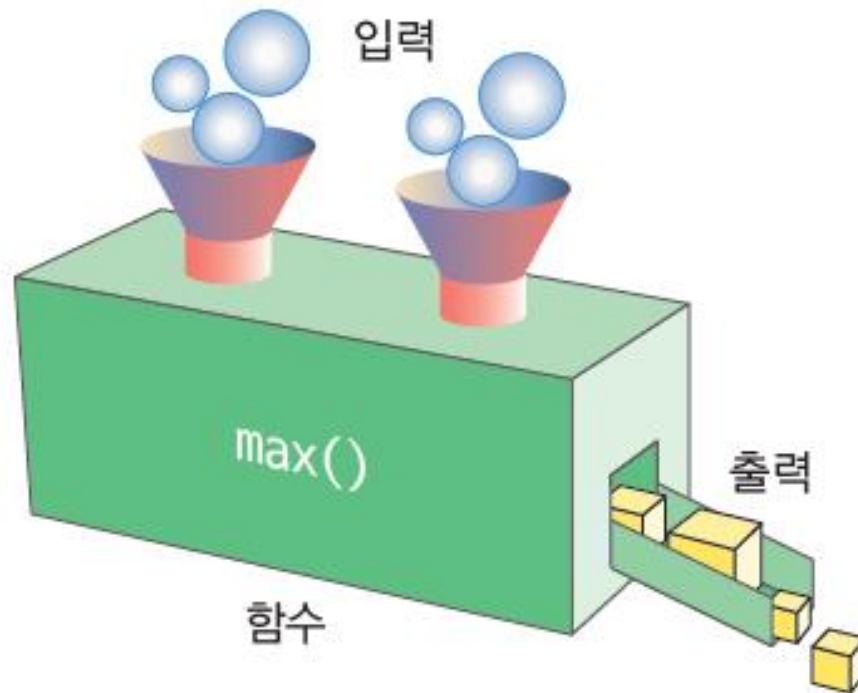


06

함수

6.1. 함수 - 함수의 개념

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



6.1. 함수 - 함수가 필요한 이유

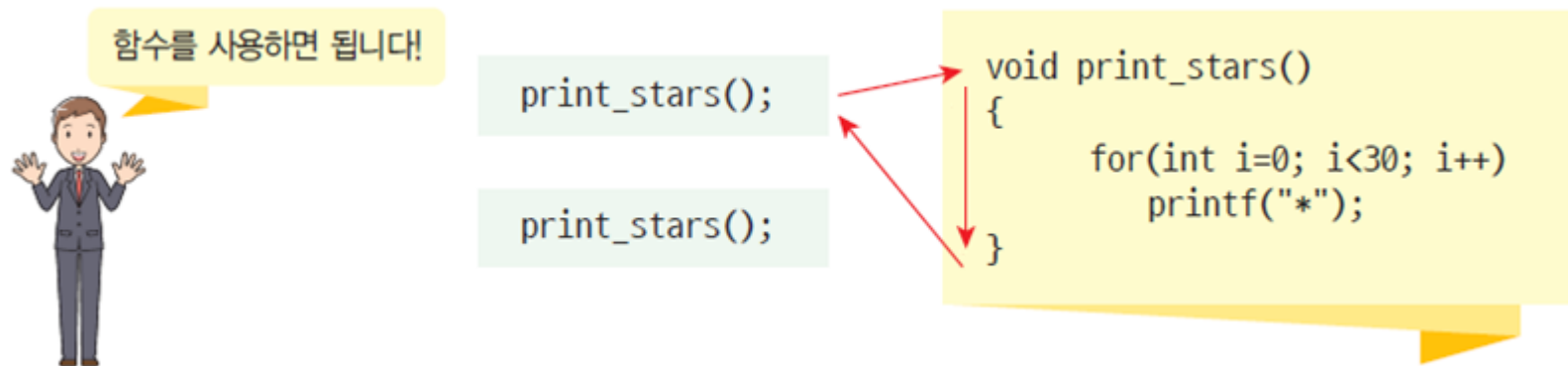
비슷한 코드인데 하나로
합칠 수 있을까?



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

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

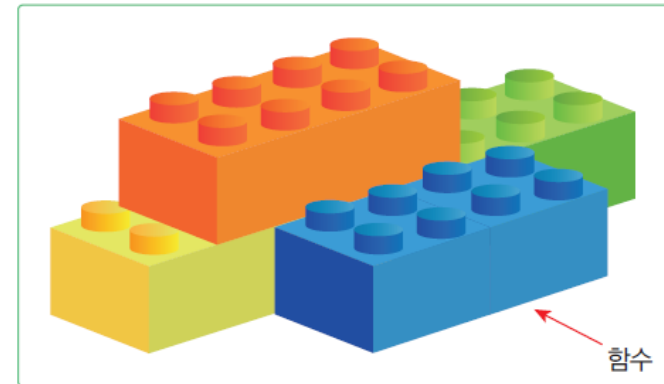
6.1. 함수 - 함수가 필요한 이유



6.1. 함수 - 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.

