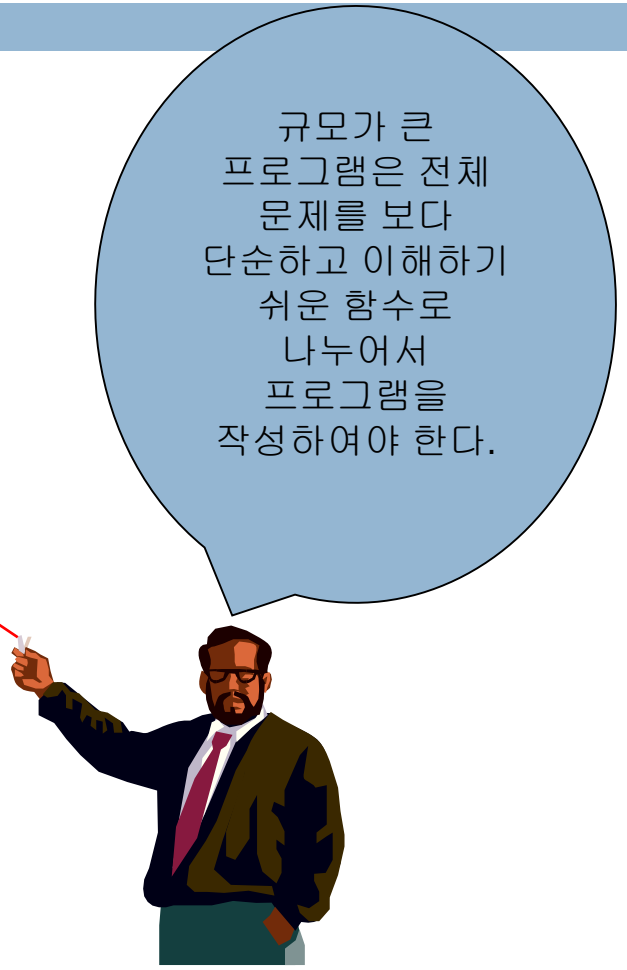
제8장 함수

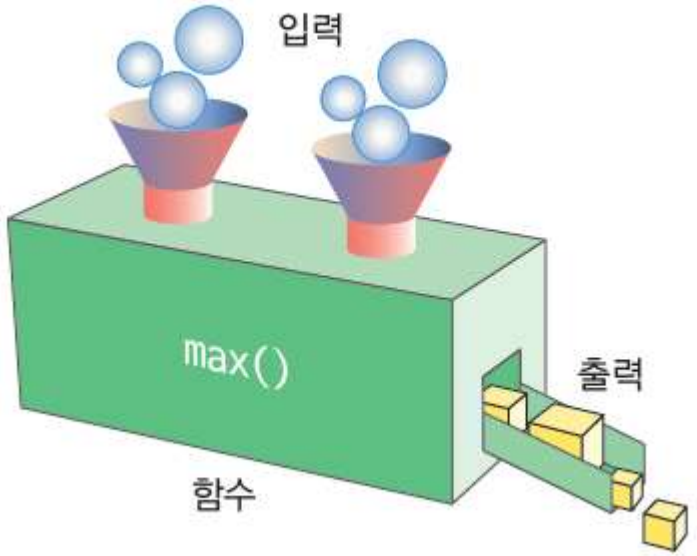
이번 장에서 학습할 내용

- 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


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

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

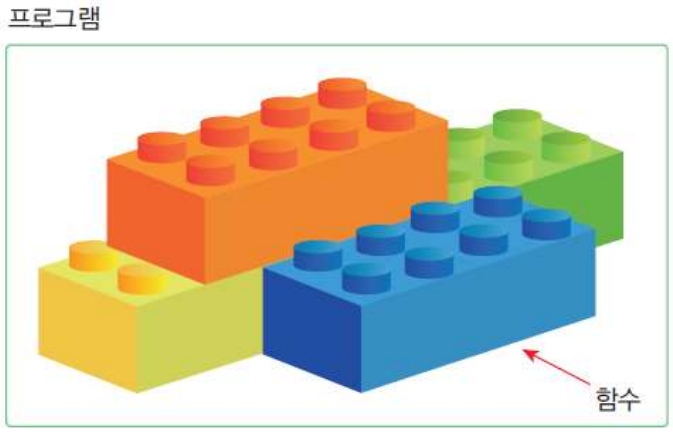
함수가 필요한 이유


```
print_stars();

print_stars();
```

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

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!
printf()
Scanf()
sqrt()
이 함수들은 기본적으로 제공됩니다.

중간 점검

- 함수가 필요한 이유는 무엇인가?
- 함수와 프로그램의 관계는?
- 컴파일러에서 지원되는 함수를 _____함수라고 한다.

