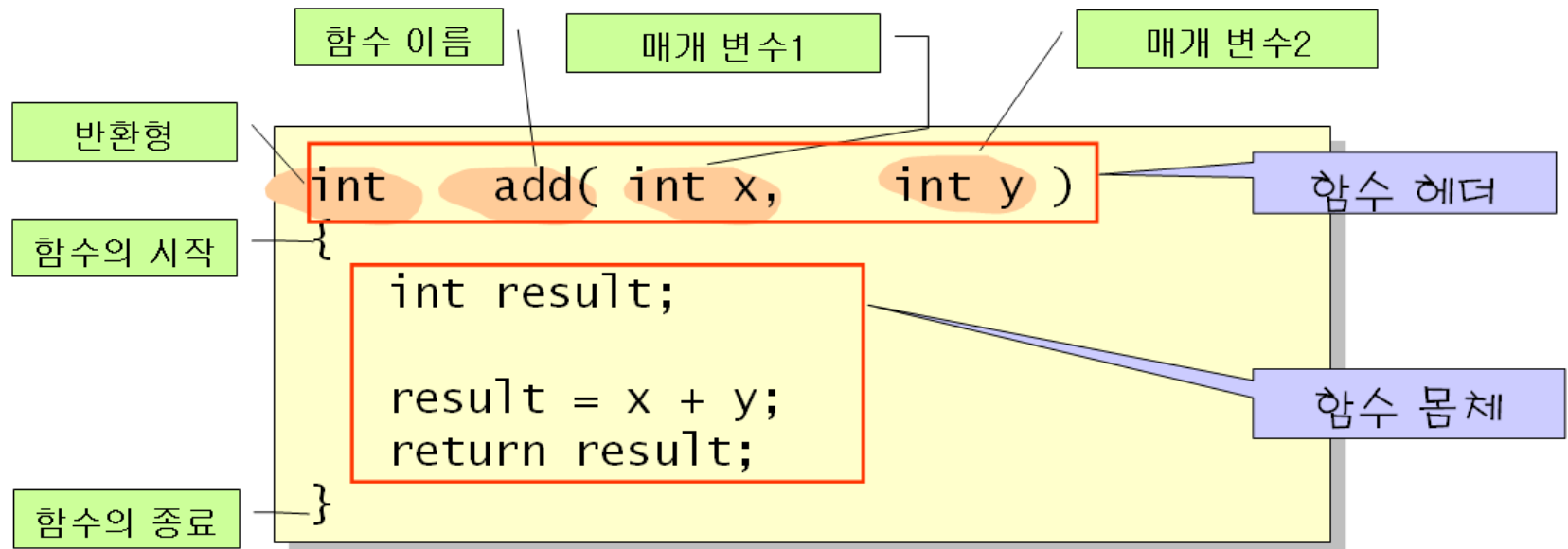
12

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



함수의 구조

13



반환형

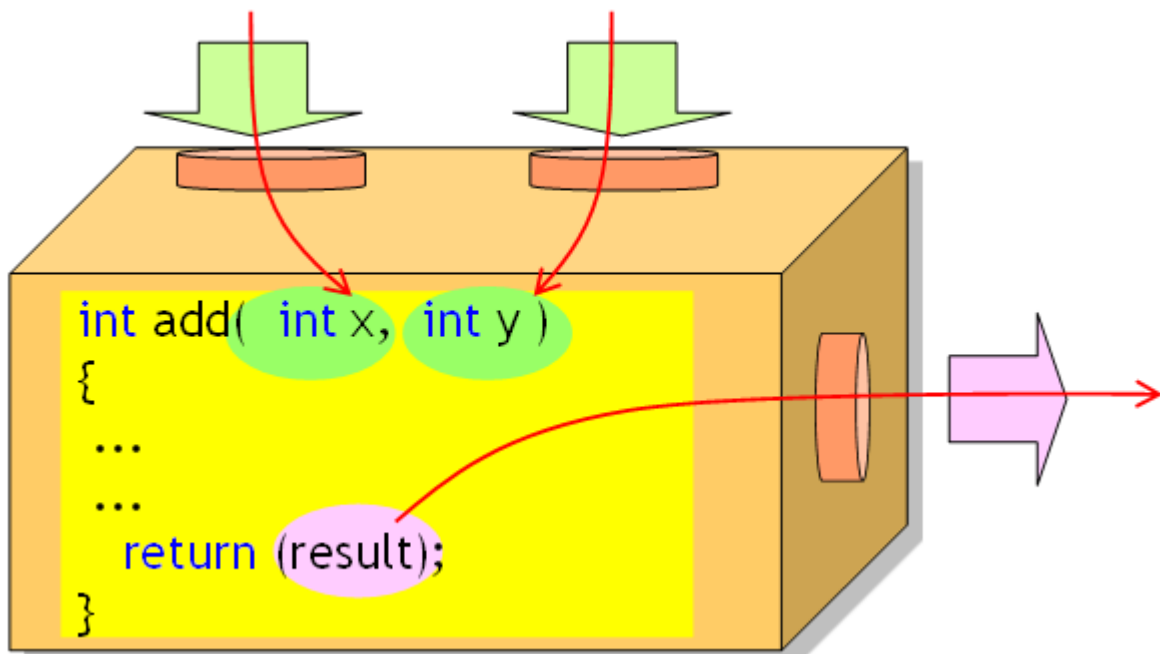
14

int
double
void

square()
compute_average()
set_cursor_type()

// int 형의 값을 반환한다.
// double 형의 값을 반환한다.
// 반환값이 없는 함수

반환형



매개 변수

15

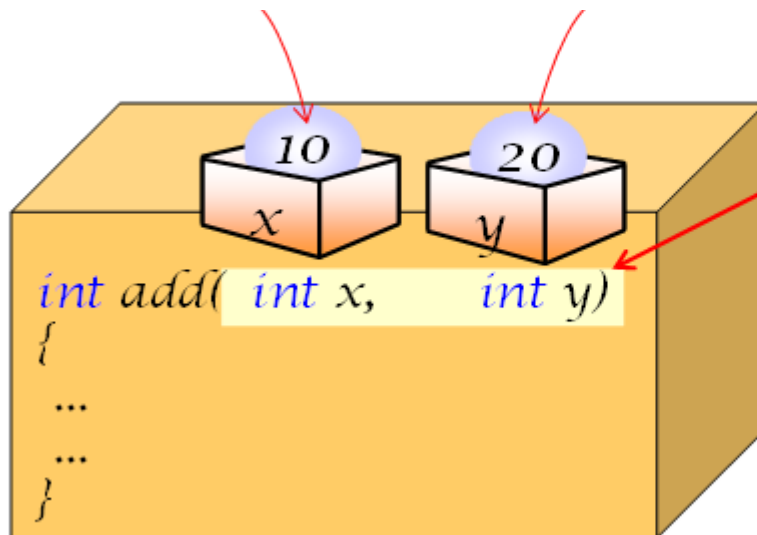
```
int square(int n)
double compute_average(double x, double y)
void get_cursor_type(void)
```