프로그램



6.1. 함수 - 함수의 종류



함수의 정의

Syntax: 함수 정의

반환형

함수 이름

예

```
void print_stars()
```

매개 변수(현재는 없다)

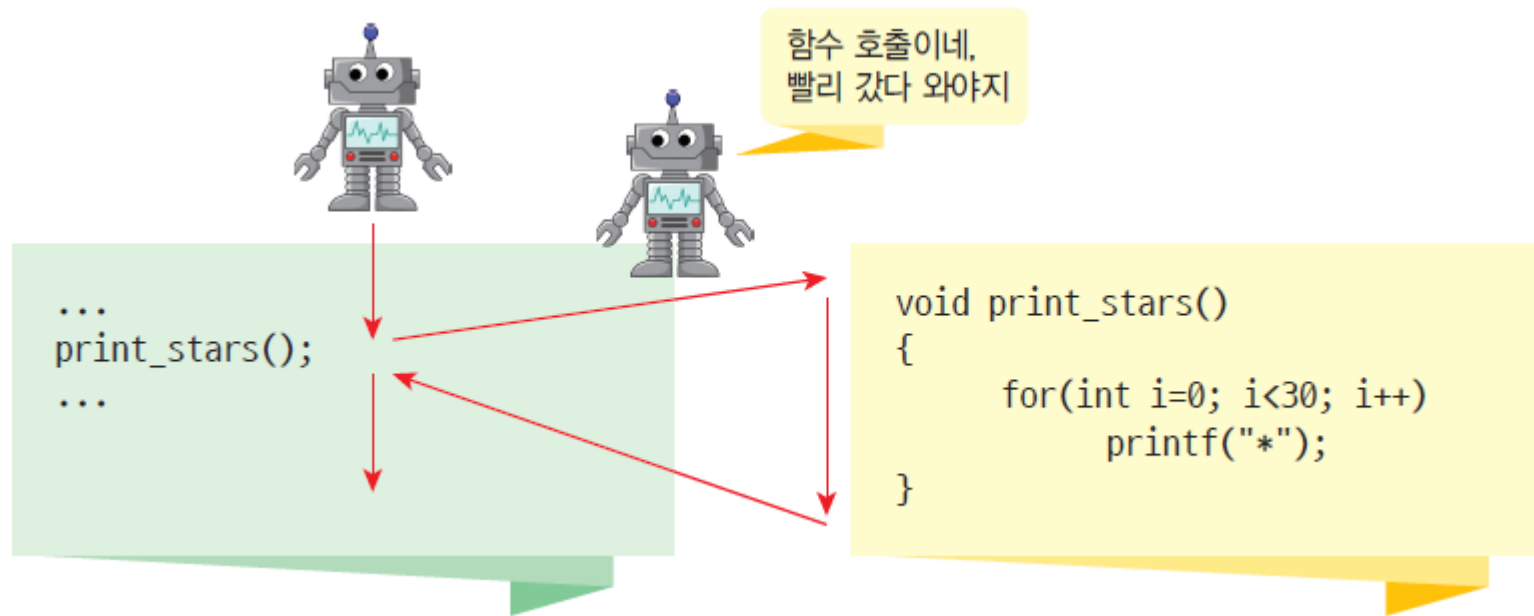
```
{
```

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

함수 몸체

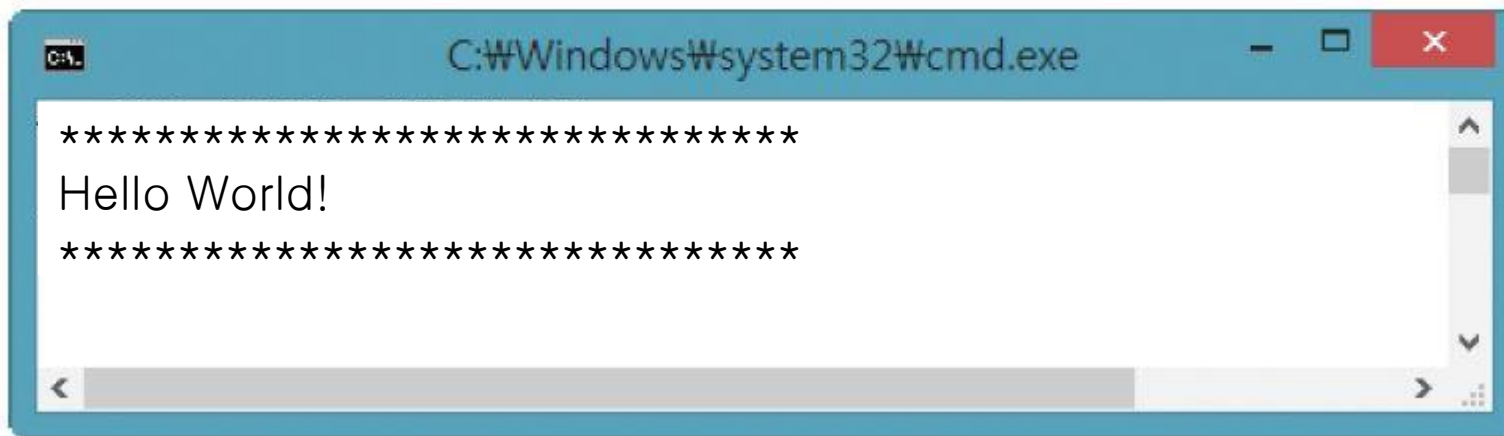
```
}
```

함수 호출



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

실행 결과



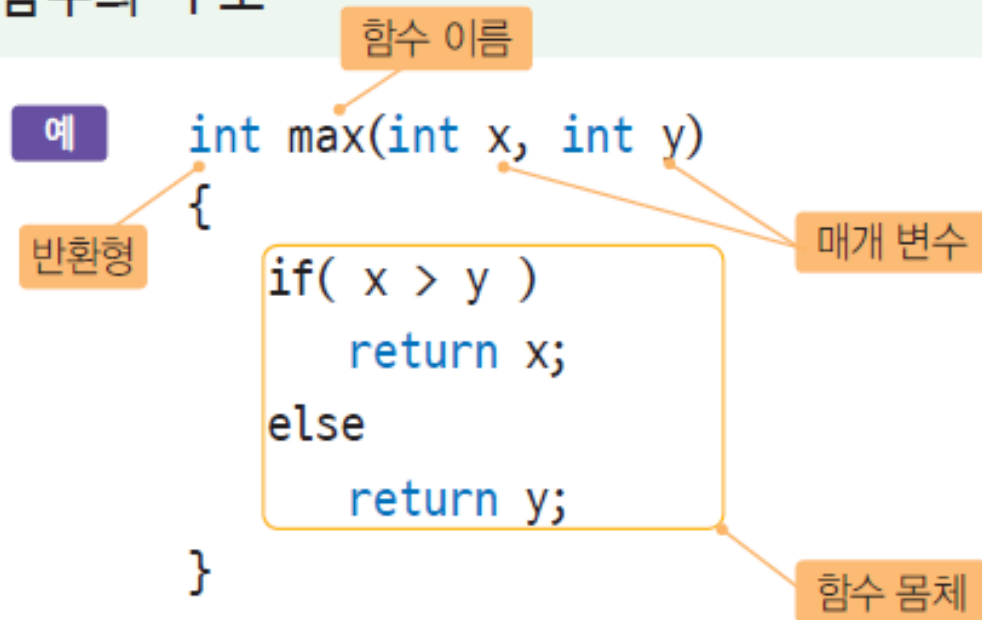
```
C:\Windows\system32\cmd.exe

*****
Hello World!
*****
```