함수의 정의

Syntax: 함수 정의

예

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

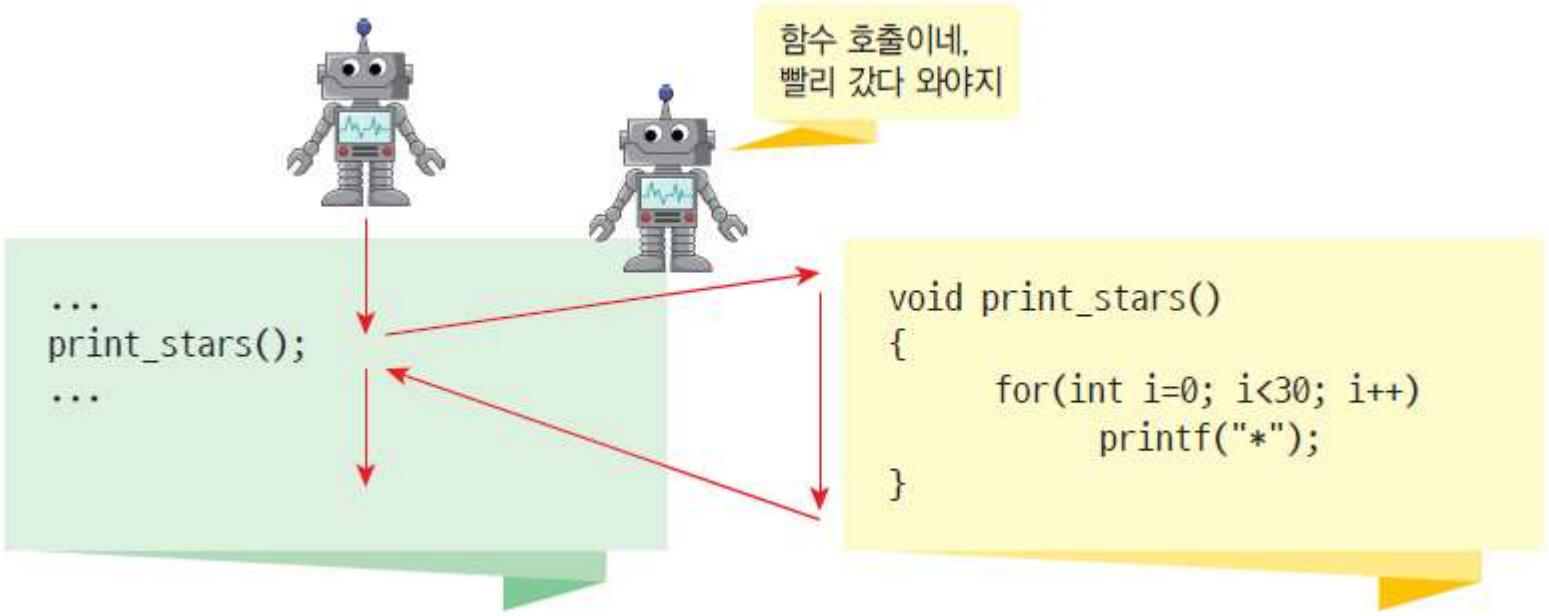
반환형

함수 이름

매개 변수(현재는 없다)

함수 몸체

함수 호출


함수 호출이네.
빨리 갔다 와야지
...
print_stars();
...
void print_stars()
{
for(int i=0; i<30; i++)
printf("*");
}

예제

- print_stars() 함수를 2번 호출하여서 다음과 같이 출력하는 프로그램을 작성해보자.