// 정수를 제공하는 함수

// 평균을 구하는 함수

// 커서의 타입을 반환하는 함수

매개 변수



매개변수는 외부에서 전달되는 데이터가 저장되는 변수

Contents

16

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

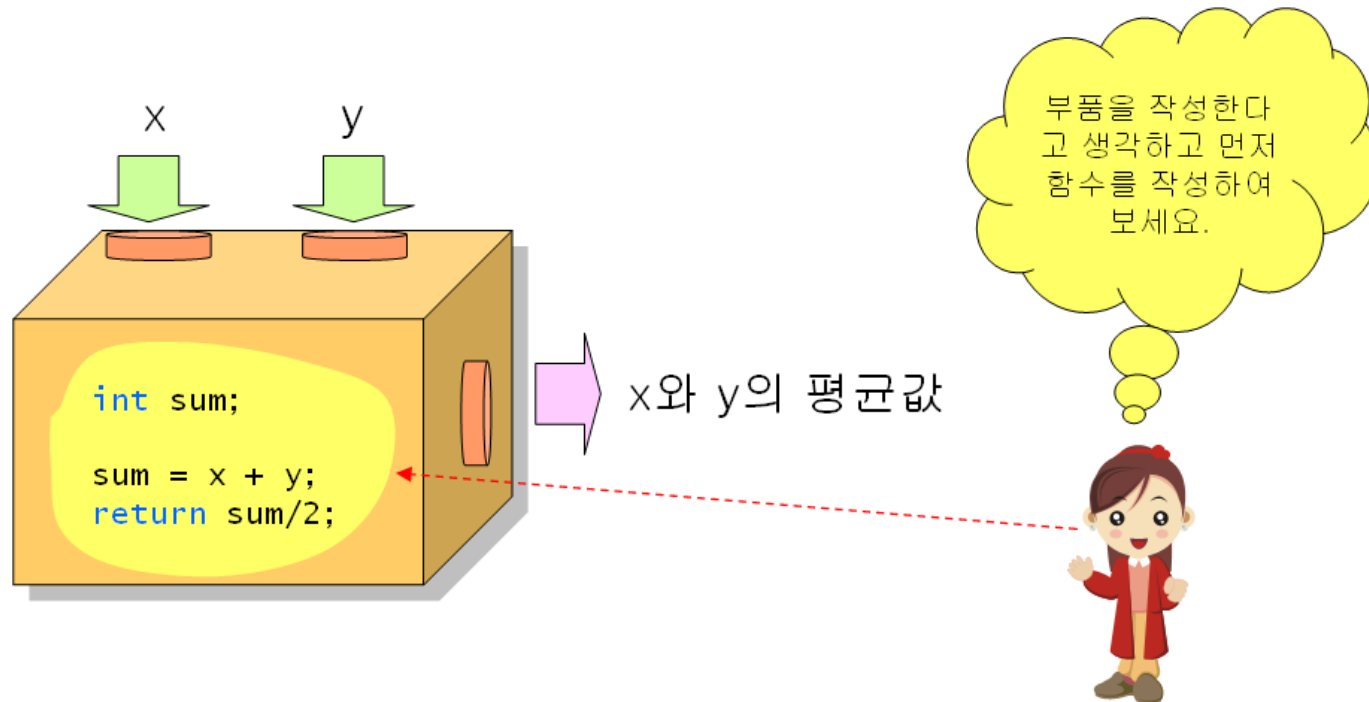
8.7

함수를 사용하는 이유

함수 정의 예제

17

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

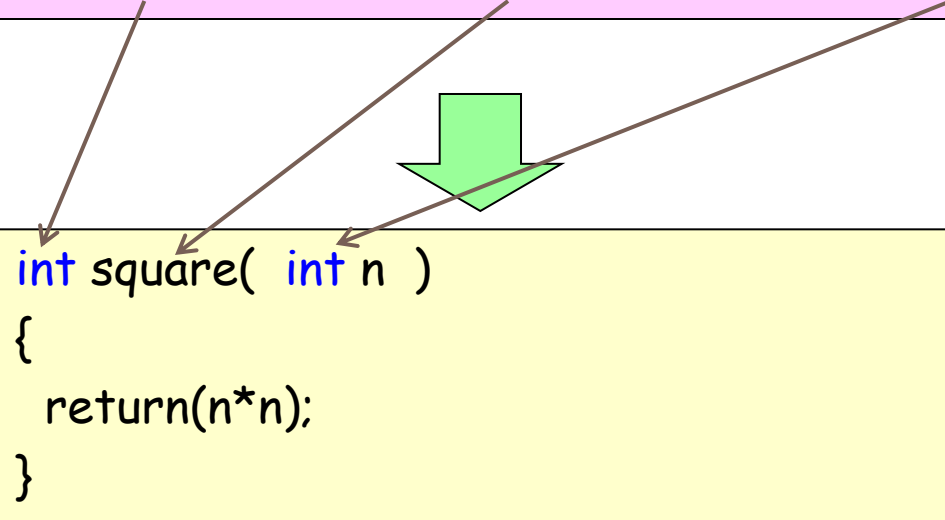


예제 #1

18

□ 정수의 제곱값을 계산하는 함수

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



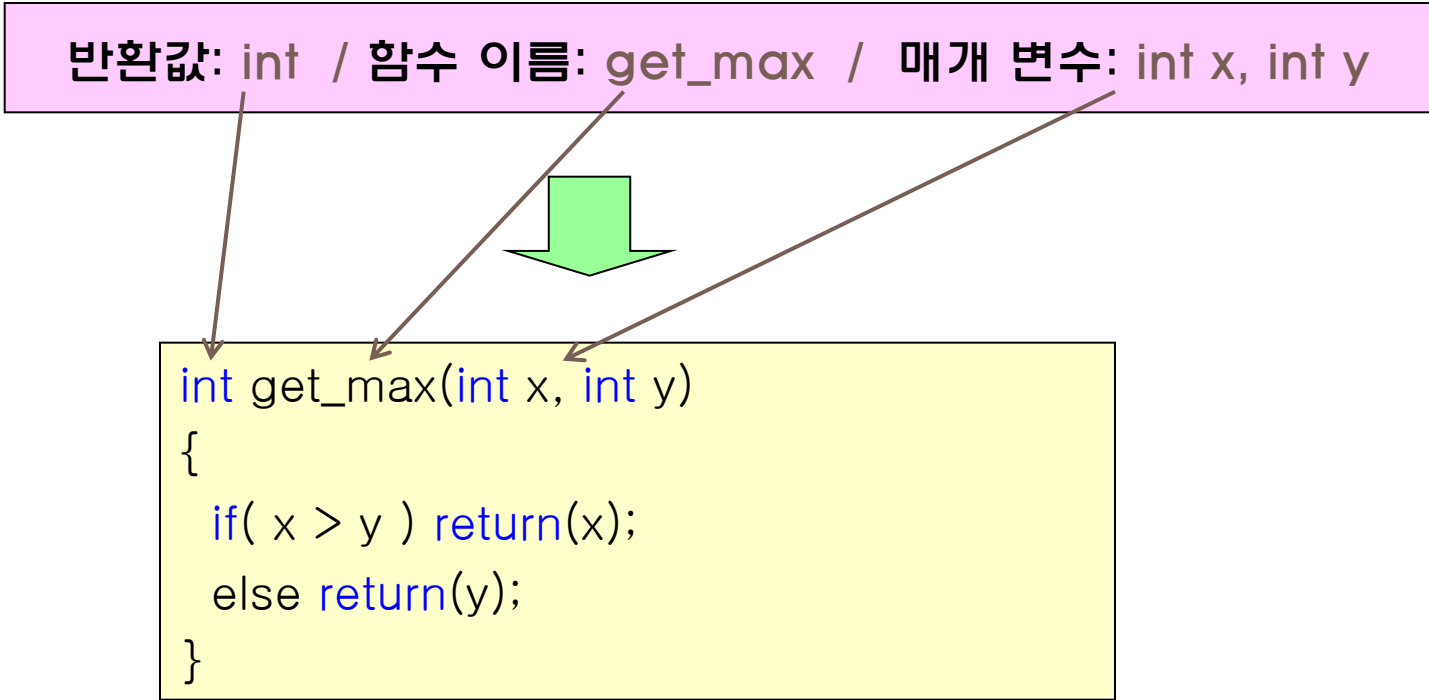
```
int square( int n )  
{  
    return(n*n);  
}
```

예제 #2

19

