



학습자가 강의 저작물을 다운로드·캡처 받아 **외부로 유출하는 행위**는 저작권자의
이용허락 없이 저작물을 복제·공중송신 또는 배포 하는 것으로 **저작권 침해 행위**에 해당함.

C 프로그래밍

(001/002)

제 6 강

신 한 대 학 교
소프트웨어융합학과
교수 송 진 희

C 프로그래밍

제 6 강

- 함수(Function)
 - 함수의 개념
 - 함수의 선언과 호출
 - 매개변수와 실인수

학습 목표

- 함수의 개념과 사용 이유를 설명할 수 있다.
- 함수 처리를 위한 헤더와 바디(Body)를 선언할 수 있다.
- 함수 실행에 필요한 매개변수 선언과 실 인수와의 연계 관계를 설명할 수 있다.
- 선언된 함수를 호출해서 사용할 수 있다.

5강 – 정리 요약

○반복 실행문

- **for**(초기값; 최종값; 증감식)
- **while**(조건식) : 조건식이 참일 경우에만 반복 실행
- **do~while()** : 1회는 반복 수행한 후, 조건을 체크해서 참이면 다시 반복 실행

○반복 구조에서 제어 빠져나가기

- **break**문
- **continue**문

○난수 발생 함수의 사용

- **rand()** : 프로그램 실행할 때마다 동일한 난수 발생
- **srand()** : seed 값에 따라 발생하는 난수가 상이함

예제 #2 : 대소문자 변환 처리

```
// 소문자를 대문자로 변경한다.  
#include <stdio.h>  
  
int main(void)  
{  
    char letter;  
  
    while(1)  
    {  
        printf("소문자를 입력하시오: ");  
        scanf(" %c", &letter);  
  
        if( letter == 'Q' )  
            break ;  
        if( letter < 'a' || letter > 'z' )  
            continue ; //소문자가 아니면  
  
        letter -= 32;  
        printf("변환된 대문자는 %c입니다.\n", letter);  
    }  
  
    return 0;  
}
```

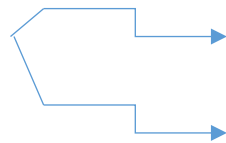
소문자를 입력하시오: a
변환된 대문자는 A입니다.
소문자를 입력하시오: b
변환된 대문자는 B입니다.
소문자를 입력하시오: c
변환된 대문자는 C입니다.
소문자를 입력하시오: Q

함수(Function)

○ 프로그램에서 반복 처리되는 기능을 독립시켜 이름(함수명)을 부여해서 사용

- 반복 처리를 위한 코드를 한 번만 기술한 후, 함수명으로 호출해서 사용 가능

➤ 예) print_char(a)



// '*'을 10번 출력

AAAAAAAAAA

// 'A'을 10번 출력

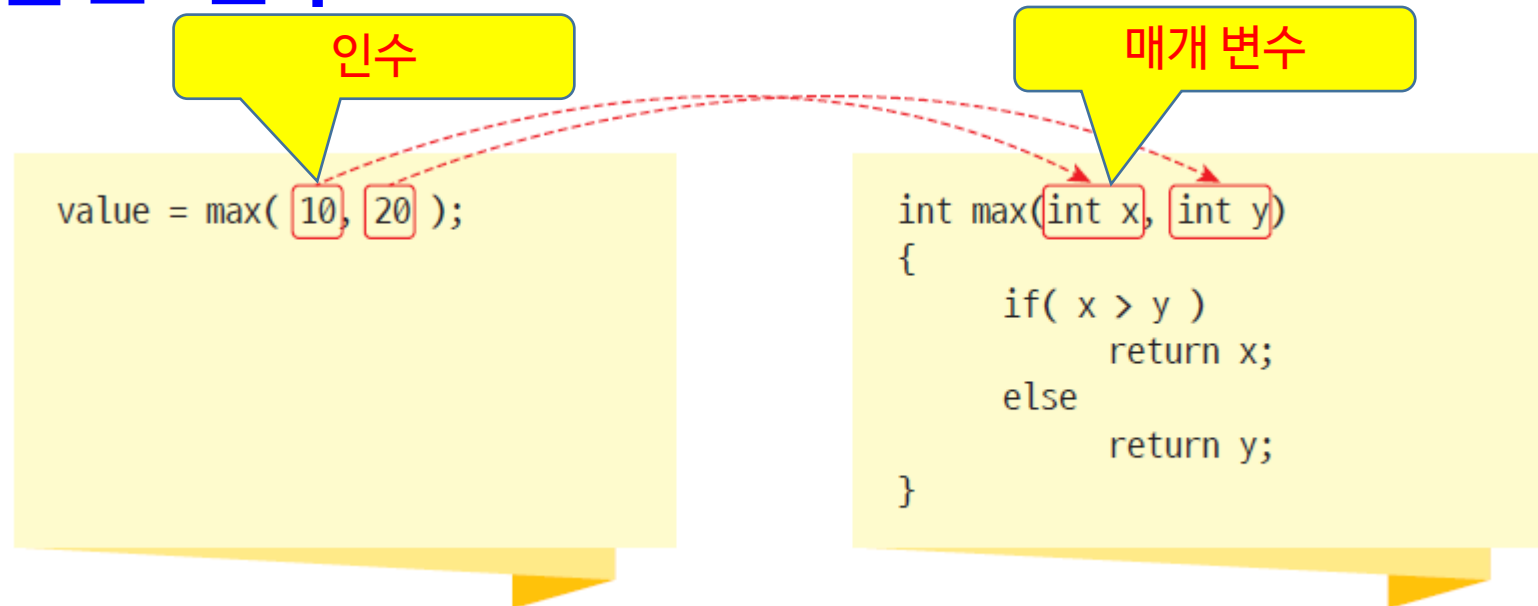
- 소스 코드의 중복 기술을 방지하고, 한 번 작성된 함수를 필요할 때 마다 재사용 가능