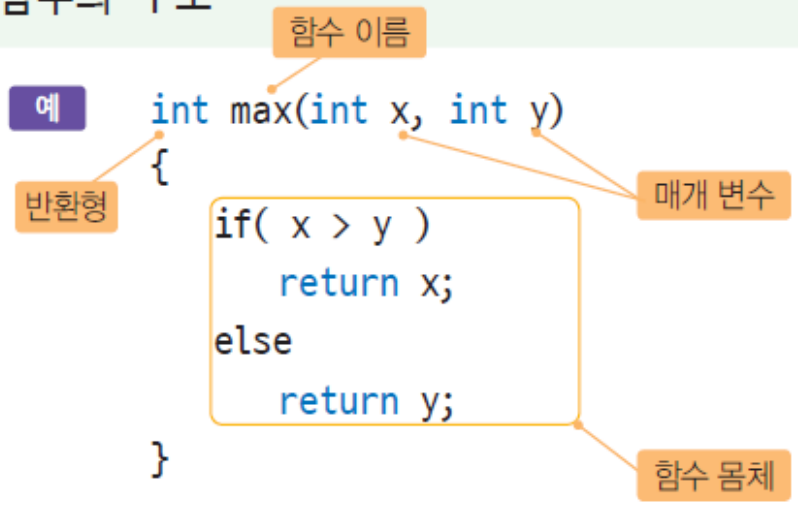
예제

func_exam1.c

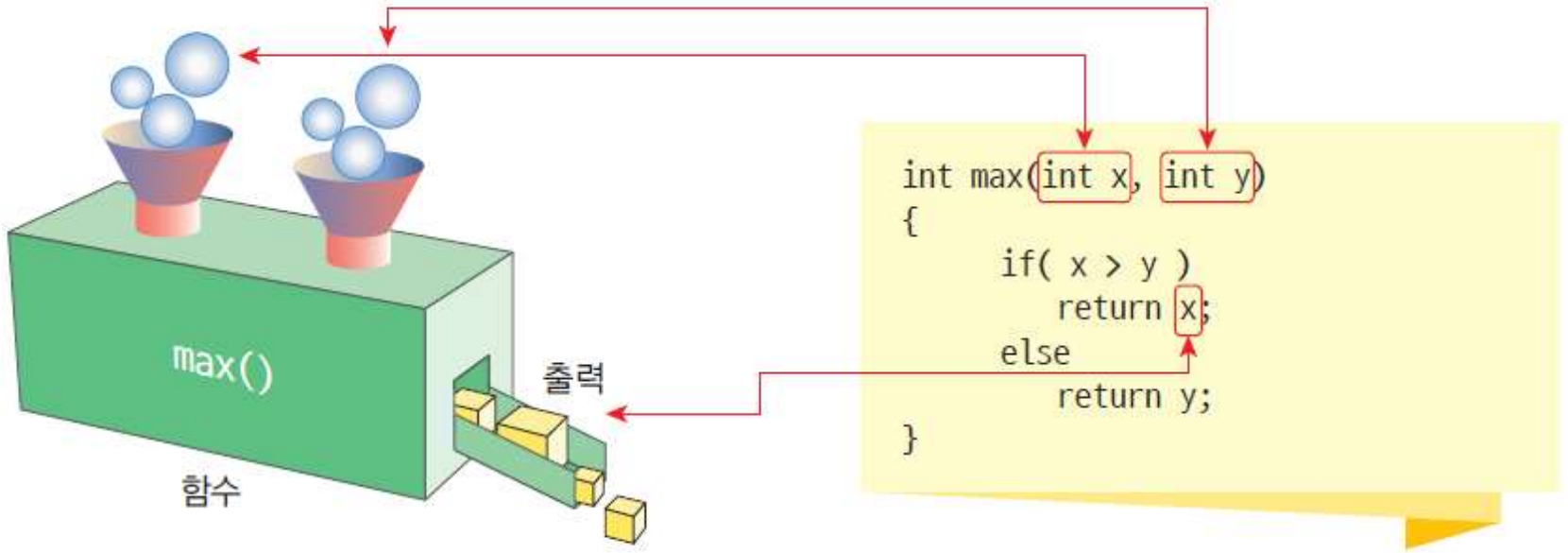

```
#include <stdio.h>

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



매개 변수와 반환값



인수와 매개변수


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

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

반환값


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

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

예제

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

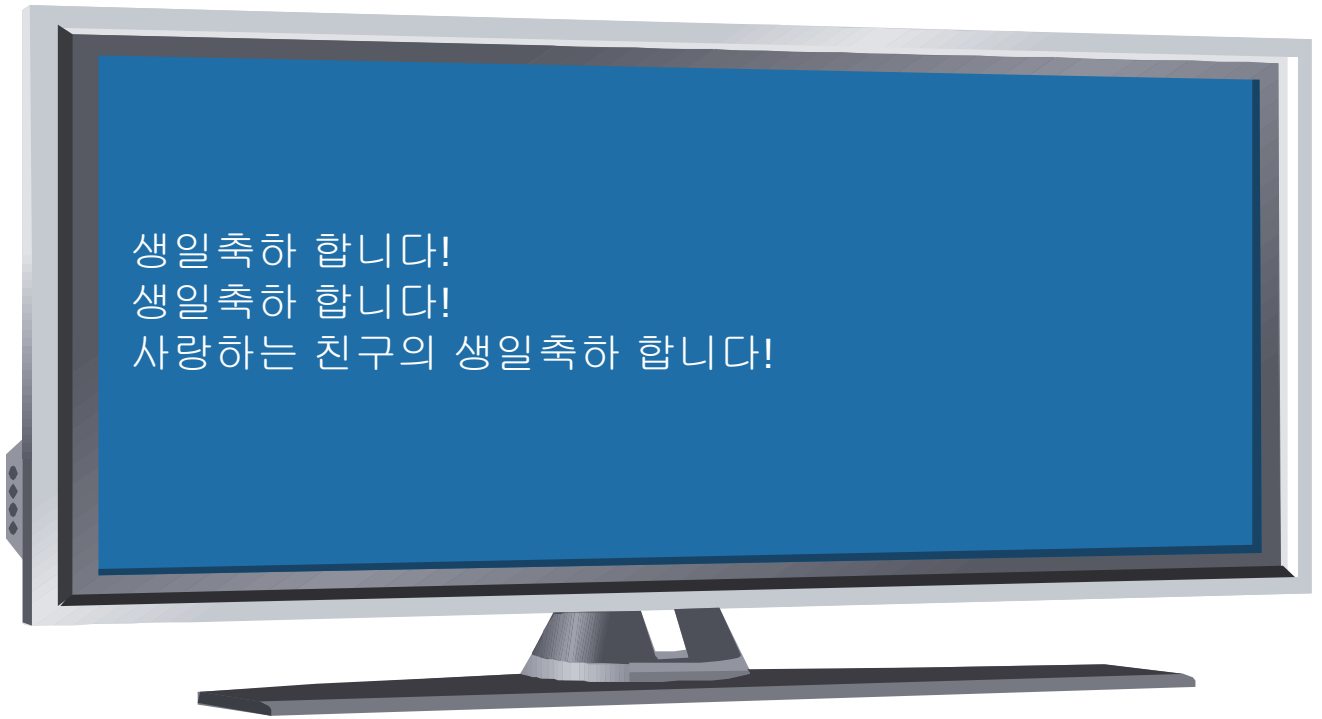
예제

max(p329).c


```c
// 두수 중에서 큰 수를 찾는 함수 예제
#include <stdio.h>

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

int main(void)
{
   int a, b;

   printf("두개의 정수를 입력하시오: ");
   scanf("%d %d", &a, &b);