- 두개의 정수중에서 큰 수를 계산하는 함수

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



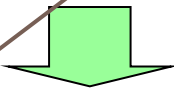
```
int get_max(int x, int y)
{
    if( x > y ) return(x);
    else return(y);
}
```

예제 #3

20

□ 별표 기호를 이용하여 정사각형을 그리는 함수

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



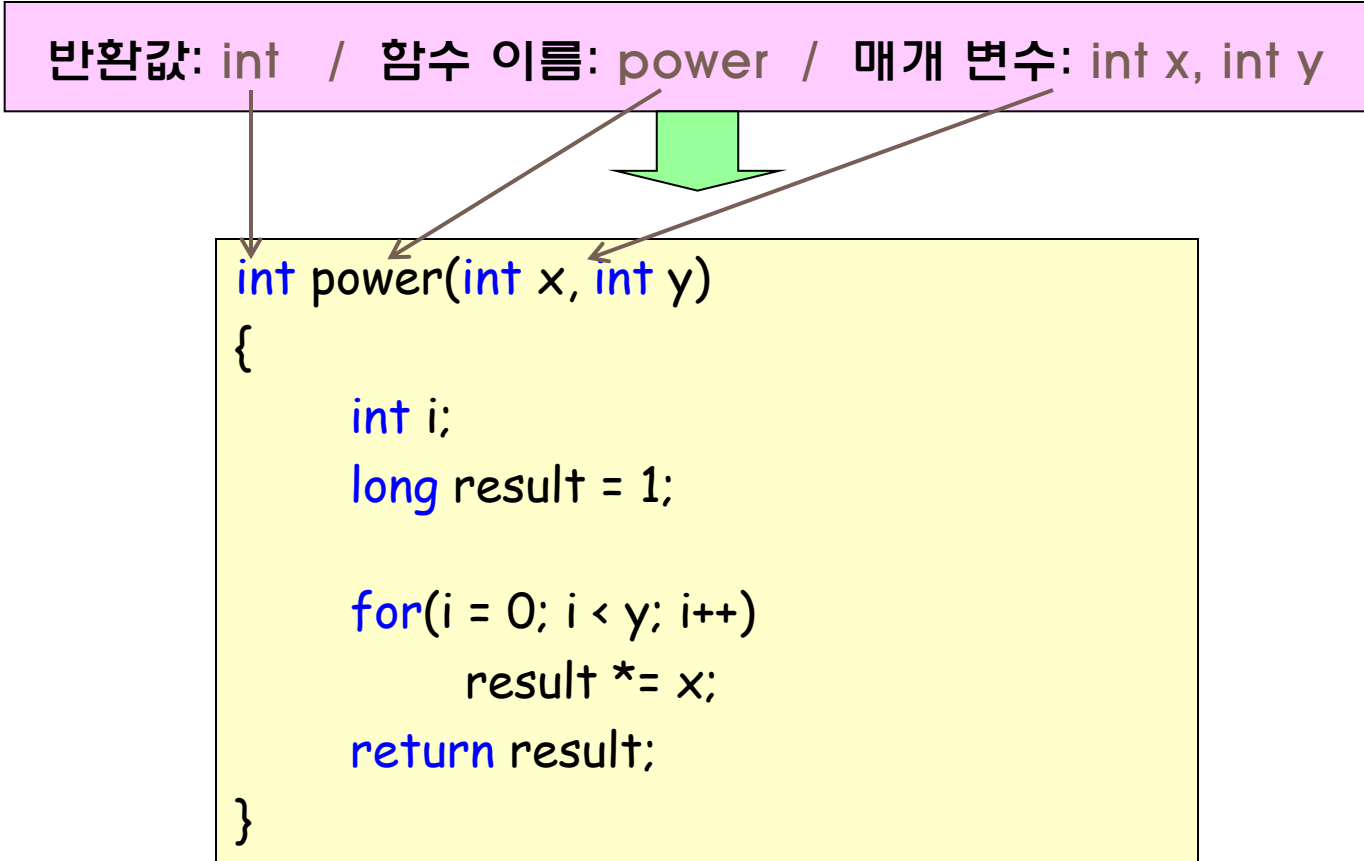
```
void draw_rect(int side)
{
    int x, y;
    for(y = 0; y < side; y++)
    {
        for(x = 0; x < side; x++)
            printf("*");
        printf("\n");
    }
    return;
}
```

