

C 언어 EXPRESS(개정3판)



제 8 장 함수



이번 장에서 학습할 내용

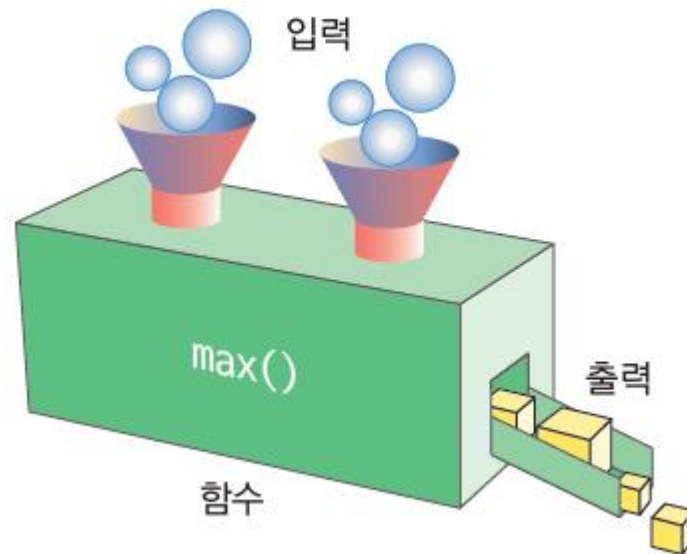


- 모듈화
- 함수의 개념, 역할
- 함수 작성 방법
- 반환값
- 인수 전달
- 함수를 사용하는 이유



함수의 개념

- 함수(function): 입력을 받아서 특정한 작업을 수행하여서 결과를 반환하는 블랙 박스(상자)와 같다





함수가 필요한 이유

- 동일한 코드가 여러 곳에서 사용된다고 하자.

비슷한 코드인데 하나로
합칠 수 있을까?



```
for(int i=0; i<30; i++)  
    printf("*");
```

```
for(int i=0; i<30; i++)  
    printf("*");
```



함수가 필요한 이유

- 함수를 작성하면 동일한 코드를 하나로 만들 수 있다.

함수를 사용하면 됩니다!



```
print_stars();
```

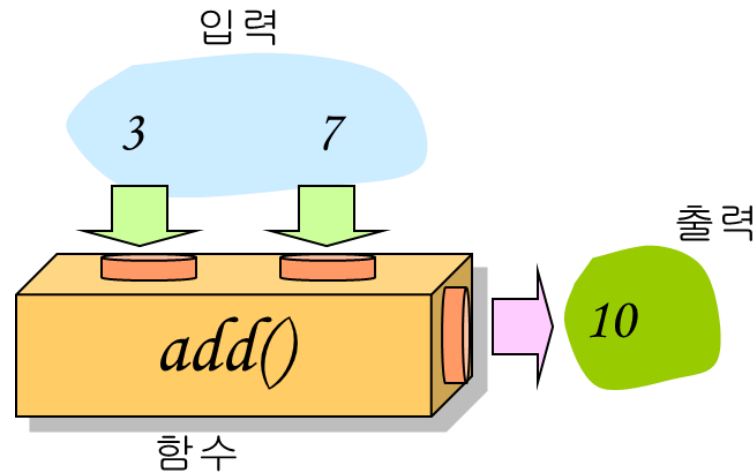
```
print_stars();
```

```
void print_stars()  
{  
    for(int i=0; i<30; i++)  
        printf("*");  
}
```



함수의 특징

- 함수는 특정한 작업을 수행하기 위한 **명령어들의 모음**이다.
- 함수는 서로 구별되는 이름을 가지고 있다.
- 함수는 특정한 작업을 수행한다.
- 함수는 **입력을 받을 수 있고 결과를 반환할 수 있다.**





함수의 장점

- 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.



함수의 종류

함수

사용자 정의 함수

라이브러리 함수



내가 원하는 함수가 없으니
직접 만들어야지!



이 함수들은 기본적으로
제공됩니다.