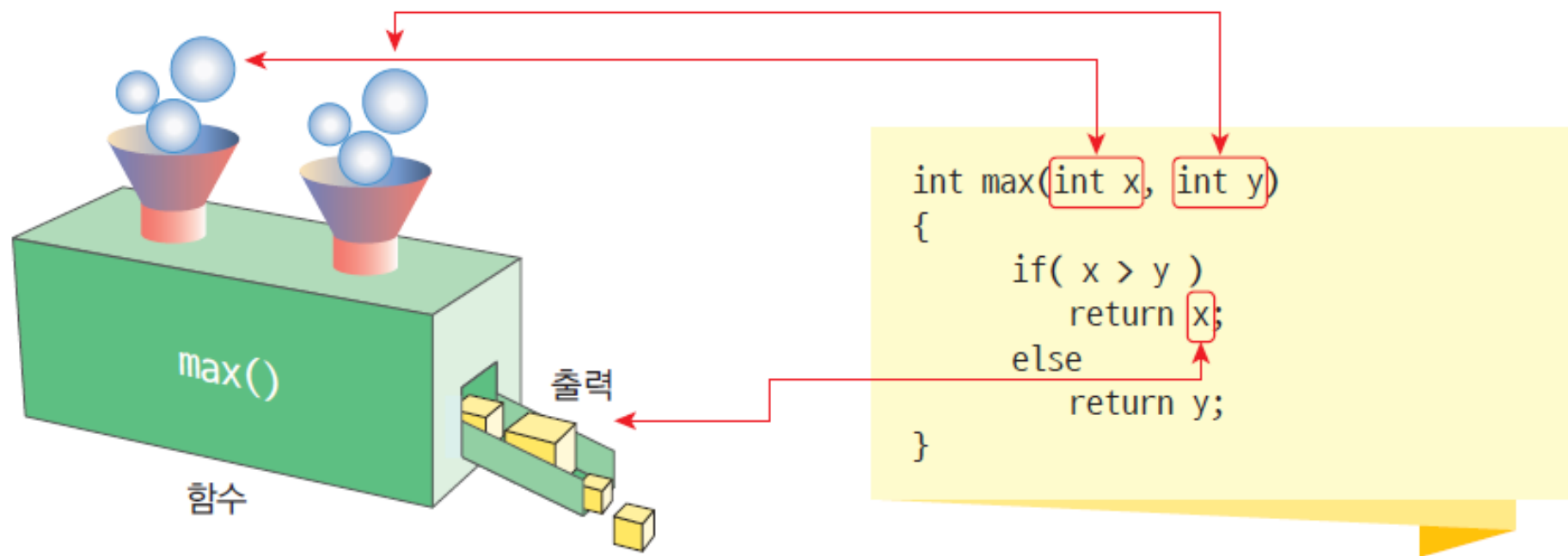
```
01 #include <stdio.h>
02
03 void print_stars()
04 {
05     int i;
06     for(int i=0; i<30; i++)
07         printf("*");
08     printf("\n");
09 }
10
11 int main(void)
12 {
13     print_stars();
14     printf("Hello World\n");
15     print_stars();
16     return 0;
17 }
```

매개 변수와 반환값

Syntax: 함수의 구조



매개 변수와 반환값




인수와 매개변수 / 반환값

매개변수

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

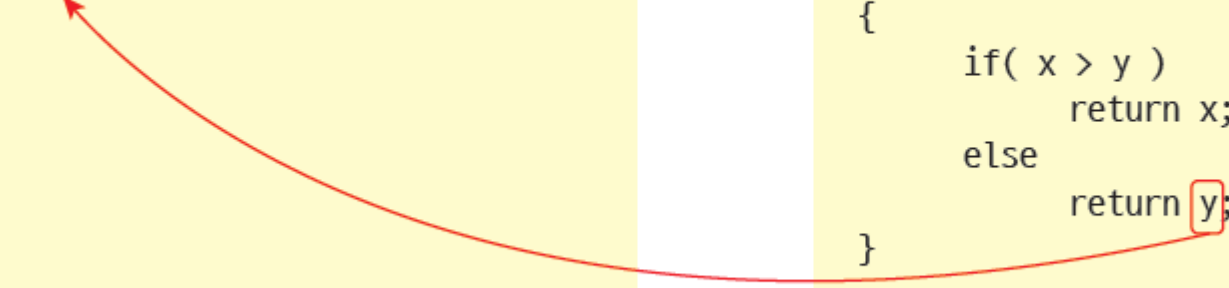
```
int max(int x, int y)
{
    if( x > y )
        return x;
    else
        return y;
}
```



반환값(return값)