예제 #4

21

□ 정수의 거듭 제곱값(xy)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;

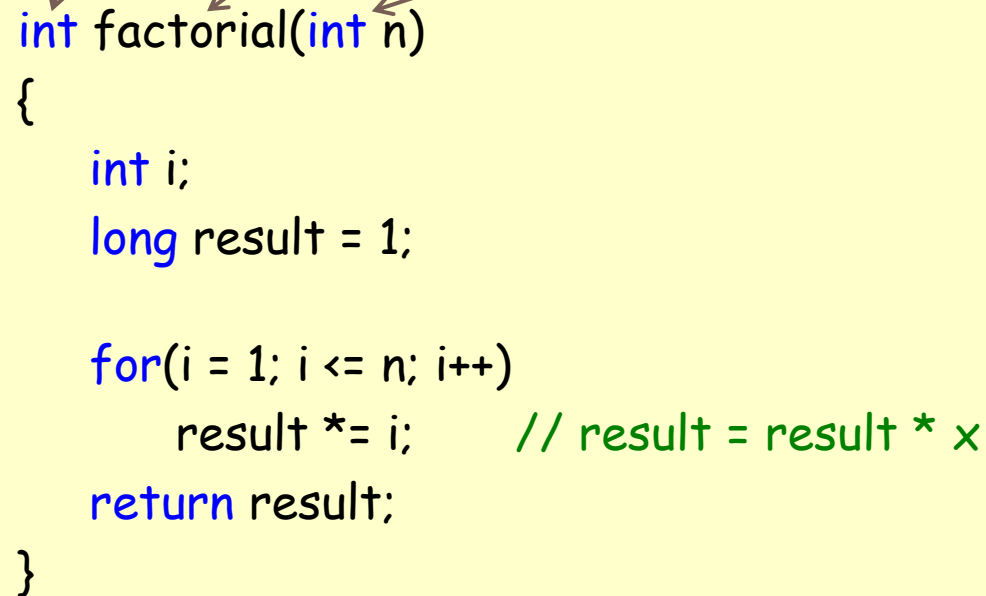
    for(i = 0; i < y; i++)
        result *= x;
    return result;
}
```

예제 #5

22

□ 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 *= i;    // result = result * x
    return result;
}
```

Contents

23

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

8.7

함수를 사용하는 이유

함수 호출과 반환

24

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

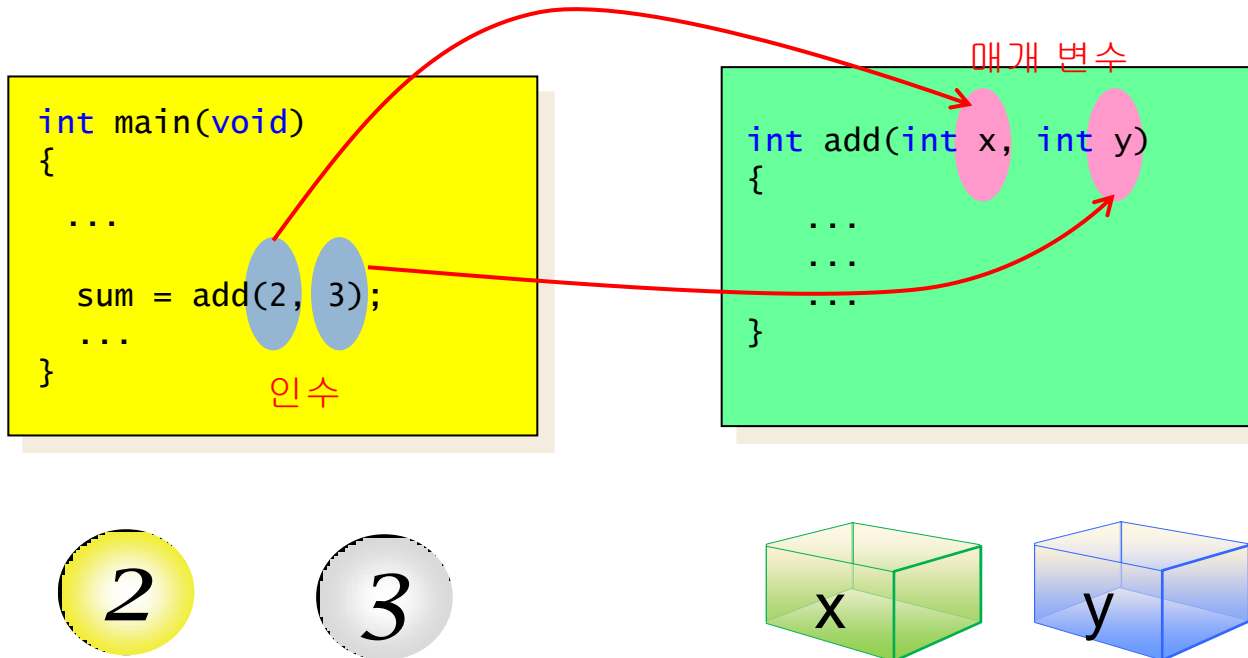
```
int main(void)
{
    ...
    sum = add(2, 3);
    ...
}
```

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


인수와 매개 변수

25

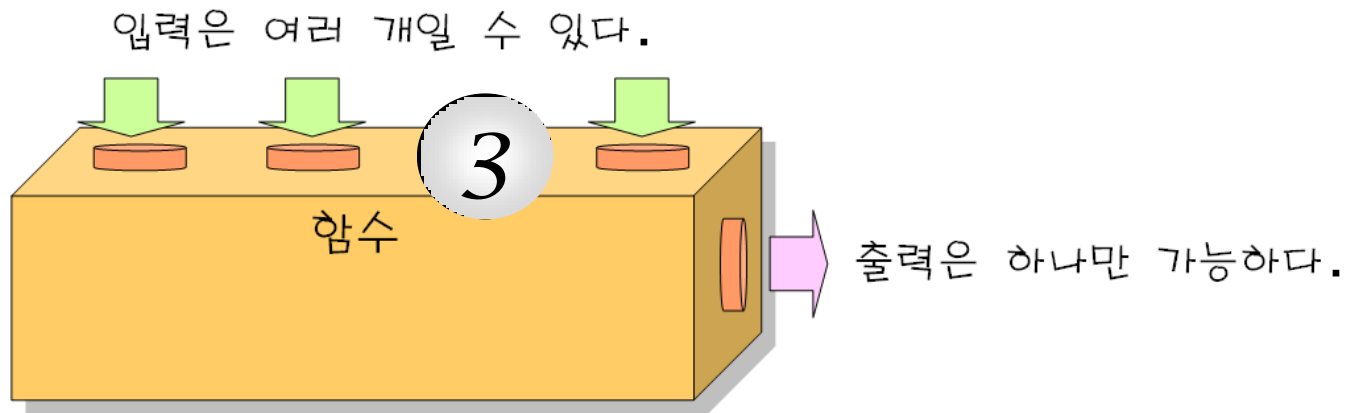
- 인수(argument): 실인수, 실매개 변수라고도 한다.
- 매개 변수(parameter): 형식 인수, 형식 매개 변수라고도 한다.