함수의 정의

Syntax: 함수 정의

예

`void print_stars()`

{

`for(int i=0; i<30; i++)
printf("*");`

}

반환형

함수 이름

매개 변수(현재는 없다)

함수 몸체



함수 정의

- 반환형
 - 반환형은 함수가 처리를 종료한 후에 호출한 곳으로 반환하는 데이터의 유형을 말한다.
- 함수 이름
 - 함수 이름은 식별자에 대한 규칙만 따른다면 어떤 이름이라도 가능하다.
 - 함수의 기능을 암시하는 (동사+명사)를 사용하면 좋다.

```
int square()  
double compute_average()  
void set_cursor_type()
```

// 정수를 제공하는 함수
// 평균을 구하는 함수
// 커서의 타입을 설정하는 함수

함수 이름

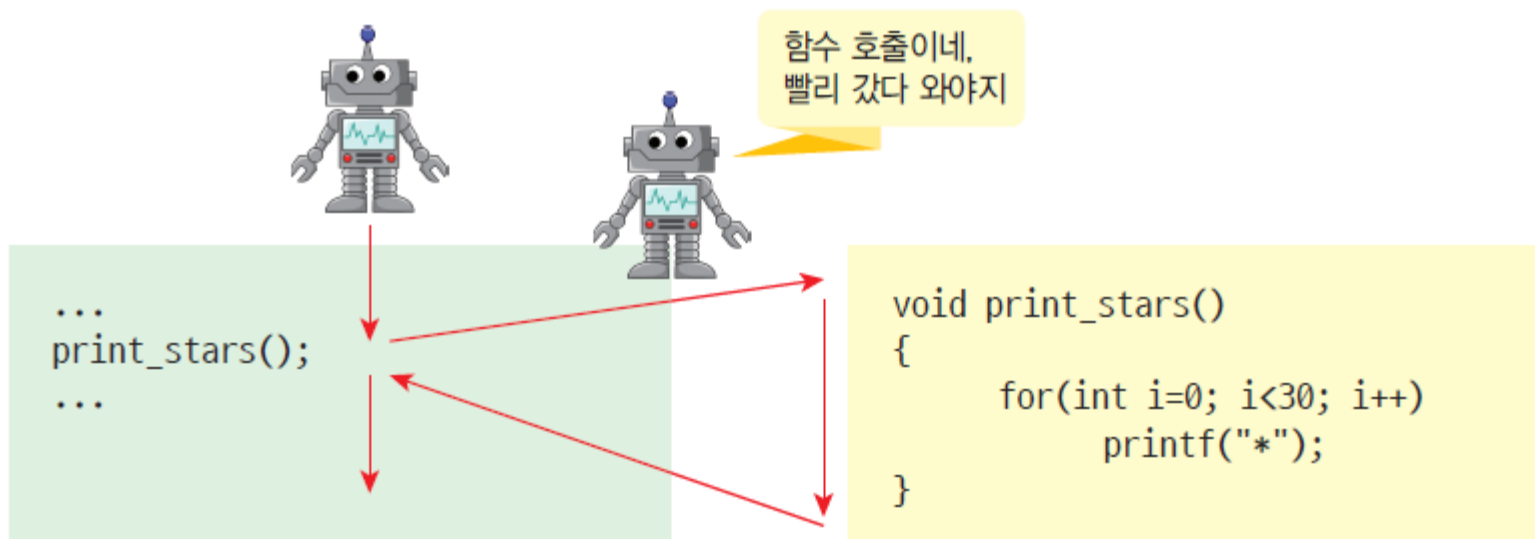


함수 호출

- 함수를 사용하려면 함수를 호출(call)하여야 한다.
- 함수 호출(function call)이란 `print_stars()`와 같이 함수의 이름을 써주는 것이다.
- 함수 안의 문장들은 호출되기 전까지는 전혀 실행되지 않는다. 함수를 호출하게 되면 현재 실행하고 있는 코드는 잠시 중단되고, 호출된 함수로 이동하여서 함수 몸체 안의 문장들이 순차적으로 실행된다.
- 호출된 함수의 실행이 끝나면 호출한 위치로 되돌아가서 잠시 중단되었던 코드가 실행을 재개한다.