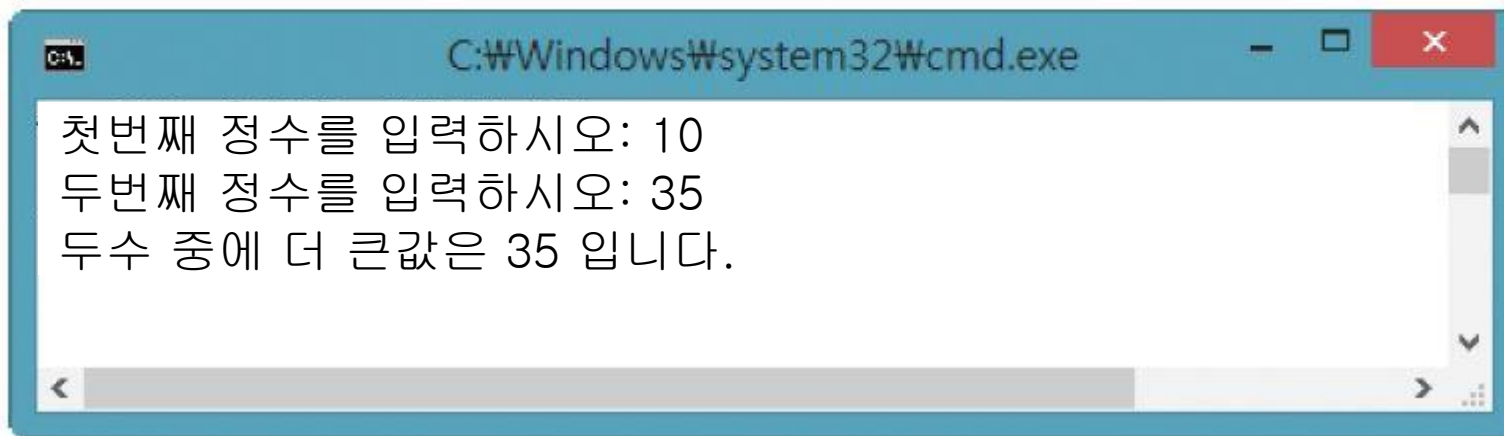
```
value = max( 10, 20 );
```

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



예제 설명 max() 함수를 호출하여서 사용자가 입력한 값 중에서 더 큰값을 찾아보자

실행 결과



```
C:\Windows\system32\cmd.exe

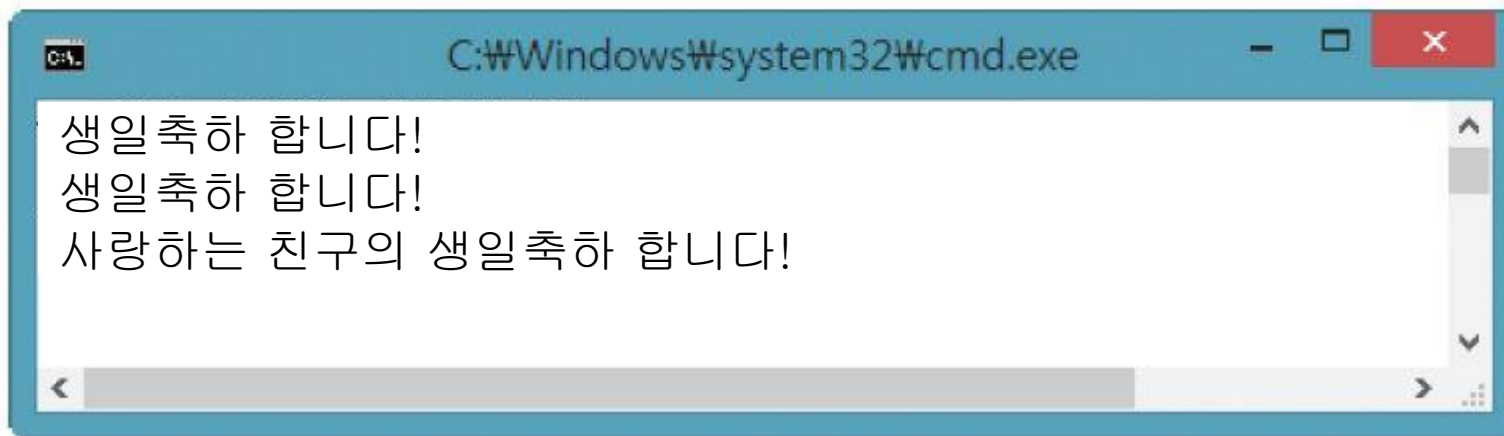
첫번째 정수를 입력하시오: 10
두번째 정수를 입력하시오: 35
두수 중에 더 큰값은 35 입니다.
```

```
01 #include <stdio.h>
02
03 int max(int x, int y)
04 {
05     if( x > y ) return(x);
06     else return(y);
07 }
08
09 int main(void)
10 {
11     int a, b;
12     printf("첫번째 정수를 입력하시오: ");
13     scanf_s("%d", &a);
14     printf("두번째 정수를 입력하시오: ");
15     scanf_s("%d", &b);
16
17     printf("두수 중에 더 큰값은 %d 입니다.\n", max(a, b));
18     return 0;
19 }
```

```
01 #include <stdio.h>
02
03 int max(int x, int y)
04 {
05     if( x > y ) return(x);
06     else return(y);
07 }
08
09 int main(void)
10 {
11     int a, b, max_value;
12     printf("첫번째 정수를 입력하시오: ");
13     scanf_s("%d", &a);
14     printf("두번째 정수를 입력하시오: ");
15     scanf_s("%d", &b);
16
17     max_value = max(a, b);
18     printf("두수 중에 더 큰값은 %d 입니다.\n", max_value);
19     return 0;
20 }
```


예제 설명 생일 축하 메시지를 출력하는 함수 happyBirthday()를 작성해보자.