반환값

26

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



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

반환값

27

// 정수의 제곱을 계산하는 함수 예제

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

```
int square(int n);
```

```
int main(void)
```

```
{
```

```
    int result;
```

```
    result = square(5);
```

```
    printf("%d ", result);
```

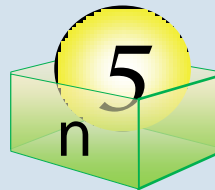
```
}
```

```
int square(int n)
```

```
{
```

```
    return(n * n);
```

```
}
```



반환값

28

/ 두수 중에서 큰 수를 찾는 함수 예제

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

```
int get_max(int x, int y);
```

```
int main(void)
```

```
{
```

```
    int a, b;
```

```
    printf("두개의 정수를 입력하시오: ");
```

```
    scanf("%d %d", &a, &b);
```

```
    printf("두수 중에서 큰 수는 %d입니다.", get_max(a, b));
```

```
    return 0;
```

```
}
```

```
int get_max(int x, int y)
```

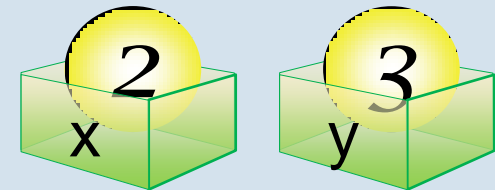
```
{
```

```
    if( x > y ) return(x);
```

```
    else return(y);
```

```
}
```

두개의 정수를 입력하시오: 2 3
두 수 중에서 큰 수는 3입니다.



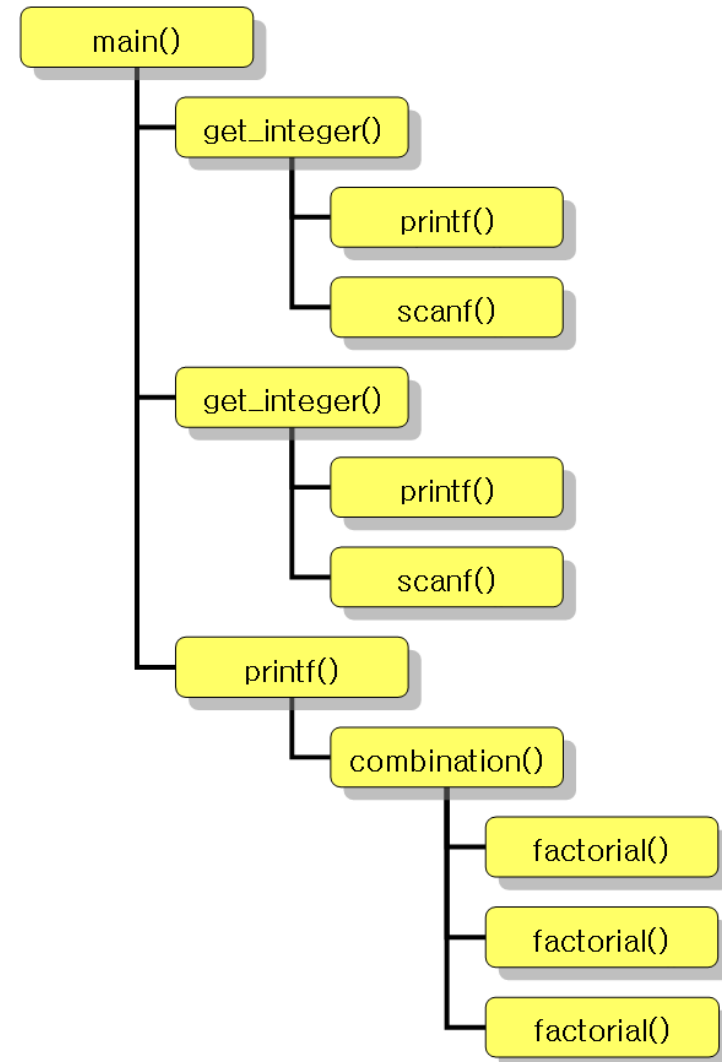
조합(combination) 계산 함수

29

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

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

- 팩토리얼 계산 함수와 get_integer() 함수를 호출하여 조합을 계산한다



예제

30

```
#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

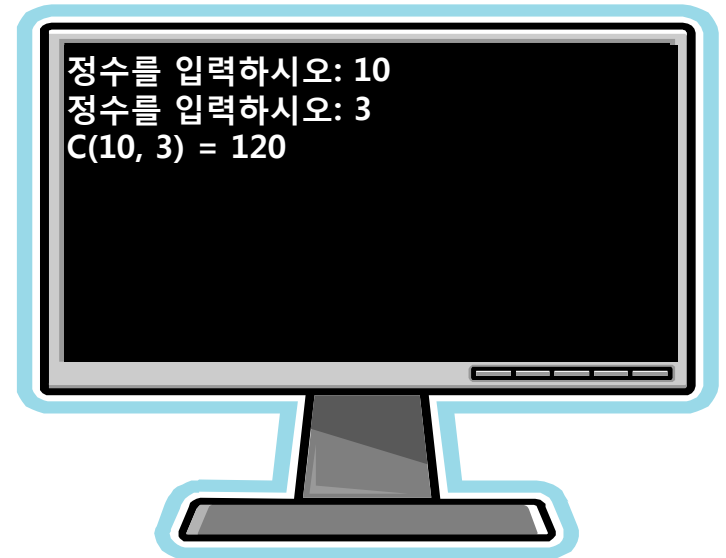
31