함수 호출





함수는 여러 번 호출될 수 있다.



```
...  
print_stars();  
...  
print_stars();
```

```
void print_stars()  
{  
    for(int i=0; i<30; i++)  
        printf("*");  
}
```



예제

```
#include <stdio.h>

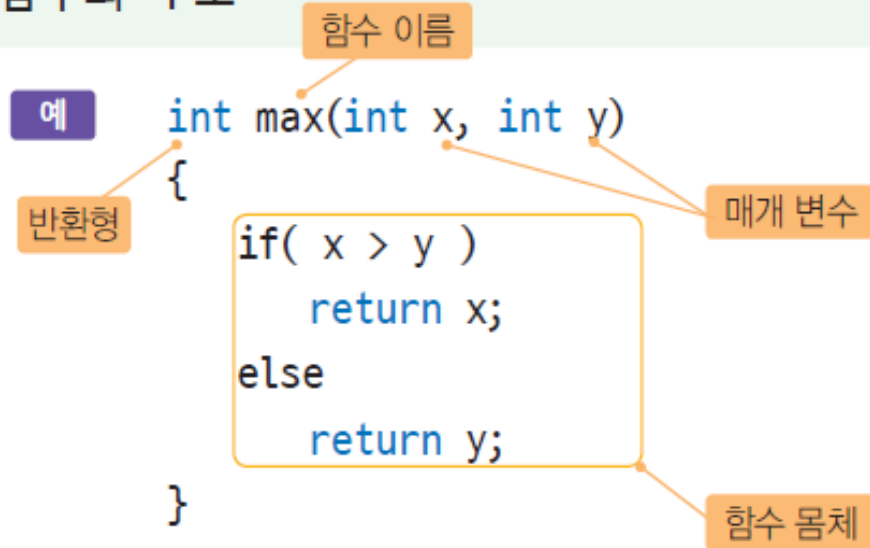
void print_stars()
{
    for (int i = 0; i < 30; i++)
        printf("*");
}

int main(void)
{
    print_stars();
    printf("\nHello World!\n");
    print_stars();
    printf("\n");
    return 0;
}
```