○ 함수 처리에 필요한 입력 자료를 전달받아 처리 가능

○ 함수의 처리 결과를 호출한 소스 프로그램으로 반환 가능

실인수와 매개변수

- 인수(argument)는 호출 프로그램에 의하여 함수에 실제로 전달되는 값
- 매개 변수(parameter)는 호출한 프로그램에서 전달한 값을 전달받는 변수



실인수와 매개 변수

○매개 변수가 없는 경우 :

- print_char(void)
- print_char()

○ 실인수와 매개 변수의 개수는 정확히 일치

```
max(10);    // max() 함수를 호출할 때는 인수가 두개이어야 한다.  
max( );    // max() 함수를 호출할 때는 인수가 두개이어야 한다.
```

○ 함수의 종류

- 1) 사용자 정의 함수 : 프로그래머가 직접 만들어서 사용하는 함수
- 2) 라이브러리 함수 : 시스템에서 기본적으로 제공(sprintf(), scanf(), ...)

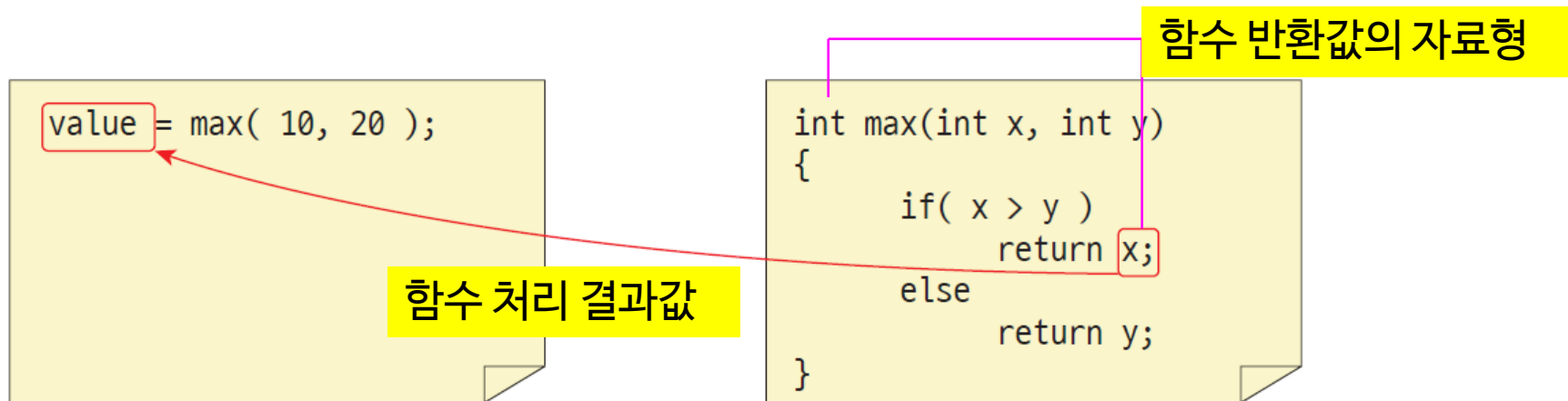
함수의 반환값(Return Value)