실행 결과



```
C:\Windows\system32\cmd.exe

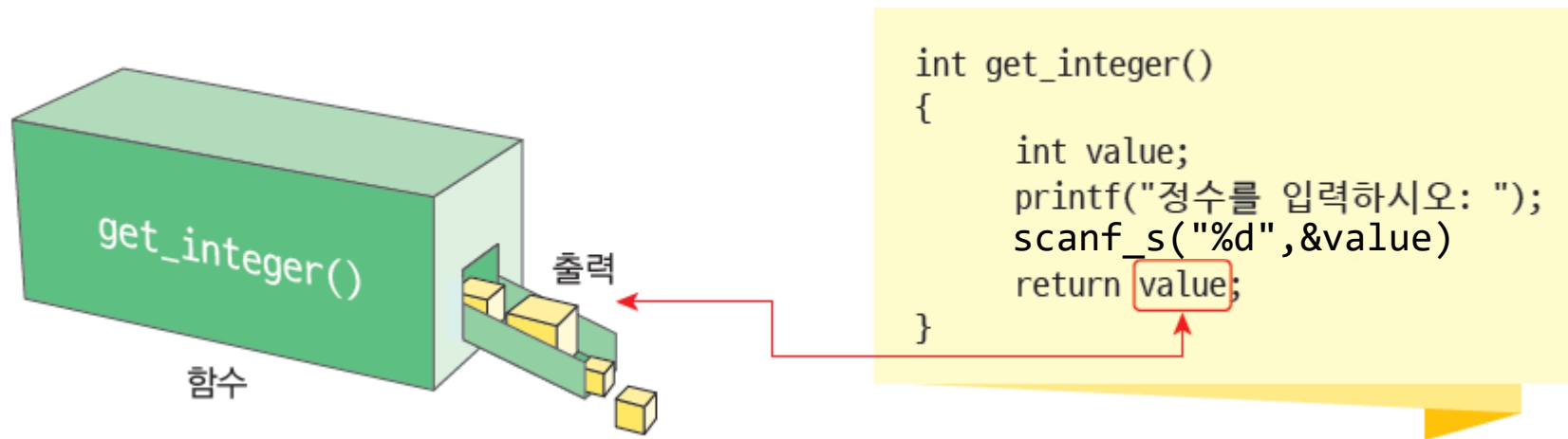
생일 축하 합니다!
생일 축하 합니다!
사랑하는 친구의 생일 축하 합니다!
```

[예제] happyBirthday() 함수 호출

18 |

```
01 #include <stdio.h>
02
03 void happyBirthday()
04 {
05     printf("생일 축하 합니다!\n");
06     printf("생일 축하 합니다!\n");
07     printf("사랑하는 친구의 생일 축하 합니다!\n");
08 }
09
10 int main(void)
11 {
12     happyBirthday();
13     return 0;
14 }
15
```