매개 변수와 반환값

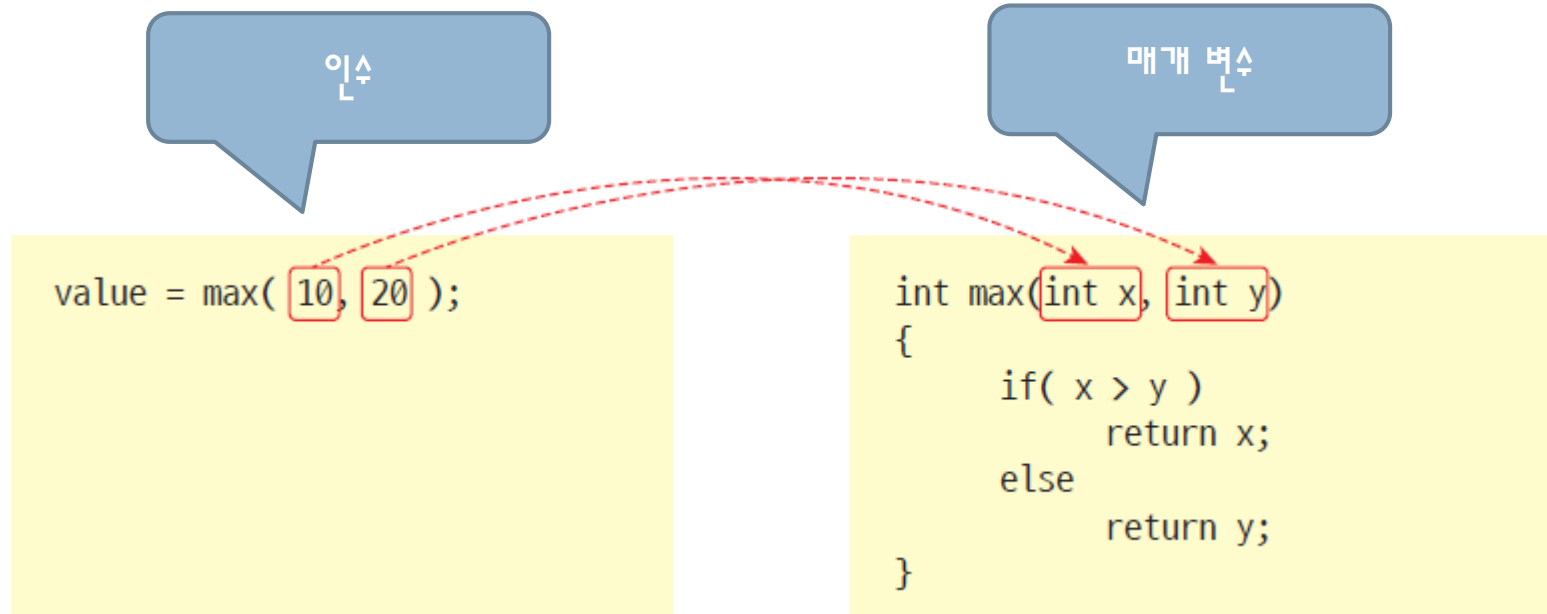
Syntax: 함수의 구조





인수와 매개변수

- 인수(argument)는 호출 프로그램에 의하여 함수에 실제로 전달되는 값이다.
- 매개 변수(parameter)는 이 값을 전달받는 변수이다.





인수와 매개 변수

- 만약 매개 변수가 없는 경우에는 `print_stars(void)`와 같이 매개변수 위치에 `void`를 써주거나 `print_stars()`와 같이 아무 것도 적지 않으면 된다.
- 함수가 호출될 때마다 인수는 달라질 수 있다.
- 매개 변수의 개수는 정확히 일치하여야 한다는 점이다. 매개 변수의 개수와 인수의 개수가 일치하지 않으면 아주 찾기 어려운 오류가 발생하게 된다.



반환값

- 반환값(return value)은 함수가 호출한 곳으로 반환하는 작업의 결과값이다.
- 값을 반환하려면 **return** 문장 다음에 수식을 써주면 수식의 값이 반환된다.
- 인수는 여러 개가 있을 수 있으나 반환값은 하나만 가능하다.

```
value = max( 10, 20 );
```

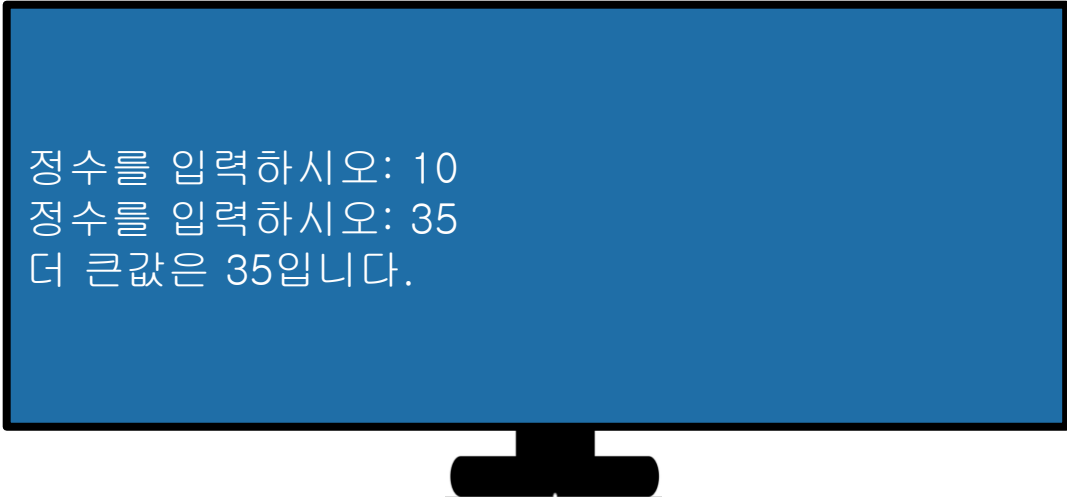
```
int max(int x, int y)
{
    if( x > y )
        return x;
    else
        return y;
}
```





예제

- 위에서 작성한 `max()` 함수를 호출하여서 사용자가 입력한 값 중에서 더 큰 값을 찾아보자.