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

   return 0;
}
```

lab: 생일 축하 함수

- 생일 축하 메시지를 출력하는 함수 happyBirthday()를 작성해보자.

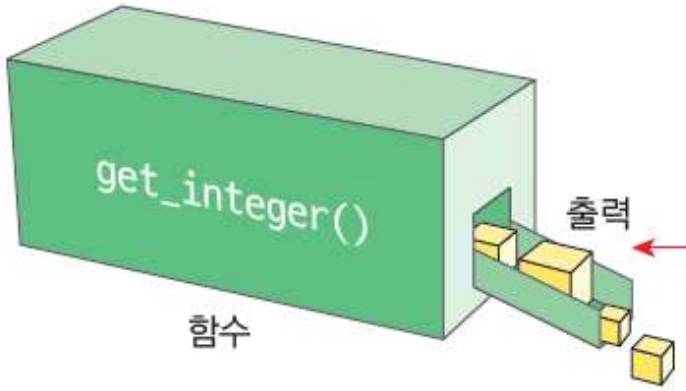


예제

func_exam2.c


```c
#include <stdio.h>

void happyBirthday()
{
   printf("생일축하 합니다! \n");
   printf("생일축하 합니다! \n");
   printf("사랑하는 친구의 ");
   printf("생일축하 합니다! \n");
}

int main(void)
{
   happyBirthday();
   return 0;
}
```

lab: 정수를 입력받는 get_integer() 함수



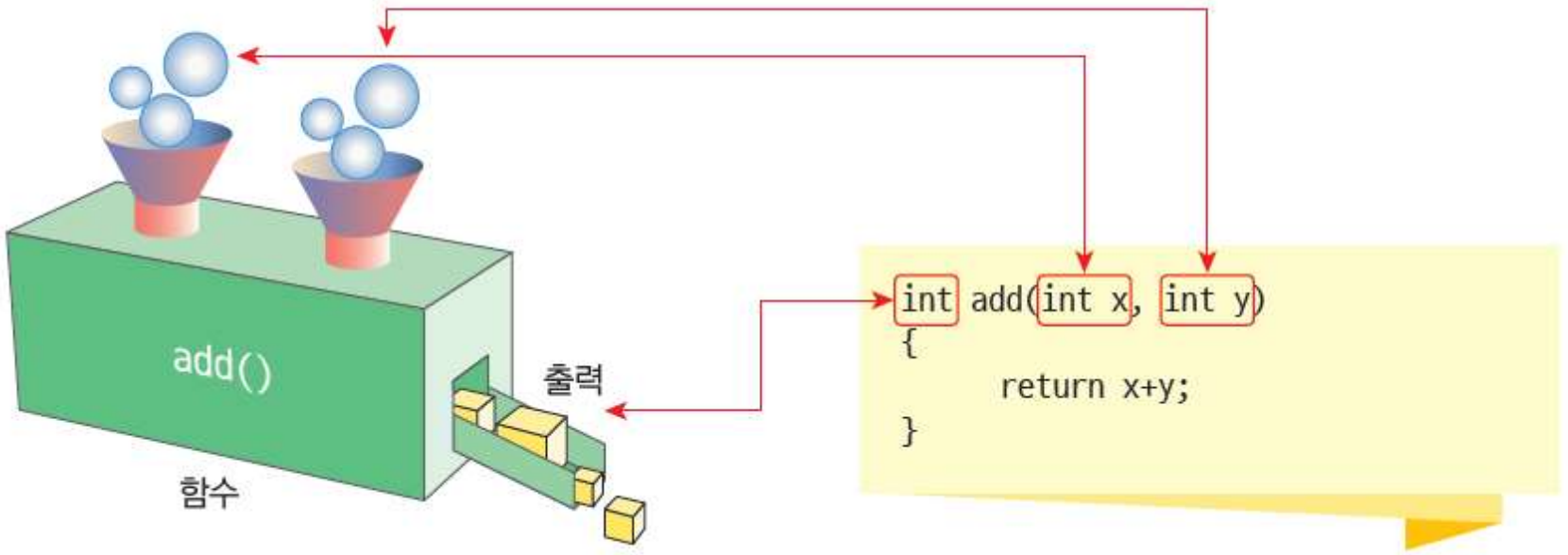