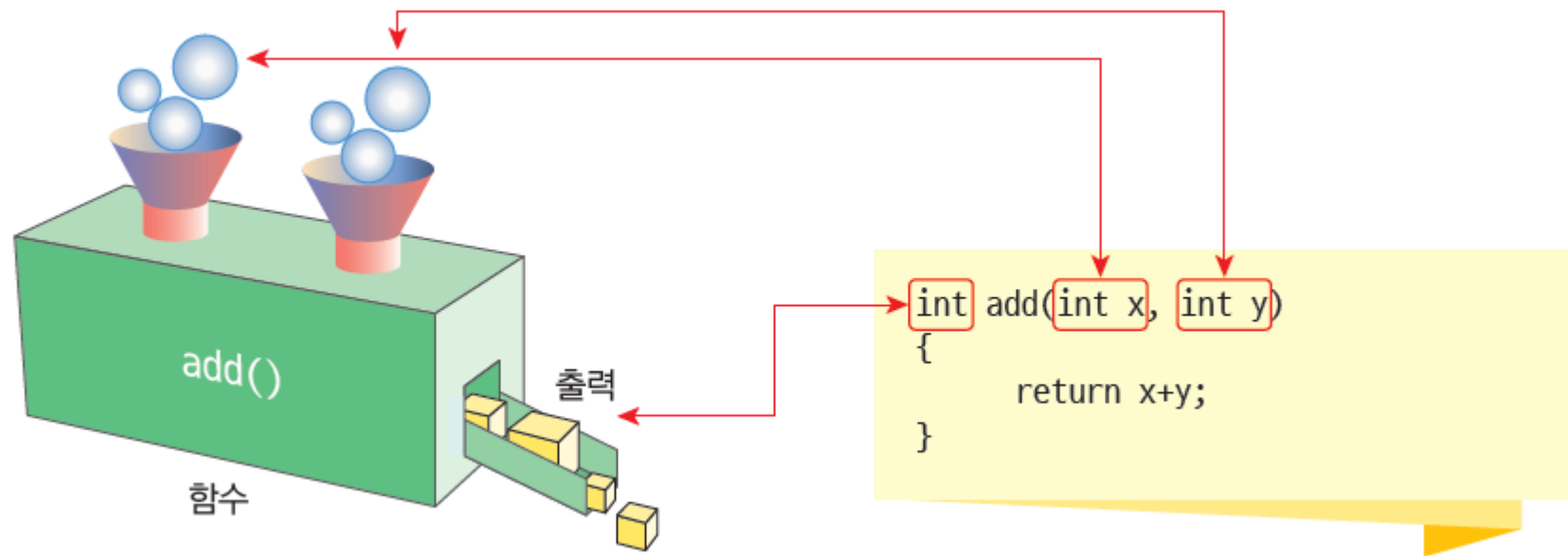
정수를 입력받는 get_integer() 함수



[예제] get_integer() 함수 호출

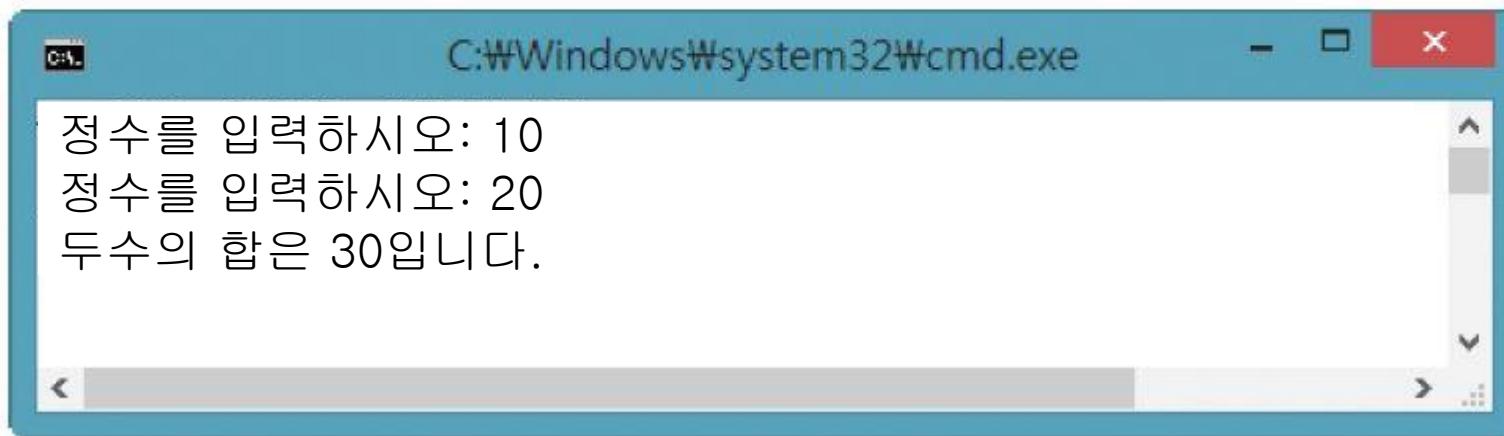
```
01 #include <stdio.h>
02
03 int get_integer(void)
04 {
05     int value;
06     printf("정수를 입력하시오: ");
07     scanf_s("%d", &value);
08
09     return value;
10 }
11
12 int main(void)
13 {
14     int number;
15     number = get_integer();
16     printf("입력한 정수는 = %d입니다.\n", number);
17     return 0;
18 }
```

정수의 합을 계산하는 add() 함수



예제 설명 정수를 입력받아 합을 계산하는 add() 함수를 사용하여 합을 계산해 보자

실행 결과



```
C:\Windows\system32\cmd.exe

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

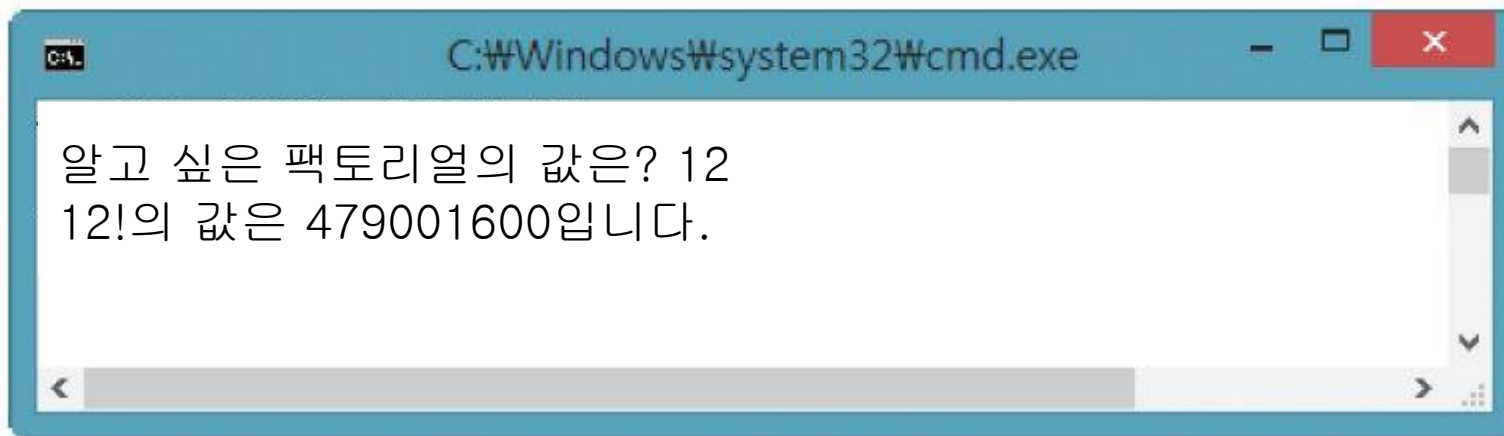
[예제] get_integer() 함수 호출

```
01 #include <stdio.h>
02
03 void get_integer()
04 {
05     int value;
06     printf("정수를 입력하시오: ");
07     scanf_s("%d", &value);
08     return value;
09 }
10
11 int add(int x, int y)
12 {
13     return x + y;
14 }
15
16 int main(void)
17 {
18     int x = get_integer();
19     int y = get_integer();
20     int sum = add(x, y);
21
22     printf("두수의 합은 %d입니다.\n", sum);
23     return 0;
24 }
```

예제 설명

정수를 입력받아 1부터 입력받은 수까지의 연속곱(팩토리얼)을 계산하는 `factorial()` 함수를 사용하여 값을 계산해 보자

실행 결과