정수를 입력하시오: 10
정수를 입력하시오: 35
더 큰값은 35입니다.



```
#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;
```

```
    printf("정수를 입력하시오: ");
```

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

```
    printf("정수를 입력하시오: ");
```

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

```
    int larger;
```

```
    larger = max(x, y);
```

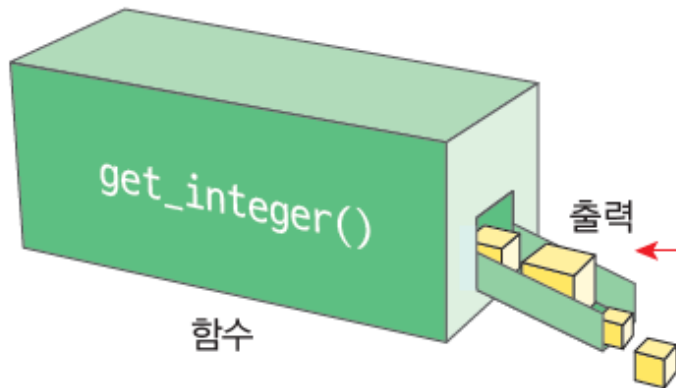
```
    printf("더 큰값은 %d입니다. \n", larger);
```

```
    return 0;
```

```
}
```



Lab: 정수를 입력받는 get_integer() 함수



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



get_integer() 함수

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

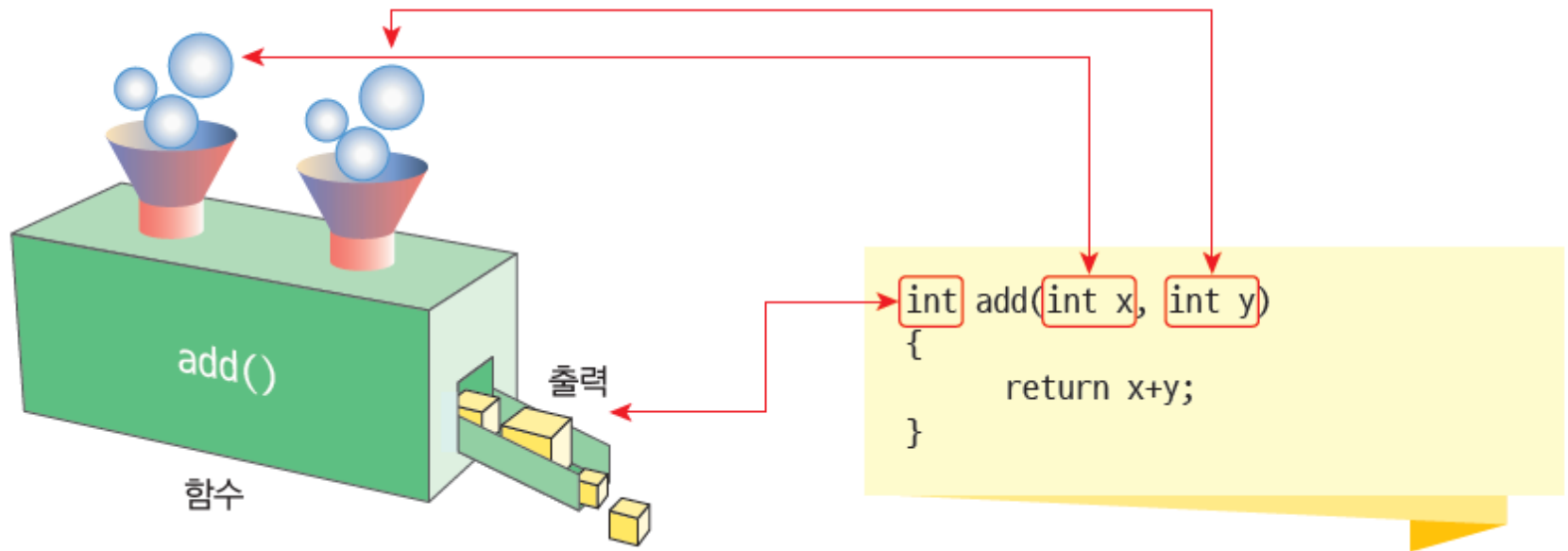
int get_integer(void)
{
    int n;