○반환값(return value)은

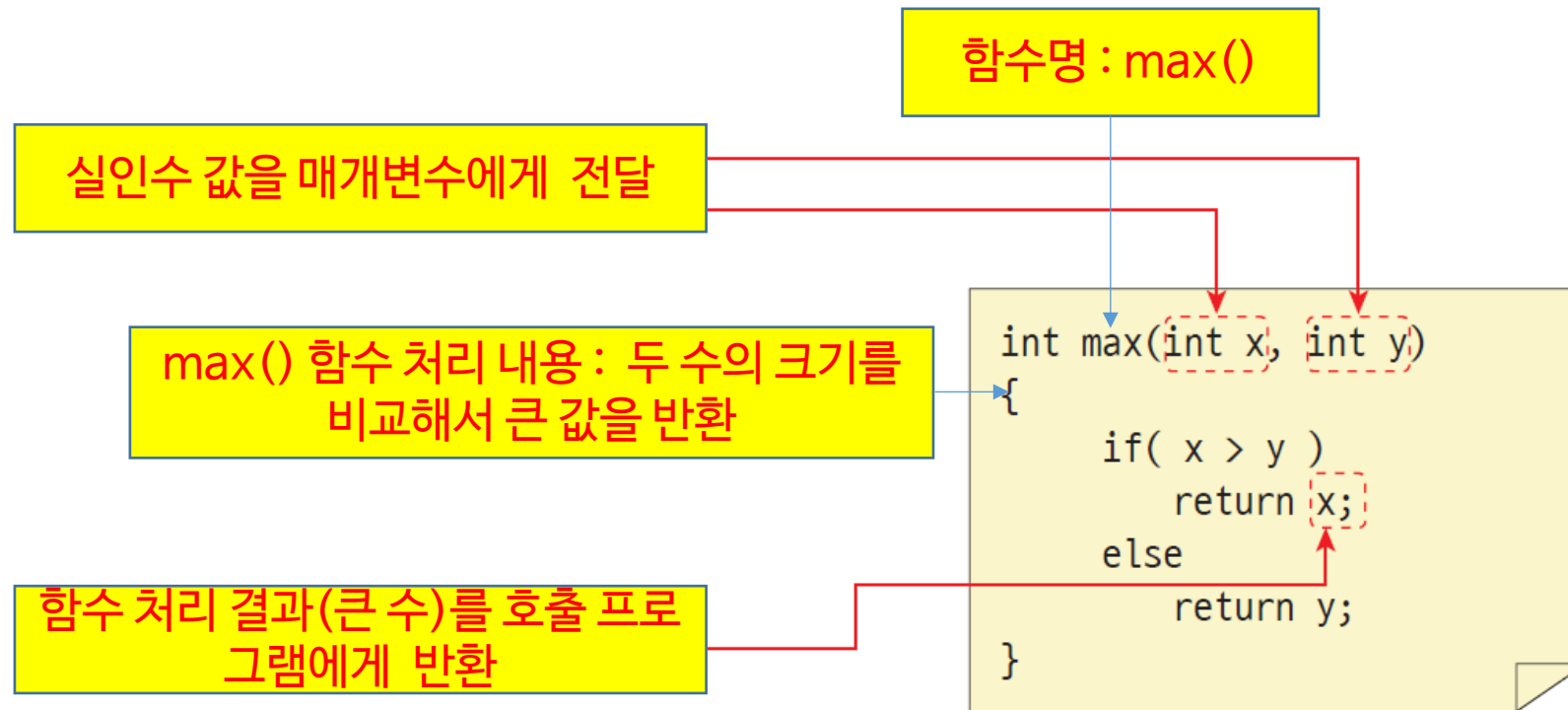
- 함수를 호출한 곳으로 함수 처리의 결과를 반환하는 값

○return 명령문

- return 문 다음에 수식 또는 값을 기술해서 함수명으로 반환



매개 변수와 반환값



예제

○ 두 개의 숫자를 전달받아 큰 수를 반환하는 max() 함수를 선언하고, 사용자가 입력한 값 중에서 더 큰 값을 출력하는 프로그램을 작성하시오.

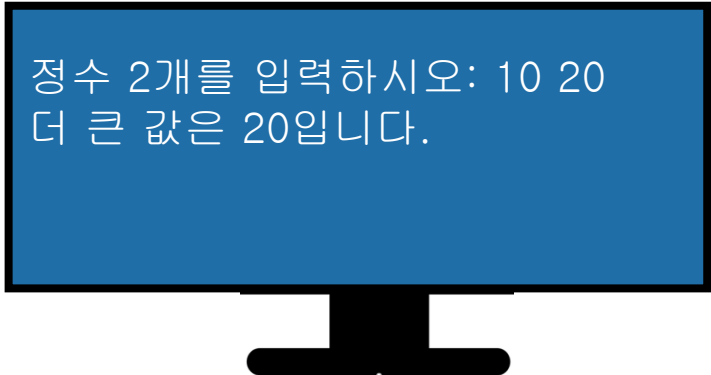
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

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

int main(void)
{
    int x, y, larger;

    printf("정수 2개를 입력하시오: ");
    scanf("%d %d", &x, &y);

    larger = max(x, y);
    printf("더 큰 값은 %d입니다. \n", larger);
    return 0;
}
```



정수 2개를 입력하시오: 10 20
더 큰 값은 20입니다.

예제

- 함수명 : `get_integer()`
- 함수 처리 내용 :
 - 정수 입력 안내 메시지를 출력
 - 사용자가 입력한 정수를 한 개 입력 받기
 - 사용자가 입력한 정수를 반환

```
// 사용자로부터 정수를 받는 함수
#include <stdio.h>

int get_integer(void)
{
    int value;

    printf("정수를 입력하십시오: ");
    scanf("%d", &value);

    return value;
}
```

예제

○ 앞의 예에서 작성한 `get_integer()`까지 사용하여 사용자로부터 받은 두 정수의 합을 계산하여 출력하는 함수 `add()` 작성하기.

```
#include <stdio.h>
//
int get_integer()
{
    int value;

    printf("정수를 입력하시오: ");
    scanf("%d", &value);
    return value;
}

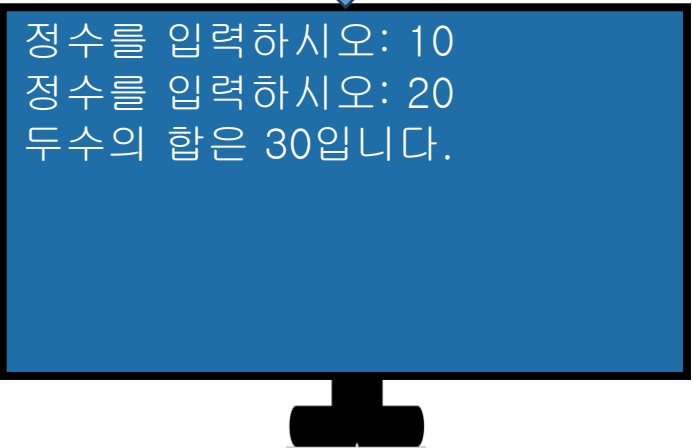

//
int add(int x, int y)
{
    return x + y;
}
```

```
int main(void)
{
    int sum;

    int x = get_integer();
    int y = get_integer();

    sum = add(x, y);
    printf("두수의 합은 %d입니다. \n", sum);

    return 0;
}
```



정수를 입력하시오: 10
정수를 입력하시오: 20
두수의 합은 30입니다.

실습 #1

○정수 n 을 받아서 1에서 n 까지의 정수들의 곱을 구하는 팩토리얼 함수를 작성하여 보자.