```
C:\Windows\system32\cmd.exe

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


[예제] factorial() 함수 호출

```
01 #include <stdio.h>
02
03 long factorial(int n)
04 {
05     long result = 1;
06     int i;
07     for (i = 1; i <= n; i++)
08         result *= i;    // result = result * i
09     return result;
10 }
11
12 int main(void)
13 {
14     int n;
15     printf("알고 싶은 팩토리얼의 값은? ");
16     scanf_s("%d", &n);
17     printf("%d!의 값은 %d입니다.\n", n, factorial(n));
18     return 0;
19 }
```

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

```
return_type 함수명(자료형 parameter1, 자료형 parameter2 ...); //함수 원형 선언
```

```
int main()
```

```
{
```

```
    변수명 = 함수명(argument1, argument2 ...);    // 함수 호출
```

```
}
```

```
return_type 함수명(자료형 parameter1, 자료형 parameter2 ...) //함수 선언부
```

```
{
```

```
    함수 본문;
```

```
    return value;
```

```
}
```

함수를 main()함수보다 먼저 선언했다면 함수의 프로토타입을 선언하지 않아도 됨

6.1 함수 - 3) 사용자 정의 함수의 구조 함수원형과 함수위치

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

```
//평균을 구하는 avg 함수의 프로토타입  
int avg(int i, int j);
```

```
void main( )  
{  
    //변수의 선언  
    int a,b,c;  
    //값입력  
    printf("첫번째 값:");  
    scanf_s("%d",&a);  
    printf("두번째 값:");  
    scanf_s("%d",&b);  
  
    //avg 함수 호출  
    c = avg(a,b);  
  
    printf("%d와 %d의 평균값은 %d입니다.\n",a,b,c);  
}
```

```
int avg(int i, int j)  
{  
    float k;  
    k=(i+j)/2.0;  
    //처리결과 반환  
    return k;  
}
```

첫번째 값:5
두번째 값:6
5와 6의 평균값은 5입니다.

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

```
int avg(int i, int j)  
{  
    float k;  
    k=(i+j)/2.0;  
    //처리결과 반환  
    return k;  
}
```

```
void main( )  
{  
    //변수의 선언  
    int a,b,c;  
    //값입력  
    printf("첫번째 값:");  
    scanf_s("%d",&a);  
    printf("두번째 값:");  
    scanf_s("%d",&b);  
  
    //avg 함수 호출  
    c = avg(a,b);  
  
    printf("%d와 %d의 평균값은 %d입니다.\n",a,b,c);  
}
```

첫번째 값:5
두번째 값:6
5와 6의 평균값은 5입니다.

```
01 #include <stdio.h>
02
03 int plus(int v1, int v2)
04 {
05     int result;
06     result = v1 + v2;
07     return result;
08 }
09
10 int main( )
11 {
12     int hap;
13
14     hap = plus(100, 200);
15
16     printf("100과 200의 plus( ) 함수 결과는 : %d\n", hap);
17 }
```

----plus() 함수를 정의한다.

-----3행에서 받은 두 매개변수의 합을 구한다

-----plus() 함수를 호출한 곳에 result 값을 반환한다.

---매개변수 2개를 지정하여 plus() 함수를 호출하고 반환값은 hap에 저장한다.

결 과

100과 200의 plus() 함수 결과는 : 300

[예제] 다음의 코드를 calc()함수를 사용하여 변경하여 보자

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a,b, result, oper;
06     printf("계산 입력 (1:+, 2:-, 3:*, 4:/) : ");
07     scanf_s("%d", &oper);
08     printf("계산할 두 숫자를 입력 : ");
09     scanf_s("%d %d", &a, &b);
10     switch (oper)
11     {
12         case 1: result = a + b; break;
13         case 2: result = a - b; break;
14         case 3: result = a * b; break;
15         case 4: result = a / b; break;
16     }
17     printf("계산 결과는 : %d\n", result);
18     return 0;
19 }
```

```
01 #include <stdio.h>
02
03 int calc(int v1, int v2, int op)
04 {
05     int result;
06     switch ( op )
07     {
08         case 1: result = v1 + v2; break;
09         case 2: result = v1 - v2; break;
10         case 3: result = v1 * v2; break;
11         case 4: result = v1 / v2; break;
12     }
13     return result;
14 }
```

---매개변수 3개를 받아서 계산하는 함수이다.

-----매개변숫값에 따라서 '1: 덧셈, 2: 뺄셈, 3: 곱셈, 4: 나눗셈'을 실행한다.

-----계산 결과를 반환한다.