    printf("정수를 입력하시오: ");
    n=get_integer();

    return n;
}
```



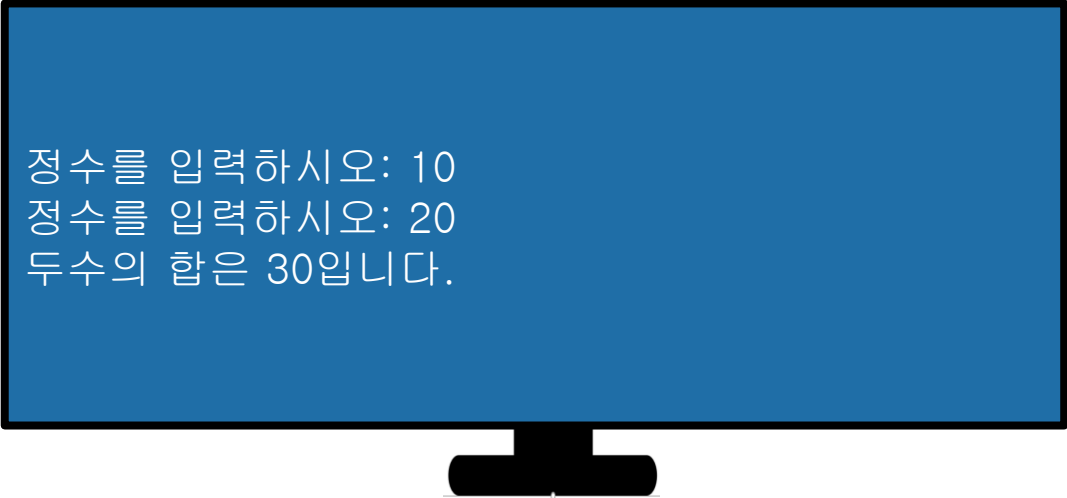
Lab: 정수의 합을 계산하는 add() 함수





Lab: 정수의 합을 계산하는 프로그램

- 앞에서 작성한 `get_integer()`까지 사용하여 사용자로 부터 받은 정수의 합을 계산하여 출력하자.



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




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

    int sum = add(x, y);
    printf("두수의 합은 %d입니다. \n", sum);
    return 0;
}
```



Lab: 팩토리얼 계산 함수

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

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

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



예제

```
#include <stdio.h>

long factorial(int n)
{
    long 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;
}
```



Lab: 온도 변환기

'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

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

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



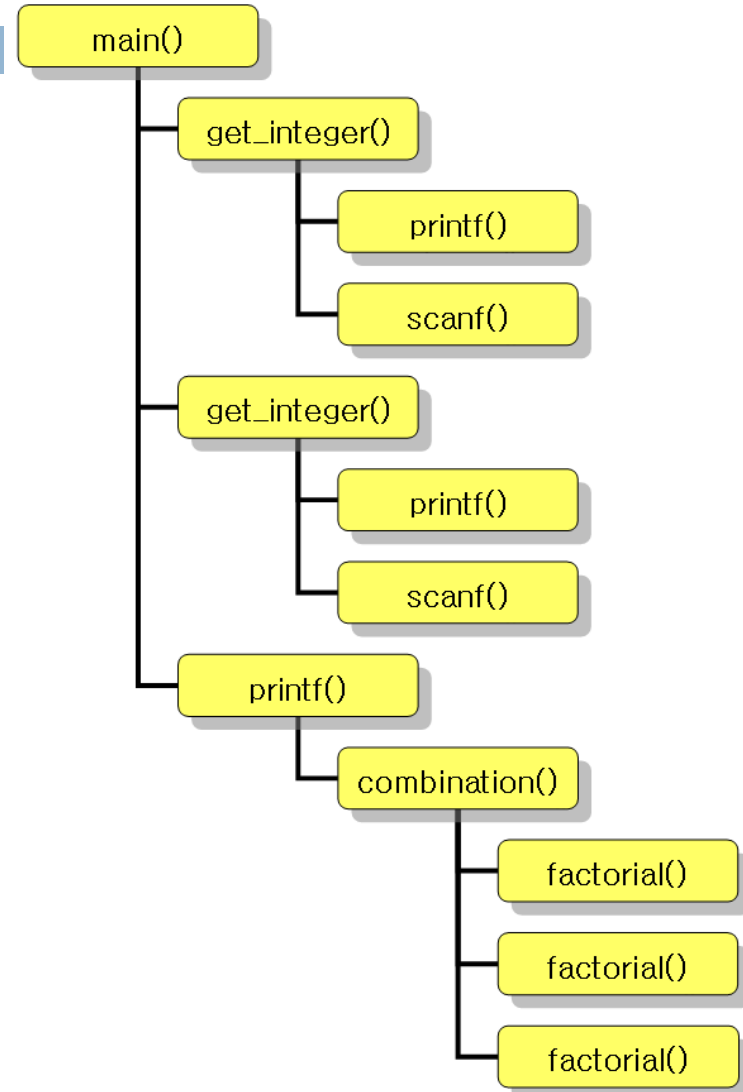
```
int main(void)
{
    char choice;
    double temp;
    while (1) {
        printOptions();
        printf("메뉴에서 선택하세요.");
        choice = getchar();
        if (choice == 'q') break;
        else if (choice == 'c') {
            printf("섭씨 온도: ");
            scanf("%lf", &temp);
            printf("화씨 온도: %lf \n", C2F(temp));
        }
        else if (choice == 'f') {
            printf("화씨 온도: ");
            scanf("%lf", &temp);
            printf("섭씨 온도: %lf \n", F2C(temp));
        }
        getchar();// 엔터키 문자를 삭제하기 위하여 필요!
    }
    return 0;
}
```