```
#include <stdio.h>

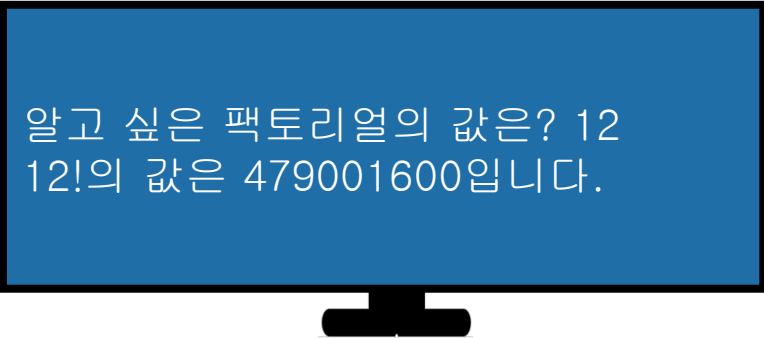
int factorial(int n)
{
    int result = 1;

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

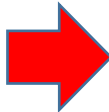
int main(void)
{
    int n;

    printf("알고 싶은 팩토리얼의 값은? ");
    scanf("%d", &n);
    printf("%d!의 값은 %d입니다. \n", n, factorial(n));
    return 0;
}
```

$$n! = n * (n-1) * (n-2) * \dots * 1$$



알고 싶은 팩토리얼의 값은? 12
12!의 값은 479001600입니다.

- 
- 1) While() 문을 이용해서 반복 실행
 - 2) 0이 입력되면 프로그램 실행이 종료 되도록 수정 하시오.

실습 #1

○while()문으로 변경하기

$$n! = n * (n-1) * (n-2) * \dots * 1$$

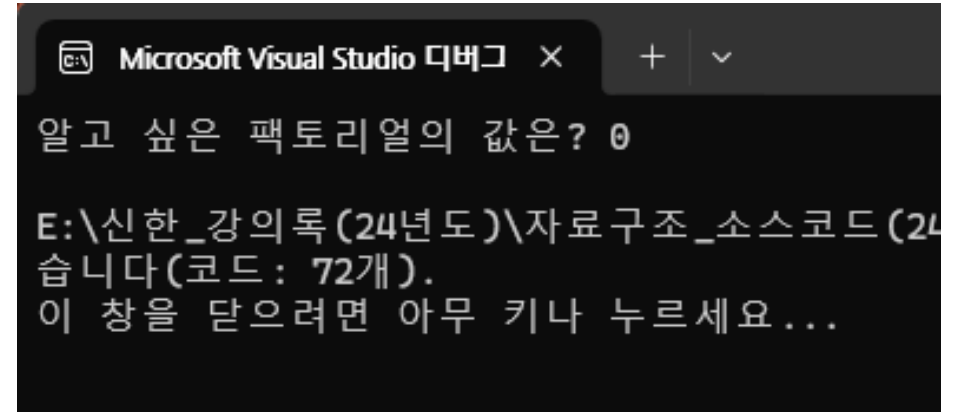
```
#include <stdio.h>

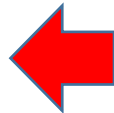
int factorial(int n)
{
    int result = 1;

    int i = 1;
    while(i <= n)
    {
        result *= i;           // result = result * i
        i++;
    }
    return result;
}

int main(void)
{
    int n;

    printf("알고 싶은 팩토리얼의 값은? ");
    scanf("%d", &n);
    if(n == 0) exit();         //프로그램 종료 함수
    printf("%d!의 값은 %d입니다. \n", n, factorial(n));
    return 0;
}
```



- 
- 1) While() 문을 이용해서 반복 실행
 - 2) 0이 입력되면 프로그램 실행이 종료 되도록 수정 하시오.

실습#2 : 온도 변환기

○ 섭씨 온도를 화씨 온도로, 또 그 반대로 변환하는 프로그램 작성해보자.

```
=====
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
=====
메뉴에서 선택하세요: f
화씨온도: 100
섭씨온도: 37.77778
=====
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
=====
메뉴에서 선택하세요:
```

예제

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

```
void printMenu()
```

```
{  
    printf("=====\\n");  
    printf("'c' 섭씨온도에서 화씨온도로 변환\\n");  
    printf("'f' 화씨온도에서 섭씨온도로 변환\\n");  
    printf("'q' 종료\\n");  
    printf("=====\\n");  
}
```

```
double C2F(double c_temp)
```

```
{  
    return 9.0 / 5.0 * c_temp + 32;  
}
```

```
double F2C(double f_temp)
```

```
{  
    return (f_temp - 32.0) * 5.0 / 9.0;  
}
```

```
int main(void)
```

```
{  
    char choice;  
    double temp;  
  
    while (1) {  
        printMenu(); //함수 호출  
        printf("메뉴에서 선택하세요: ");  
        choice = getchar();  
  
        if (choice == 'q')  
            break;  
        else if (choice == 'c') {  
            printf("섭씨온도: ");  
            scanf("%lf", &temp);  
            printf("화씨온도: %lf \\n\\n", C2F(temp)); //함수 호출  
        }  
        else if (choice == 'f') {  
            printf("화씨온도: ");  
            scanf("%lf", &temp);  
            printf("섭씨온도: %lf \\n\\n", F2C(temp));  
        }  
        getchar(); // 엔터키 문자를 삭제하기 위하여 필요!  
    }  
    return 0;  
}
```

실습 #3 : 소수 찾기

- 주어진 숫자가 소수(prime)인지를 결정하는 프로그램이다.
- 양의 정수 n 이 소수가 되는 조건
 - 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

알고리즘

1. 사용자로부터 정수를 입력받아서 변수 n 에 저장한다.
2. `for(i=2; i<n ; i++)`
3. n 을 i 로 나누어서 나머지가 0인지 본다.
4. 나머지가 0이면 약수가 있는 것이므로 소수가 아니다.
5. 반복이 정상적으로 종료되고 약수가 없다면 소수이다.

정수를 입력하십시오: 12229
12229은 소수가 아닙니다.

실습 #4

○1부터 사용자가 입력한 숫자 n 사이의 모든 소수를 찾도록 위의 프로그램을 변경하여 보자. [사이버캠퍼스에 제출]

함수 원형

○아래의 코드를 컴파일하면 오류가 발생한다.