```
int get_integer()
{
      int value;
      printf("정수를 입력하시오: ");
      scanf("%d", &value);
      return value;
}
```

예제

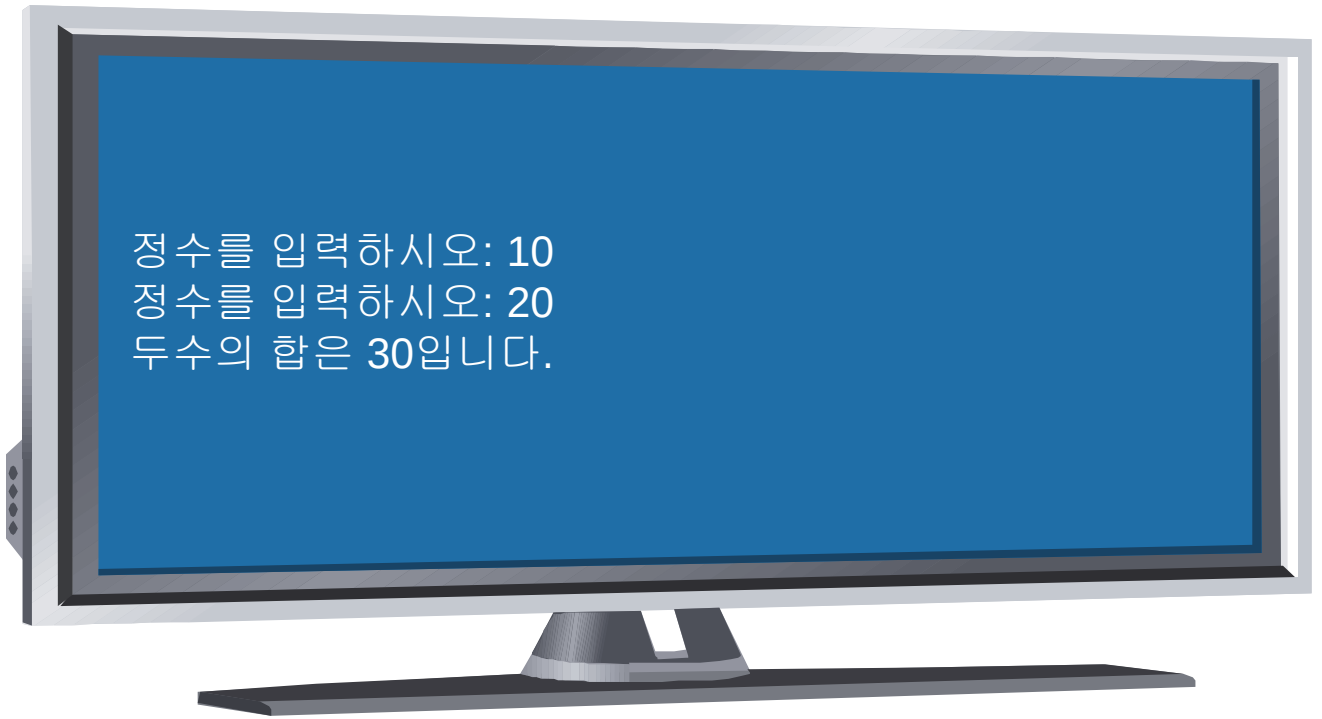
get_integer.c


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

int get_integer(void)
{
   int n;

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

   return n;
}
```