조합(combination) 계산 함수

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

- 팩토리얼 계산 함수와 `get_integer()` 함수를 호출하여 조합을 계산한다



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

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

```
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;

    return result;
}
```

```
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;
}
```




Lab: 소수 찾기

- 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



Lab: 소수 찾기

정수를 입력하시오: 12229
12229은 소수가 아닙니다.



알고리즘

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

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

```
int get_integer(void)
```

```
{
    int n;

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

```
int is_prime(int n)
```

```
{
    int i;

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

```
int main(void)
```

```
{
    int n, result;
    n = get_integer();

    if (is_prime(n) == 1)
        printf("%d은 소수입니다.\n", n);
    else
        printf("%d은 소수가 아닙니다.\n", n);
    return 0;
}
```



함수 연형

- 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;
}

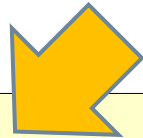
double c_to_f(double c_temp)
{
    return 9.0 / 5.0 * c_temp + 32;
}
```

전체 솔루션		1 오류	2 경고	0 메시지	빌드 + IntelliSense	검색 오류 목록		
코드	설명	프로젝트	파일	줄	Suppress			
C4013	'c_to_f'(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 %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;
}
```



함수 원형

- 함수 원형은 함수의 이름, 매개변수, 반환형을 함수가 정의되기 전에 미리 알려주는 것이다.
- 함수 원형은 함수 헤더에 세미콜론(;)만을 추가한 것과 똑같다. 다만 함수 원형에서는 매개 변수의 이름은 적지 않아도 된다. 매개 변수의 자료형만 적으면 된다.

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

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



함수 원형을 사용하지 않는 예제

```
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 원형을 사용하지 않는 예제

```
#include <stdio.h>
double sub1(double d)
{
    sub2(100.0);
}
double sub2(double d)
{
    sub1(20.0);
}
int main(void)
{
    return 0;
}
```

이런 경우에는 원형 말고는 방법이 없음

오류 목록						
전체 솔루션		1 오류	1 경고	0 메시지	빌드 + IntelliSense	검색 오류 목록
코드	설명	프로젝트	파일	줄	비표시 오류(Suppr...)	
C4013	'sub2'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.	Project14	소스.c	4		
C2371	'sub2': 재정의. 기본 형식이 다릅니다.	Project14	소스.c	6		



라이브러리 함수

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





난수 함수

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





예제: 로또 번호 생성하기

- 1부터 45번 사이의 난수 발생



4 21 22 34 37 38 + 보너스번호 33 [내 번호 당첨조회](#)



실습 코드

```
#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



1 부터 45 사이로 제한

- `printf("%d ", 1+(rand()%45));`



42 18 35 41 45 20

- 하지만 실행할 때마다 항상 똑같은 난수가 발생된다.