```
#include <stdio.h>

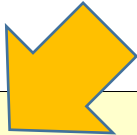
int main(void)
{
    printf("섭씨 %lf도는 화씨 %lf입니다. \n", 36.0, c_to_f(36.0));
    return 0;
}

double c_to_f(double c_temp)
{
    return 9.0 / 5.0 * c_temp + 32;
}
```

전체 솔루션		1 오류	2 경고	0 메시지	빌드 + IntelliSense	검색 오류 목록	
코드	설명	프로젝트	파일	줄	Suppress		
C4013	'c_to_f'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.	ConsoleApplication3	test.c	5			
C4477	'printf' : 서식 문자열 '%lf'에 'double' 형식의 인수가 필요하지만 variadic 인수 2의 형식이 'int'입니다.	ConsoleApplication3	test.c	5			
C2371	'c_to_f': 재정의. 기본 형식이 다릅니다.	ConsoleApplication3	test.c	9			

함수 원형

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



```
#include <stdio.h>
double c_to_f(double c_temp); // 함수 원형

int main(void)
{
    printf("섭씨 %f도는 화씨 %f입니다. \n", 36.0, c_to_f(36.0));
    return 0;
}

double c_to_f(double c_temp)
{
    return 9.0 / 5.0 * c_temp + 32;
}
```

함수 원형과 실행 오류

- 함수 원형은 함수의 이름, 매개변수, 반환형을 함수가 정의되기 전에 미리 알려주는 것
- 함수 원형은 함수 헤더에 세미콜론(;)만을 추가한 것과 동일
 - 함수 원형에서는 매개 변수의 자료형만 적으면 된다.

```
double c_to_f(double);  
int get_integer(void);
```

매개 변수의 이름은 생략하여도 된다.
반드시 끝에 ;을 붙여야 한다.

오류 예방법(1)

```
#include <stdio.h>  
  
int main(void)  
{  
    printf("섭씨 %f도는 화씨 %f입니다. \n", 36.0, c_to_f(36.0));  
    return 0;  
}  
  
double c_to_f(double c_temp)  
{  
    return 9.0 / 5.0 * c_temp + 32;  
}
```

컴파일 오류 발생

```
#include <stdio.h>  
double c_to_f(double c_temp); // 함수 원형  
  
int main(void)  
{  
    printf("섭씨 %f도는 화씨 %f입니다. \n", 36.0, c_to_f(36.0));  
    return 0;  
}  
  
double c_to_f(double c_temp)  
{  
    return 9.0 / 5.0 * c_temp + 32;  
}
```

전체 솔루션 1 오류 2 경고 0 메시지 빌드 + IntelliSense 검색 오류 목록

코드	설명	프로젝트	파일	줄	Suppress
C4013	'c_to_f'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.	ConsoleApplication3	test.c	5	
C4477	'printf' : 서식 문자열 '%f'에 'double' 형식의 인수가 필요하지만 variadic 인수 2의 형식이 'int'입니다.	ConsoleApplication3	test.c	5	
C2371	'c_to_f': 재정의의 기본 형식이 다릅니다.	ConsoleApplication3	test.c	9	

함수 원형과 실행 오류

오류 예방법 (2)

```
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.

라이브러리 함수

○ 라이브러리 함수(library function): 컴파일러에서 제공하는 함수

- 표준 입출력
- 수학 연산
- 문자열 처리
- 시간 처리
- 오류 처리
- 데이터 검색과 정렬



표준 라이브러리 함수(수학 함수)

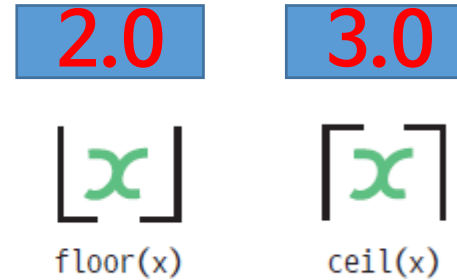
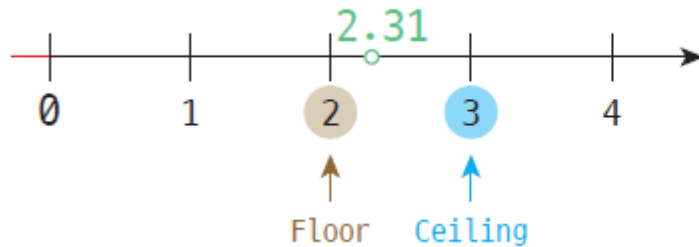
○수학 관련 함수

➤math.h

- 라이브러리 함수를 사용하면 복잡한 산술 연산을 할 수 있다.
- 수학 함수는 일반적으로 double형의 매개 변수와 반환값을 가진다.