lab: 정수의 합을 계산하는 add() 함수



lab: 정수의 합을 계산하는 add() 함수



예제

add.c

```
#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;
}
```

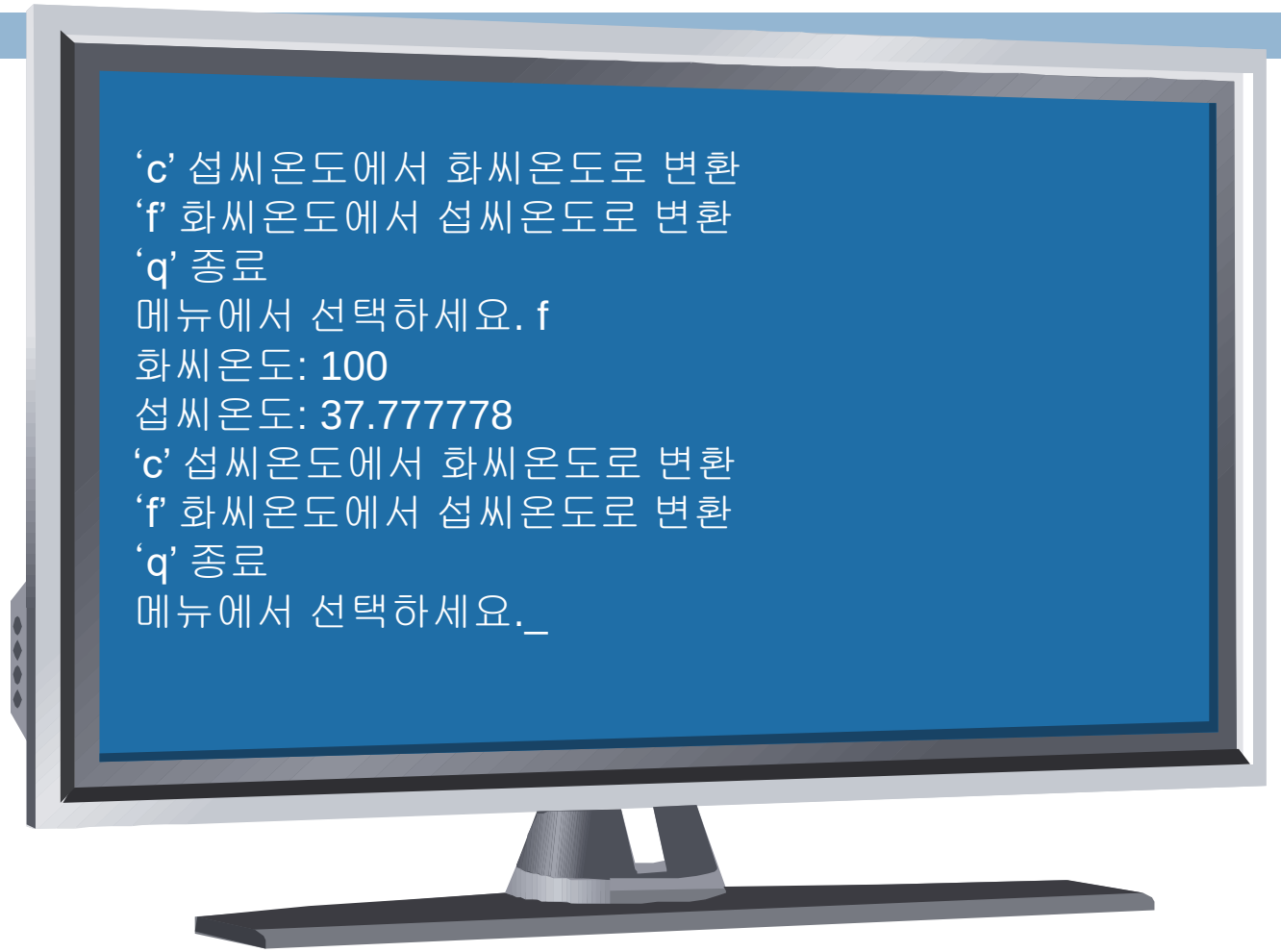
lab: 팩토리얼 계산 함수

예제

factorial.c


```
#include <stdio.h>

int 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: 37.777778
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
메뉴에서 선택하세요._

예제

temperature.c


```
#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;
}
```

예제

temperature.c

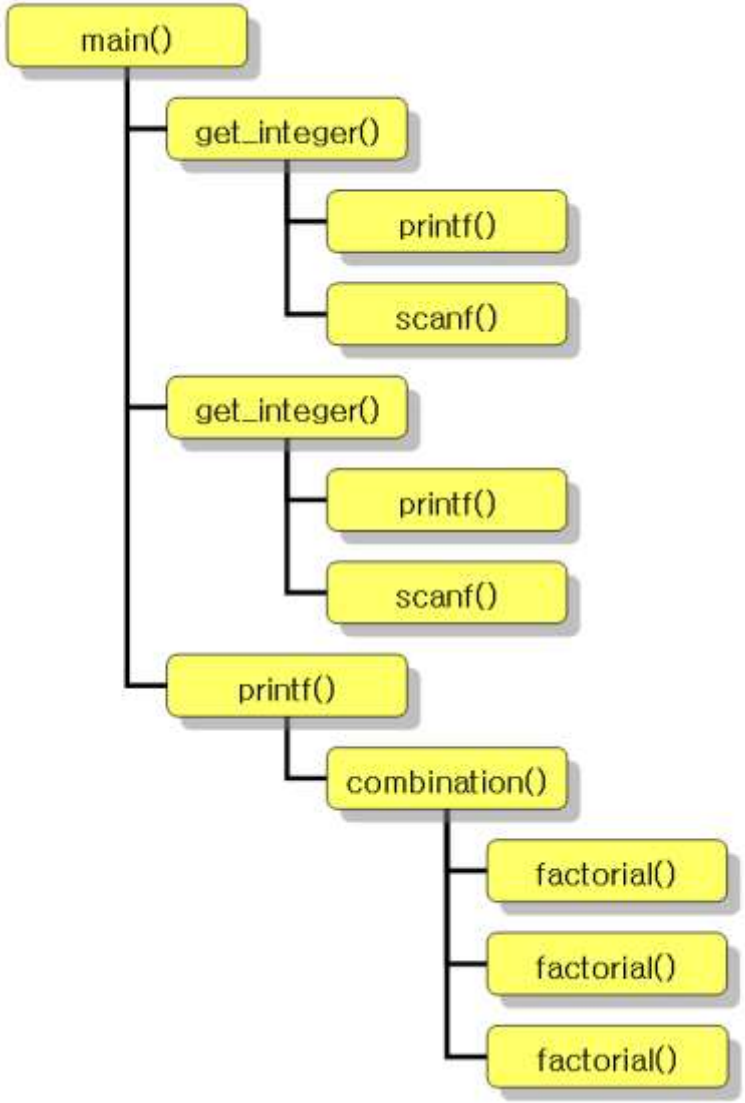
```
int main(void)
{
    char choice;
    double temp;
    while (1) {
        printOptions();
        printf("메뉴에서 선택하세요.");
        choice = getchar();
        if (choice == 'q') break;
        else if (choice == 'c') {
                printf("섭씨온도: ");
                scanf("%lf", &temp);
       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

조합(combination) 계산 함수

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

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

- 팩토리얼 계산 함수와 get_integer() 함수를 호출하여 조합을 계산한다



combination.c

```c
#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)));
}
```


```c
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;
}
```

중간 점검

- 인수와 매개 변수는 어떤 관계가 있는가?
- 반환값이 실수로 정의된 함수에서 실수로 정수를 반환하면 어떤 일이 발생하는가?

lab: 소수 찾기

- 주어진 숫자가 소수(prime)인지를 결정하는 프로그램이다.
- 양의 정수 n이 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

알고리즘

사용자로부터 정수를 입력받아서 변수 n에 저장한다.
약수의 개수를 0으로 초기화한다.
for(i=1; i<=n ; i++)
n을 i로 나누어서 나머지가 0인지 본다.
나머지가 0이면 약수의 개수를 증가한다.
약수의 개수가 2이면 정수 n은 소수이다.

prime(p337).c

```c
#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;
}
```


```c
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);
}
```

효율적인 코딩


```
int is_prime(int n)
{
    int i;

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

}
```

도전문제

- 1부터 사용자가 입력한 숫자 n 사이의 모든 소수를 찾도록 위의 프로그램을 변경하여 보자.