```
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 *= i; // result = result * i
    return result;
}
```



실습: 소수 찾기

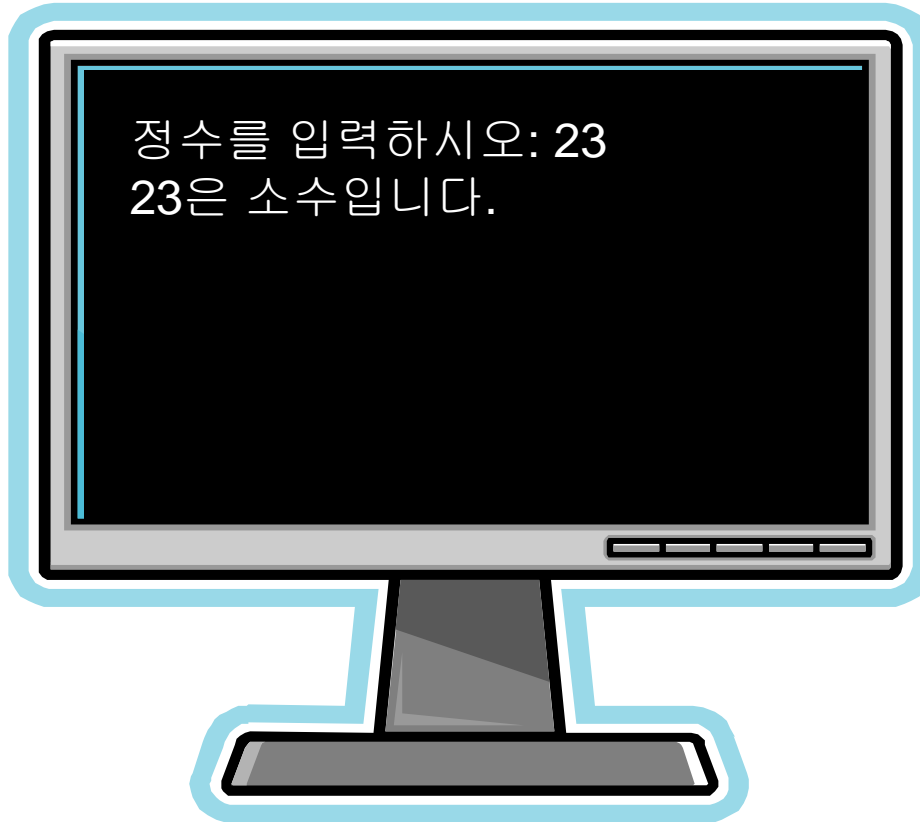
32

- 주어진 숫자가 소수(prime)인지를 결정하는 프로그램이다.
- 양의 정수 n 이 소수가 되려면 1과 자기 자신만을 약수로 가져야 한다.
- 암호학에서 많이 사용

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

실행결과

33



알고리즘

34

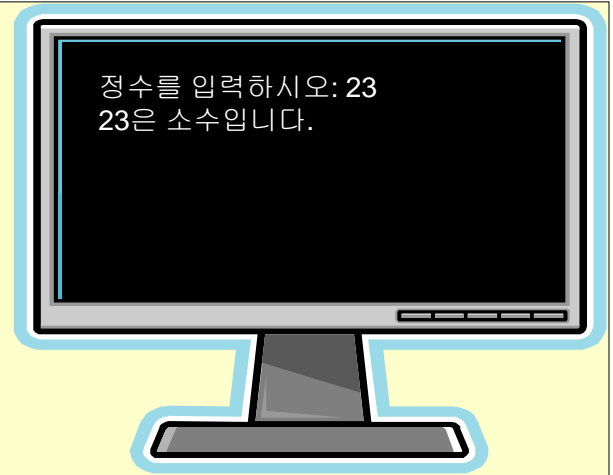
- 사용자로부터 정수를 입력받아서 변수 n 에 저장한다.
- 약수의 개수를 0으로 초기화한다.
- `for(i=1; i<=n ; i++)`
 - n 을 i 로 나누어서 나머지가 0인지 본다.
 - 나머지가 0이면 약수의 개수를 증가한다.
- 약수의 개수가 2이면 정수 n 은 소수이다.



```
#include <stdio.h>
int is_prime(int);
int get_integer(void);
main()
{
    int n, result;
    n = get_integer();
    result = is_prime(n);
    if ( result == 1 )
        printf("%d은 소수입니다.\n", n);
    else
        printf("%d은 소수가 아닙니다.\n", n);
    return 0;
}
```

```
int get_integer(void)
{
    int n;
    printf("정수를 입력하시오: ");
    scanf("%d", &n);
    return n;
}

int is_prime(int n)
{
    int divisors = 0, i;
    for ( i = 1 ; i <= n ; i++ )
    {
        if ( n%i == 0 )
            divisors++;
    }
    return (divisors == 2);
}
```



도전문제

37

- `is_prime()` 함수의 실행 속도를 빠르게 하기 위하여 어떤 코드를 추가할 수 있는지 생각해보자. 현재 버전은 검사하는 숫자가 매우 크면 비효율적이다. 예를 들어서 1,000,000에 대하여 호출되면 백만 번 반복을 하여야 한다. 한 가지 방법은 1보다 크고 n 보다 작은 숫자 중에서 약수가 하나라도 발견되면 이미 n 은 소수가 아니라고 생각하는 것이다. 이것을 코드로 작성하여 추가하여 보자.

Contents

38

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

8.7

함수를 사용하는 이유

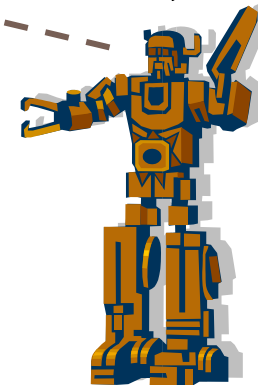
함수 원형

39

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

```
int compute_sum(int n);  
int main(void)  
{  
    int sum;  
    sum = compute_sum(100);  
    printf("sum=%d \n", sum);  
}  
int compute_sum(int n)  
{  
    int i;  
    int result = 0;  
    for(i = 1; i <= n; i++)  
        result += i;  
    return result;  
}
```

compute_sum()
은 함수
이름이었지...



컴파일러

함수 원형의 형식

40

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

반환형 함수이름(매개변수1, 매개변수2, ...);

- (예)

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

- (예)

```
int get_integer(void);  
int combination(int, int);
```

자료형만 적어주어도 됨!

함수 원형을 사용하지 않는 예제

41

```
int compute_sum(int n)
{
    int i;
    int result = 0;
    for(i = 1; i <= n; i++)
        result += i;
    return result;
}
```