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}

floor() / ceil() 함수



```
double result, value = 1.6;
```

```
result = floor(value);           // result는 1.00이다.  
printf("%lf ", result);
```

```
result = ceil(value);           // result는 2.00이다.  
printf("%lf ", result);
```

fabs()

○fabs() : 실수를 입력 받아서 절대값을 반환

```
printf("12.0의 절대값은 %f\\n", fabs(12.0));  
printf("-12.0의 절대값은 %f\\n", fabs(-12.0));
```



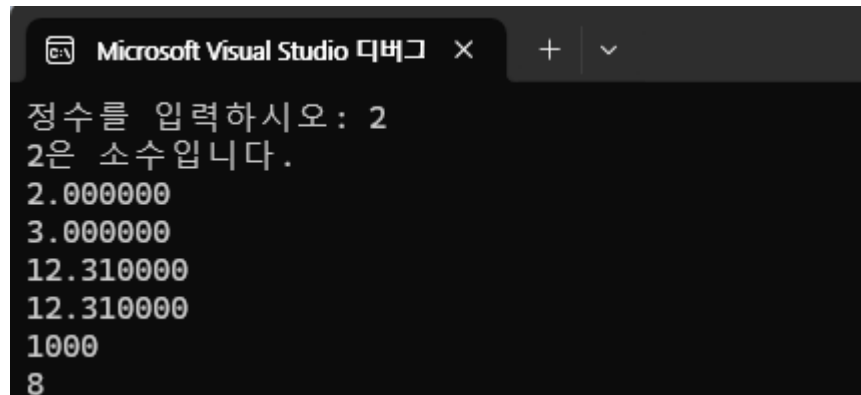
10의 3승은 1000.
16의 제곱근은 4.

○pow() / sqrt()

```
printf("10의 3승은 %.0f. \\n", pow(10.0, 3.0));  
printf("16의 제곱근은 %.0f. \\n", sqrt(64));
```

//수학 함수 활용하기

```
printf("%lf", floor(2.31));  
printf("Wn%lf", ceil(2.31));  
printf("Wn%lf", fabs(12.31));  
printf("Wn%lf", fabs(-12.31));  
printf("Wn%.0f", pow(10.0, 3.0));  
printf("Wn%.0f", sqrt(64));
```



```
Microsoft Visual Studio 디버그 x + v  
정수를 입력하시오: 2  
2은 소수입니다.  
2.000000  
3.000000  
12.310000  
12.310000  
1000  
8
```

cos(double x) / sin(double x), /tan(double x)

// 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 = cos(x);

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

}

여러 수학 함수들을 포함하는 표준 라이브러리

sin(1.570796) = 1.000000

cos(1.570796) = 0.000000

기타 함수

함수	설명
<code>exit()</code>	<code>exit()</code> 를 호출하면, 실행 중인 프로그램을 종료시킨다.
<code>system("...")</code>	<code>system()</code> 은 운영 체제의 명령 프롬프트에게 명령어를 전달하여서 실행시키는 함수이다. 예를 들어서 DOS 명령어인 DIR이나 COPY, TYPE, CLS, DEL, MKDIR와 같은 명령어들을 실행시킬 수 있다.
<code>time(NULL)</code>	현재 시각을 반환한다. 1970년 1월 1일부터 흘러온 초를 반환한다.

예제: 시간 맞추기 게임

○사용자에게 정확한 시간을 예측하게 하는 게임을 만들어보자.

- 사용자에게 10초가 지나면 엔터키를 누르라고 메시지 출력
- 사용자가 입력한 시간과 실제 경과 시간의 차이를 출력

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

```
int main(void)
{
```

```
    time_t start, end; // time_t는 unsigned long과 동일하다.
```

```
    start = time(NULL);
```

```
    printf("10초가 되면 아무 키나 누르세요\n");
```

```
    while (1) {
        if (getchar())
            break;
    }
```

```
    printf("종료되었습니다.\n");
```

```
    end = time(NULL);
```

```
    printf("경과된 시간은 %ld 초입니다. \n", end - start);
```

```
    return 0;
```

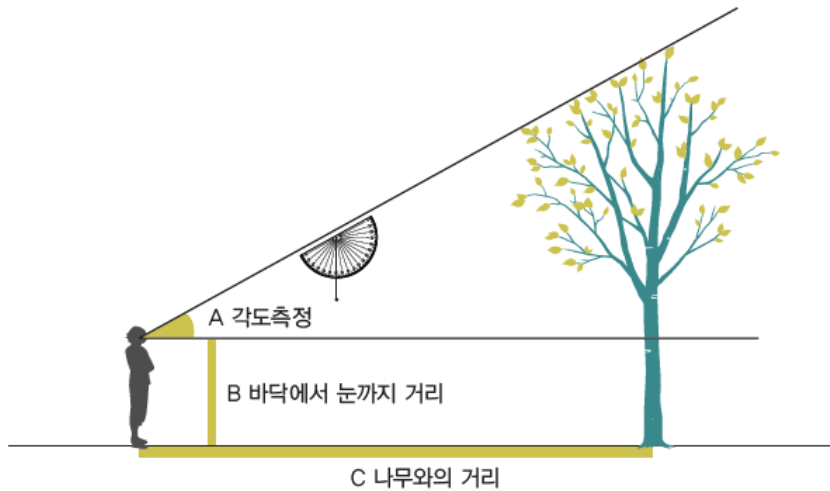
```
}
```

C 라이브러리 함수 time()은 1900년 1월 1일 이후의 시간을 초 단위로 반환합니다.

10초가 되면 엔터키를 누르세요
종료되었습니다.
경과된 시간은 6 초입니다.