실행할 때마다 다르게 하려면

- 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.



아직도 문제는 있다.

- 난수가 겹칠 수 있다.
- 겹치는 난수 검사는 뒤에서 학습하는 배열을 사용하는 것이 최선

12

28

12

1

9

18



lab: 동전 던지기 게임

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

동전의 앞면: 53
동전의 뒷면: 47



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

```
int coin_toss(void);
```

```
int main(void)
{
    int toss;
    int heads = 0;
    int tails = 0;
    srand((unsigned)time(NULL));

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

```
printf("동전의 앞면: %d \n", heads);
printf("동전의 뒷면: %d \n", tails);
return 0;
```

```
}

int coin_toss(void)
{
    int i = rand() % 2;
    if (i == 0)
        return 0;
    else
        return 1;
}
```

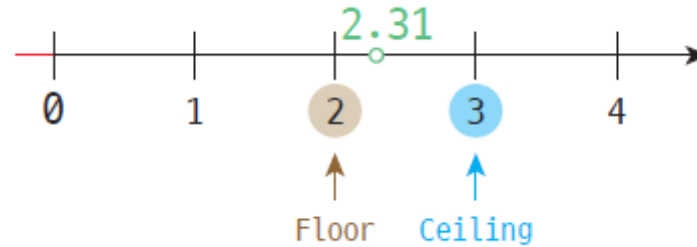


수학 라이브러리 함수

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() / fabs() 함수



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

```
result = floor(value);           // result는 1.0이다.  
printf("%lf ", result);
```

```
result = ceil(value);           // result는 2.0이다.  
printf("%lf ", result);
```

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



함수를 사용하는 이유

- 소스 코드의 중복성을 없애준다.
- 한번 제작된 함수는 다른 프로그램을 제작할 때도 사용이 가능하다.
- 복잡한 문제를 단순한 부분으로 분해할 수 있다.



복잡한 프로그램은 함수로 분리한다.

```
int main(void)
{
    // 숫자들의 리스트를 키보드에서 읽어들이는 코드
    ...
    // 숫자들을 크기순으로 정렬하는 코드
    ...
    // 정렬된 숫자들의 리스트를 화면에 출력하는 코드
    ...
}
```



```
int main(void)
{
    ...
    read_list(); // 숫자들의 리스트를 키보드에서 읽어 들이는 함수
    sort_list(); // 숫자들의 리스트를 크기순으로 정렬하는 함수
    print_list(); // 숫자들의 리스트를 화면에 출력하는 함수
    ...
}
```



Mini Project

- 사용자가 입력한 실수들의 합을 계산하는 프로그램을 작성해보자.

```
실수를 입력하시오: 10  
실수를 입력하시오: 20  
실수의 합=30.000000
```



Mini Project

- C 프로그램 작성 시에 도움을 줄 수 있는 함수들을 몇 가지 작성하고 테스트하자.
- `int get_integer()`: 안내 메시지를 출력하고 정수를 받아서 반환한다.
- `double get_double()`: 안내 메시지를 출력하고 `double`형의 실수를 받아서 반환한다.
- `char get_char()`: 안내 메시지를 출력하고 문자를 받아서 반환한다.


```
#include <stdio.h>

int get_integer();
double get_double();
char get_char();
int main(void)
{
    double f, g;
    f = get_double();
    g = get_double();
    printf("실수의 합=%lf\n", f + g);
    return 0;
}

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

```
double get_double()
{
    double n;
    printf("실수를 입력하시오: ");
    scanf("%lf", &n);
    return n;
}

char get_char()
{
    char n;
    printf("문자를 입력하시오: ");
    scanf(" %c", &n);
    return n;
}
```



문제

두 개의 음이 아닌 정수를 입력받아 큰 수의 제곱에서 작은 수의 제곱을 뺀 결과값을 출력하는 프로그램을 작성하시오.

(두 정수를 전달받아 제곱의 차를 리턴하는 함수를 이용할 것)

입력 예

8 10

출력 예

36