```
int main(void)
{
    int sum;
    sum = compute_sum(100);
    printf("sum=%d \n", sum);
}
```

함수 정의가 함수 호출보다 먼저 오면 함수 원형을 정의하지 않아도 된다.

그러나 일반적인 방법은 아니다.

함수 원형과 헤더 파일

42

- 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

Contents

43

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

8.7

함수를 사용하는 이유

라이브러리 함수

44

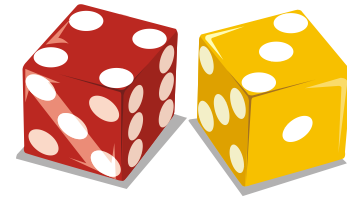
- 라이브러리 함수(library function): 컴파일러에서 제공하는 함수
 - ▣ 표준 입출력
 - ▣ 수학 연산
 - ▣ 문자열 처리
 - ▣ 시간 처리
 - ▣ 오류 처리
 - ▣ 데이터 검색과 정렬



난수 함수

45

- 난수(random number)는 규칙성이 없이 임의로 생성되는 수이다.
- 난수는 암호학이나 시뮬레이션, 게임 등에서 필수적이다.
- rand()
 - ▣ 난수를 생성하는 함수
 - ▣ 0부터 RAND_MAX까지의 난수를 생성



46

-
- A cartoon illustration of a man with a large, bulbous nose and a wide, toothy grin. He has a dollar sign (\$) on his forehead. He is wearing a white lab coat and holding a large stack of pink banknotes. The background is a bright blue cloud-like shape with black outlines and several black 'X' marks, suggesting a bright or intense light source.

내 번호 당첨조회

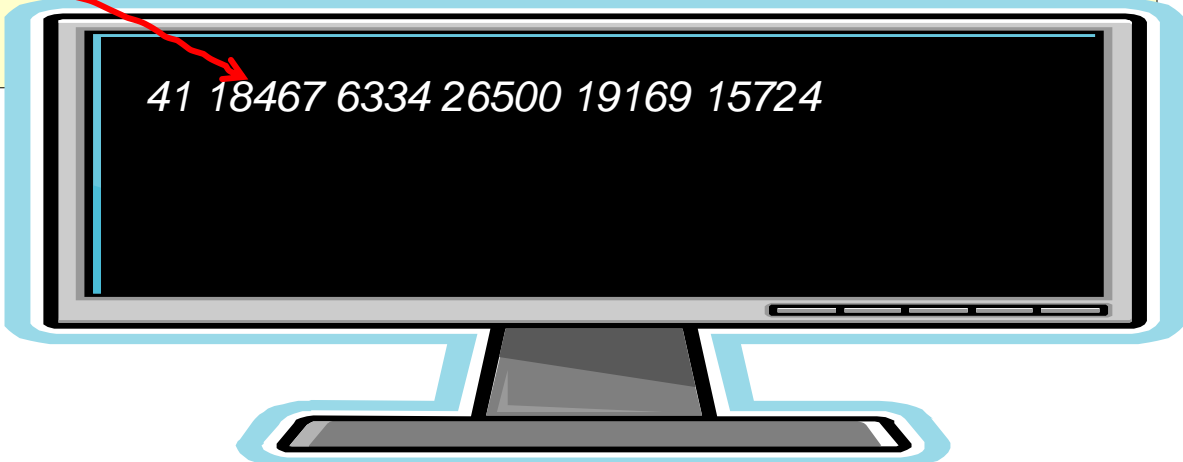
실습 코드

47

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i;
    for(i = 0; i < 6; i++)
        printf("%d ", rand());

    return 0;
}
```

0에서 32767 사이의 정수로 생성

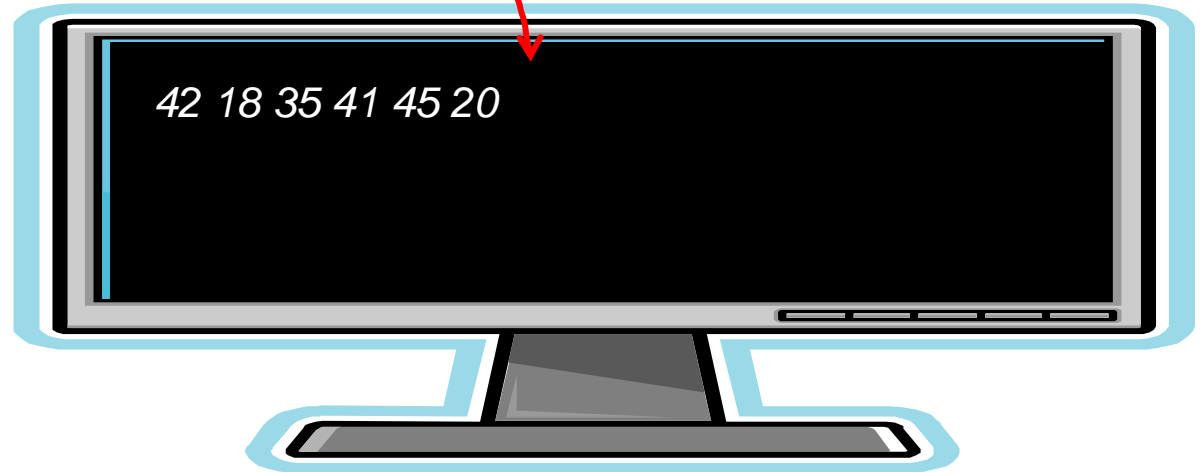


41 18467 6334 26500 19169 15724

1부터 45 사이로 제한

48

□ `printf("%d ", 1+(rand()%45));`



□ 하지만 실행할 때마다 항상 똑같은 난수가 발생된다.

실행할 때마다 다르게 하려면

49

- 매번 난수를 다르게 생성하려면 시드(seed)를 다르게 하여야 한다.

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

#define MAX 45
int main( void )
{
    int i;
    srand( (unsigned)time( NULL ) );
    for( i = 0; i < 6; i++ )
        printf("%d ", 1+rand()%MAX );
    return 0;
}
```

시드를 설정하는 가장 일반적인 방법은 현재의 시각을 시드로 사용하는 것이다. 현재 시각은 실행할 때마다 달라지기 때문이다.

실습: 자동차 게임


50

- 난수를 이용하여서 자동차 게임을 작성
- 사용자가 키를 누를 때마다 1초씩 주행하도록 하자.
- 주행 거리는 난수로 결정된다.



실행 결과

51