예제: 나무 높이 측정

○ 각도기와 삼각 함수를 이용한 나무 높이를 측정하는 프로그램을 작성하자.



나무와의 길이(단위는 미터): 4.2
측정자의 키(단위는 미터): 1.8
각도(단위는 도): 62
나무의 높이(단위는 미터): 9.699047


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

```
int main(void)
{
    double height, distance, tree_height, degrees, radians;
```

```
    printf("나무와의 거리(단위는 미터): ");
    scanf("%lf", &distance);
```

```
    printf("측정자의 키(단위는 미터): ");
    scanf("%lf", &height);
```

```
    printf("각도(단위는 도): ");
    scanf("%lf", &degrees);
```

```
    radians = degrees * (3.141592 / 180.0);
```

```
    tree_height = tan(radians) * distance + height;
    printf("나무의 높이(단위는 미터): %lf \n", tree_height);
```

```
    return 0;
```

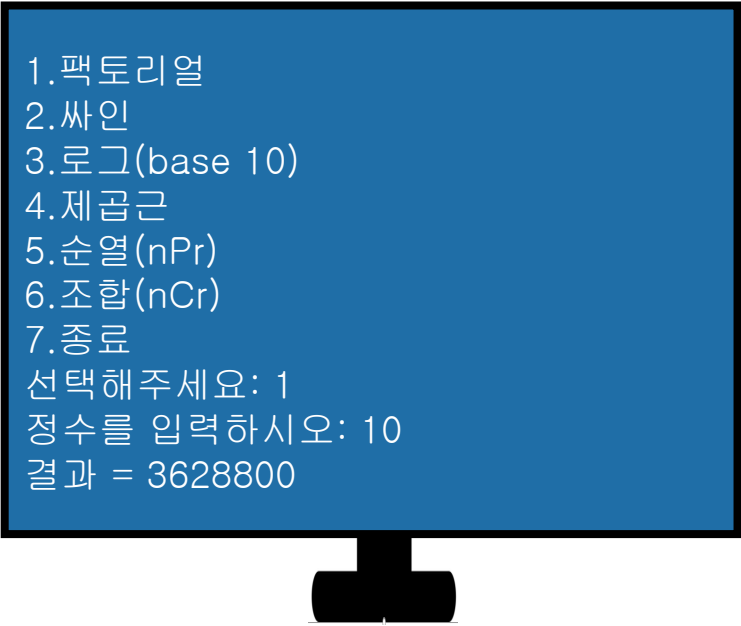
```
}
```

예제 : 공학용 계산기 만들기

○싸인값이나 코싸인값을 계산할 수 있는 공학용 계산기를 만들어보자.

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

int menu(void)
{
    int n;
    printf("1.팩토리얼\n");
    printf("2.싸인\n");
    printf("3.로그(base 10)\n");
    printf("4.제곱근\n");
    printf("5.순열(nPr)\n");
    printf("6.조합(nCr)\n");
    printf("7.종료\n");
    printf("선택해주세요: ");
    scanf("%d", &n);
    return n;
}
```



1.팩토리얼
2.싸인
3.로그(base 10)
4.제곱근
5.순열(nPr)
6.조합(nCr)
7.종료
선택해주세요: 1
정수를 입력하시오: 10
결과 = 3628800

예제 : 공학용 계산기 만들기

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

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

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

    printf("결과 = %lld\\n\\n", result);
}

void sine()
{
    double a, result;

    printf("각도를 입력하시오: ");
    scanf("%lf", &a);

    result = sin(a);
    printf("결과 = %lf\\n\\n", result);
}
```

```
void logBase10()
{
    double a, result;

    printf("실수값을 입력하시오: ");
    scanf("%lf", &a);

    if (a <= 0.0)
        printf("오류\\n");
    else {
        result = log10(a);
        printf("결과 = %lf\\n\\n", result);
    }
}
```


```
int main(void)
{
    while (1)
    {
        switch (menu()) {
            case 1:
                factorial();
                break;
            case 2:
                sine();
                break;
            case 3:
                logBase10();
                break;
            case 7:
                printf("종료합니다. \\n");
                return 0;
            default:
                printf("잘못된 선택입니다. \\n");
                break;
        }
    }
}
```

[문제1]: 동전 던지기 게임

○ 동전을 100번 던져서 앞면이 나오는 횟수와 뒷면이 나오는 횟수를 출력한다.

➤ `coin_toss()` 함수를 이용 : 동전 앞면, 뒷면 나오는 경우를 처리

```
int coin_toss( void )
```



동전의 앞면: 53
동전의 뒷면: 47

소스 코드

```
#include <stdlib.h>    //srand() 함수 사용
#include <stdio.h>
#include <time.h>      // time()함수 사용

int coin_toss(void);

int main(void)
{
    int toss;
    int heads = 0;      //동전 앞면 횟수 누적 변수 초기화
    int tails = 0;      //동전 뒷면 횟수 누적 변수 초기화

    srand((unsigned)time(NULL));

    for (toss = 0; toss < 100; toss++) {
        if (coin_toss() == 1)
            heads++;
        else
            tails++;
    }
```