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 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

중간 점검

- 함수 정의의 첫 번째 줄에는 어떤 정보들이 포함되는가? 이것을 무엇이라고 부르는가?
- 함수가 반환할 수 있는 값의 개수는?
- 함수가 값을 반환하지 않는다면 반환형은 어떻게 정의되어야 하는가?
- 함수 정의와 함수 원형의 차이점은 무엇인가?
- 함수 원형에 반드시 필요한 것은 아니지만 대개 매개 변수들의 이름을 추가하는 이유는 무엇인가?
- 다음과 같은 함수 원형을 보고 우리가 알 수 있는 정보는 어떤 것들인가?

```
double pow(double, double);
```

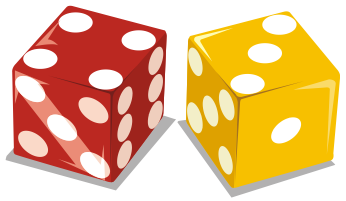
라이브러리 함수

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

난수 함수

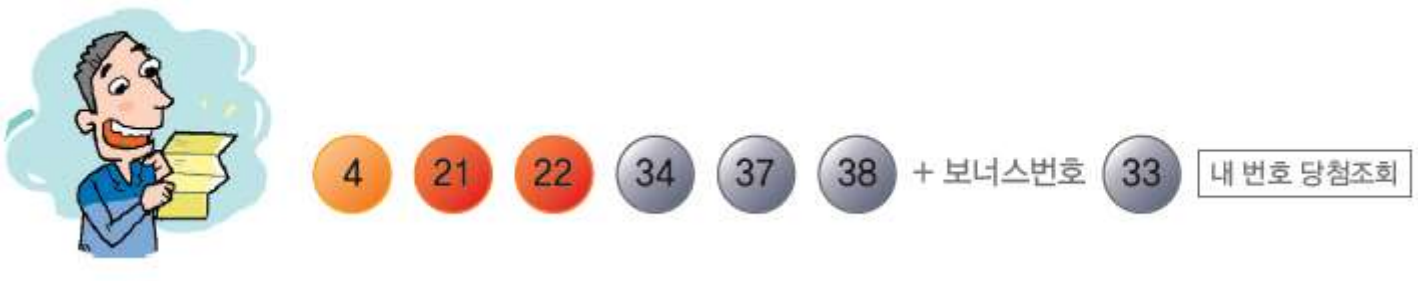
- **난수(random number)**는 규칙성이 없이 임의로 생성되는 수이다.
- 난수는 암호학이나 시뮬레이션, 게임 등에서 필수적이다.

- rand()
  - 난수를 생성하는 함수
  - 0부터 RAND_MAX까지의 난수를 생성



예제: 로또 번호 생성하기

- 1부터 45번 사이의 난수 발생

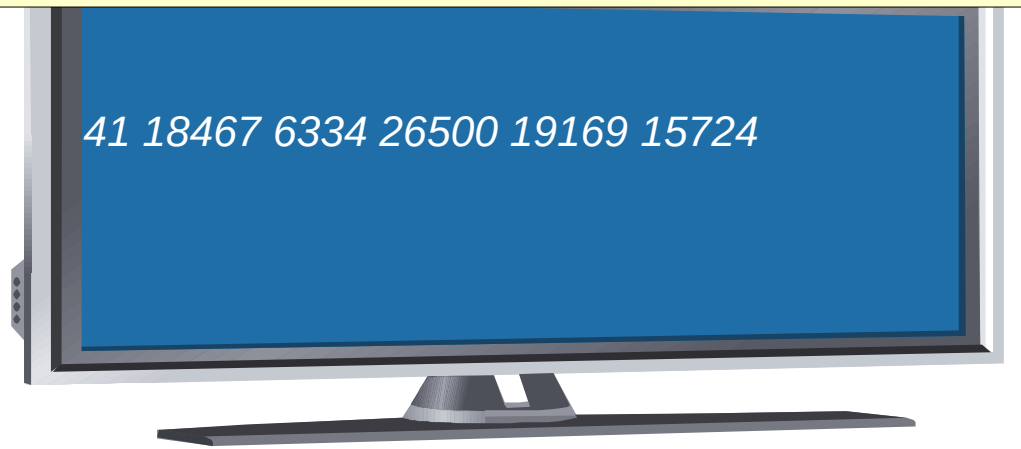


실습 코드

lotto1.c

```c
#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



1부터 45 사이로 제한

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

- 하지만 실행할 때마다 항상 똑같은 난수가 발생된다.

실행할 때마다 다르게 하려면

lotto2.c

- 매번 난수를 다르게 생성하려면 **시드(seed)를 다르게 하여야 한다.**
  - srand((unsigned)time(NULL));

```c
#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.

lab: 동전 던지기 게임

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

coin_toss(p345).c

```
#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;
}
```

lab: 자동차 게임

- 난수를 이용하여서 자동차 게임을 작성

알고리즘

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.*

소스

racing_game.c

```
#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 사용하여서 0에서 99로 제한하였다.

소스