```
CAR #1:****  
CAR #2:  
-----  
CAR #1:*****  
CAR #2:****  
-----  
CAR #1:*****  
CAR #2:*****  
-----  
CAR #1:*****  
CAR #2:*****  
-----  
CAR #1:*****  
CAR #2:*****  
-----  
CAR #1:*****  
CAR #2:*****  
-----  
CAR #1:*****  
CAR #2:*****  
-----
```

알고리즘

52

- 난수 발생기를 초기화한다
- `for(i=0; i<주행시간; i++)`
 - ▣ 난수를 발생하여서 자동차1의 주행거리에 누적한다.
 - ▣ 난수를 발생하여서 자동차2의 주행거리에 누적한다.
 - ▣ `disp_car()`를 호출하여서 자동차1을 화면에 *표로 그린다.
 - ▣ `disp_car()`를 호출하여서 자동차2을 화면에 *표로 그린다.



53

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
void disp_car(int car_number, int distance);

int main(void)
{
    int i;
    int car1_dist=0, car2_dist=0;

    srand( (unsigned)time( NULL ) );

    for( i = 0; i < 6; i++ ) {
        car1_dist += rand() % 100;
        car2_dist += rand() % 100;
        disp_car(1, car1_dist);
        disp_car(2, car2_dist);
        printf("-----\n");
        getch();
    }
    return 0;
}
```

rand()를 이용하여서 난수를 발생한다. 난수의 범위는 %연산자를 사용하여 0에서 99로 제한하였다.



```
void disp_car(int car_number, int distance)
{
    int i;
    printf("CAR #d:", car_number);
    for( i = 0; i < distance/10; i++ ) {
        printf("*");
    }
    printf("\n");
}
```

도전문제

55

- 위의 프로그램을 참고하여서 숫자야구 게임을 작성해보자. 숫자 야구 게임은 1~9 까지의 숫자 중에서 3개를 뽑아서 문제를 낸다. 단 숫자가 중복되면 안된다.

예를 들어 029라고 하자. 사용자는 이 숫자를 맞추게 된다. 각 자리수와 숫자가 모두 일치하면 스트라이크, 숫자만 맞으면 볼이라고 출력한다.

029 vs 092 -> 1스트라이크 2볼

유틸리티 함수

56

함수	설명
exit(int status)	exit()를 호출하면 호출 프로세스를 종료시킨다.
int system(const char *command)	system()은 문자열 인수를 운영 체제의 명령어 셸에 게 전달하여서 실행시키는 함수이다.

```
#include <stdlib.h>
#include <stdio.h>
int main( void )
{
    system("dir");
    printf("아무 키나 치세요\n");
    getch();
    system("cls");
    return 0;
}
```

```
[ 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 507A-3B27
c:\source\chapter02\hello\hello 디렉터리
2011-11-28 오후 04:32 <DIR> .
2011-11-28 오후 04:32 <DIR> ..
2011-11-16 오전 11:01 20 binary.bin
...
4개 파일 5,296 바이트
3개 디렉터리 69,220,450,304 바이트 남음
아무 키나 치세요
```


수학 라이브러리 함수

57

분류	함수	설명
삼각함수	<code>double sin(double x)</code>	사인값 계산
	<code>double cos(double x)</code>	코사인값 계산
	<code>double tan(double x)</code>	탄젠트값 계산
역삼각함수	<code>double acos(double x)</code>	역코사인값 계산 결과값 범위 $[0, \pi]$
	<code>double asin(double x)</code>	역사인값 계산 결과값 범위 $[-\pi/2, \pi]$
	<code>double atan(double x)</code>	역탄젠트값 계산 결과값 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>	실수 x의 절대값
	<code>int abs(int x)</code>	정수 x의 절대값
	<code>double pow(double x, double y)</code>	x^y
	<code>double sqrt(double x)</code>	\sqrt{x}

예제

58

// 삼각 함수 라이브러리

#include <math.h>

#include <stdio.h>

여러 수학 함수들을 포함하는 표준
라이브러리

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);

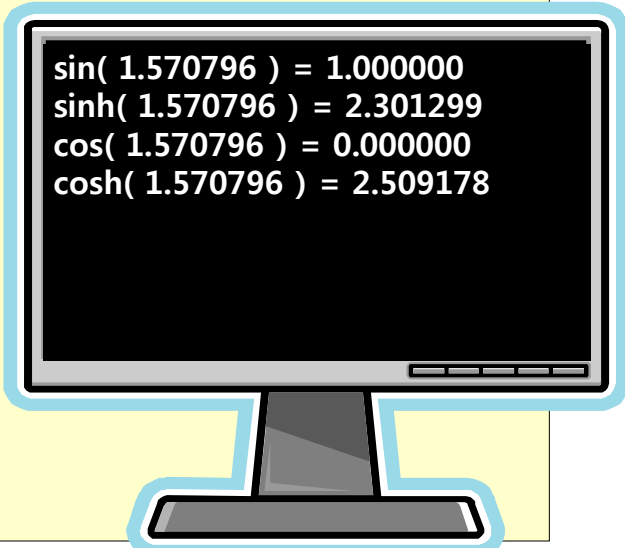
y = cos(x);

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
```

예제

59

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

#define RAD_TO_DEG (45.0/atan(1))

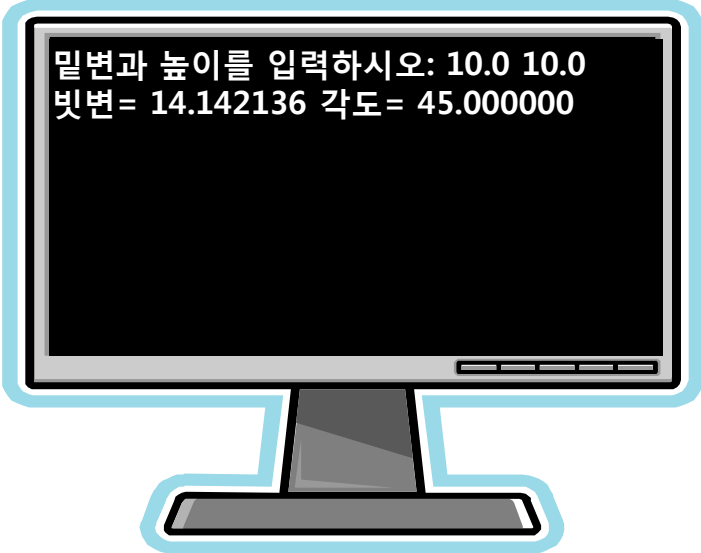
int main(void)
{
    double w, h, r, theta;

    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;
}
```

상수를 정의하는 전처리
명령문



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

수학 라이브러리 함수들

60

- `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 `xy` 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`

수학 라이브러리 함수들

61

- `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`

Contents

62

8.1

함수란?

8.2

함수 정의

8.3

함수정의 예제

8.4

함수 호출과 반환

8.5

함수 원형

8.6

라이브러리 함수

8.7

함수를 사용하는 이유

함수를 사용하는 이유

63

- 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.