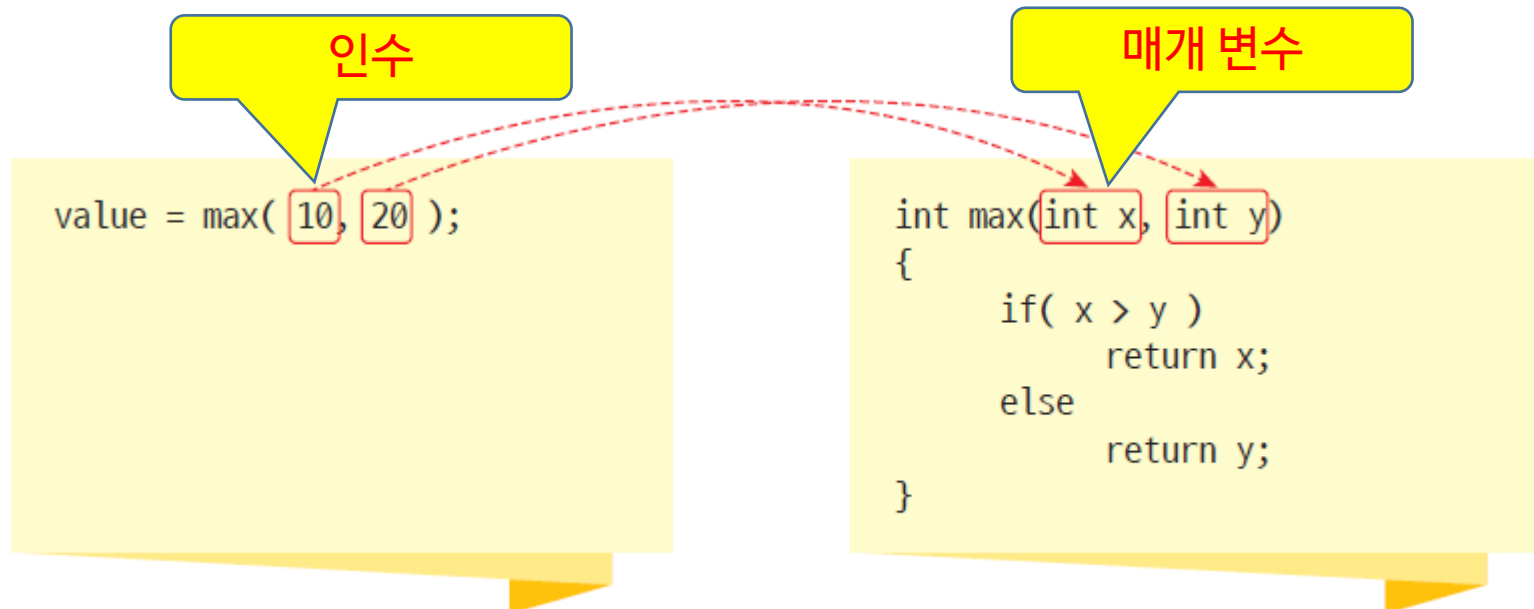
소스 코드

```
printf("동전의 앞면: %d \n", heads);  
printf("동전의 뒷면: %d \n", tails);  
return 0;  
  
}  
  
int coin_toss( void )  
{  
    int head = rand() % 2;  
    return head;  
}
```

6강 - 정리 요약

○ 함수 처리에 필요한 입력 자료를 전달받아 처리 가능

- 실 인수 : 함수에 전달되는 실제 데이터
- 매개 변수 : 함수를 호출한 프로그램에서 전달한 데이터를 수신할 변수
- 실인수와 매개 변수의 개수는 정확히 일치



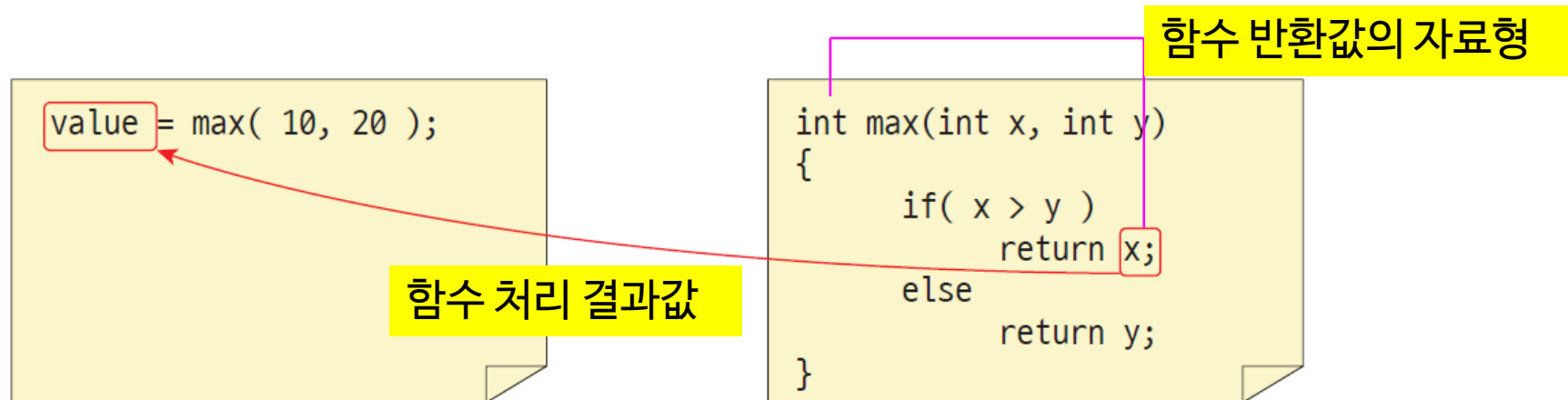
6강 - 정리 요약

○ 함수 반환값(return value)은

- ▶ 함수를 호출한 곳으로 함수 처리의 결과를 반환하는 값

○ return 명령문

- ▶ return 문 다음에 수식 또는 값을 기술해서 함수명으로 반환



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