```
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");
}
```

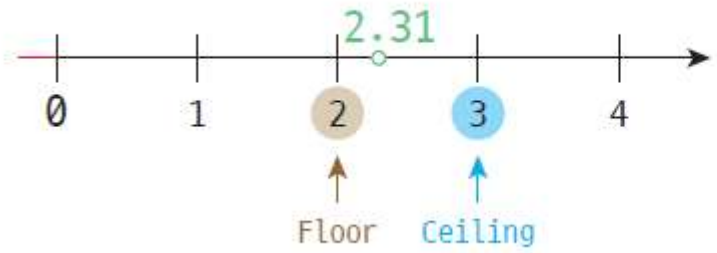
도전문제

- 자동차를 3개로 늘려보자.

수학 라이브러리 함수

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

floor()와 ceil() 함수

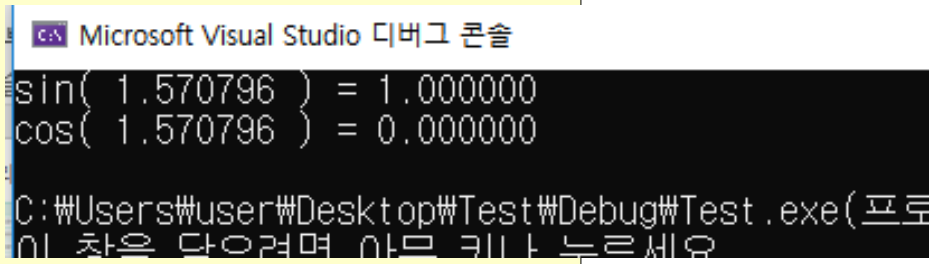


예제

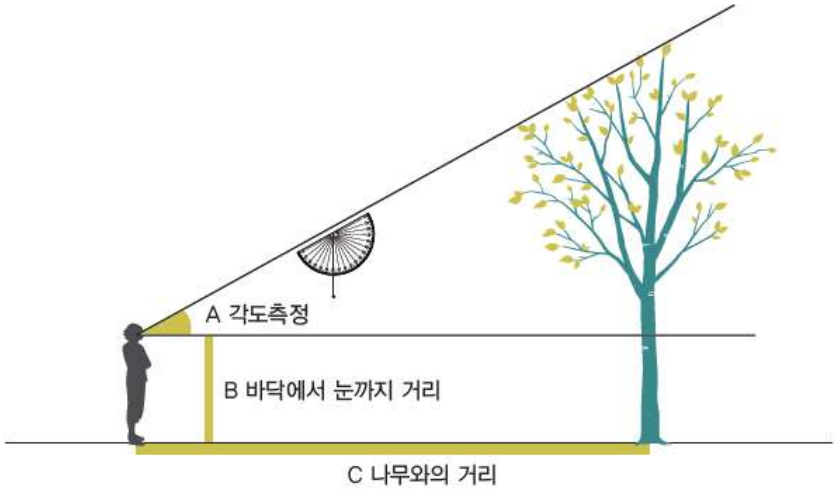
sin(p351).c

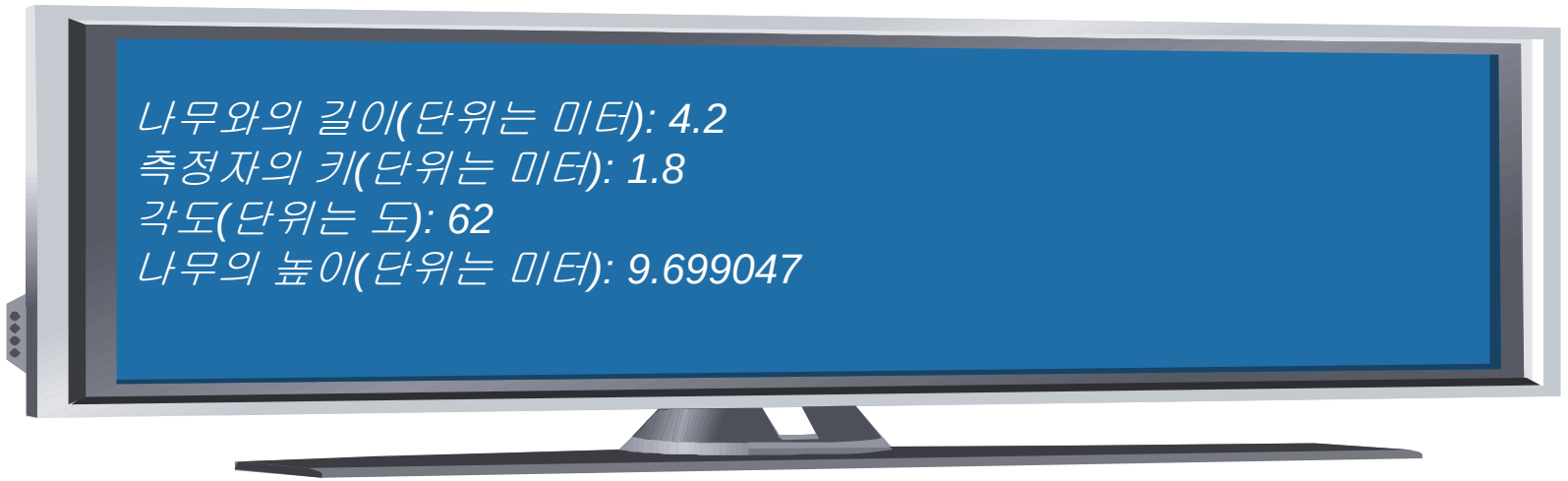

```c
// 삼각 함수 라이브러리
#include <math.h>
#include <stdio.h>

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



lab: 나무 높이 측정


A 각도측정
B 바닥에서 눈까지 거리
C 나무와의 거리


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

tree_height.c

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

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

        printf("나무와의 길이(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;
}
```

함수를 사용하는 이유

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

Q & A