```
15
16 int main( )
17 {
18     int res, oper, a, b;
19
20     printf("계산 입력 (1:+, 2:-, 3:*, 4:/) : ");
21     scanf_s("%d", &oper);
22     printf("계산할 두 숫자를 입력 : ");
23     scanf_s("%d %d", &a, &b);
24
25     res = calc(a, b, oper) ;
26
27     printf("계산 결과는 : %d\n", res);
28 }
```

---계산 결과, 연산자, 입력 숫자에 대한 변수이다.

---연산자를 입력한다.

---계산할 두 숫자를 입력한다

---매개변수 3개를 넣고 calc() 함수를 호출한다.
계산 결과는 res에 저장한다.

계산 입력 (1:+, 2:-, 3:*, 4:/) :3
계산할 두 숫자를 입력 :7 8
계산 결과는 : 56

라이브러리 함수

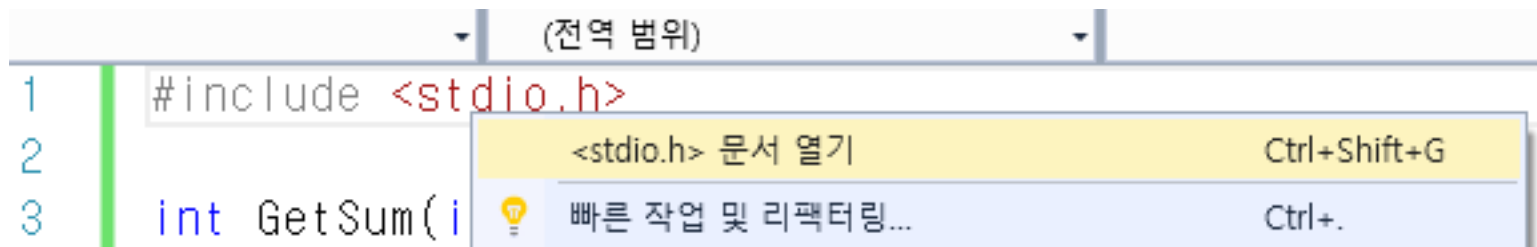
■ 라이브러리 함수(library function): 컴파일러에서 제공하는 함수

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



6.1. 함수 – 2) 표준함수와 헤더 파일

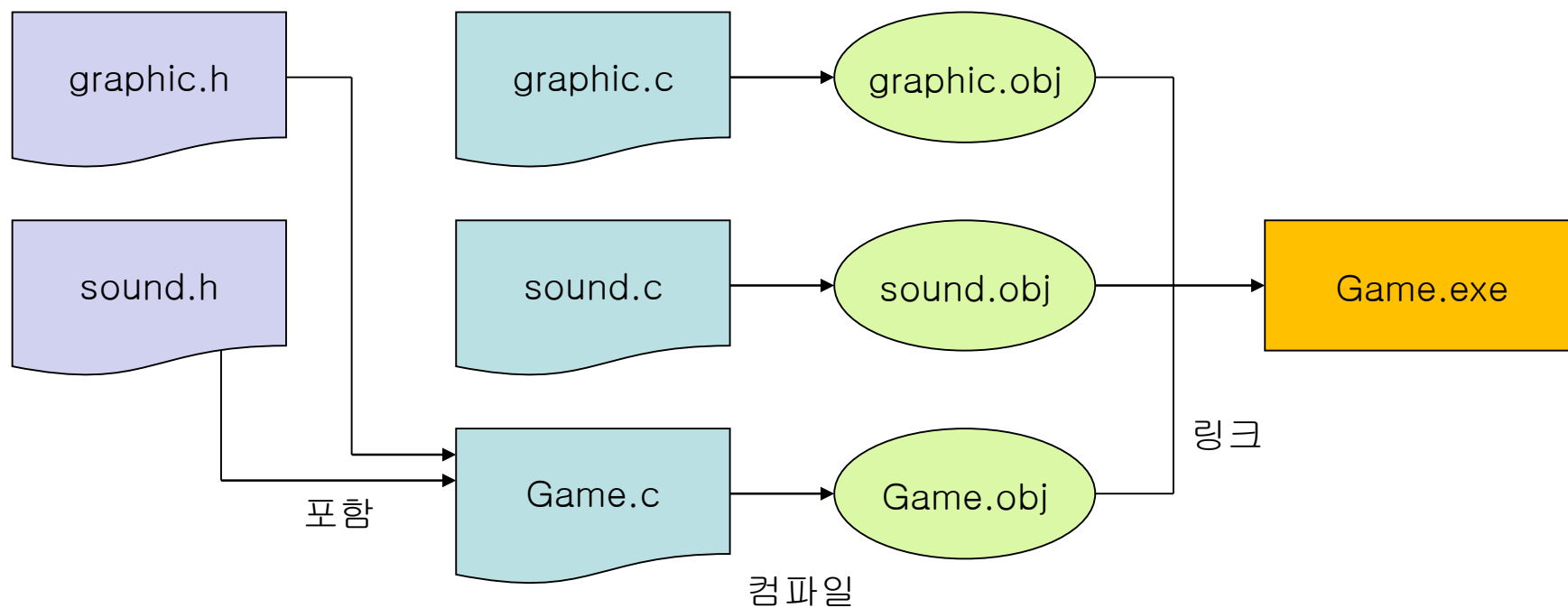
- 표준 함수의 원형을 선언해 놓은 것이며 #include문으로 포함시켜 사용한다.
- 소스창에서 열어 내용을 확인해 볼 수 있다.
- 기능별로 여러 개의 헤더 파일에 분산되어 있다. 사용하고자 하는 함수가 있는 헤더 파일을 포함시켜야 한다.



헤더 파일	함수	예
stdio.h	표준 입출력에 관한 함수	printf, scanf_s, puts
conio.h	키보드 및 화면 입출력 함수	getch, cprintf, kbhit
stdlib.h	자료 변환 함수들	atoi, itoa, malloc, free
math.h	수학 함수들	sin, cos, log
string.h	문자열 조작 함수들	strcpy, strlen

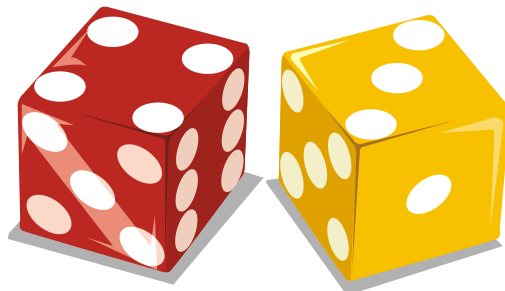
- 모듈 : 컴파일되는 단위이며 소스와 헤더의 쌍이다.
- 기능별로 여러개의 모듈로 분할하여 개발하며 링커로 합친다.

- 컴파일 속도가 빠르다.
- 분담 작업이 용이하다
- 재사용이 쉽다.



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

[예제] rand()사용하기

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(void)
05 {
06     int i;
07     for(i=0; i<6; i++)
08         printf("%d ", rand());
09
10     return 0;
11 }
```

0에서 32767 사이의 정수로 생성

결 과

41 18467 6334 26500 19169 15724

1부터 45 사이로 제한

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

결과

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.

```
srand( (unsigned)time( NULL ) );
```

```
#include <stdio.h>
#include <stdlib.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;
}
```

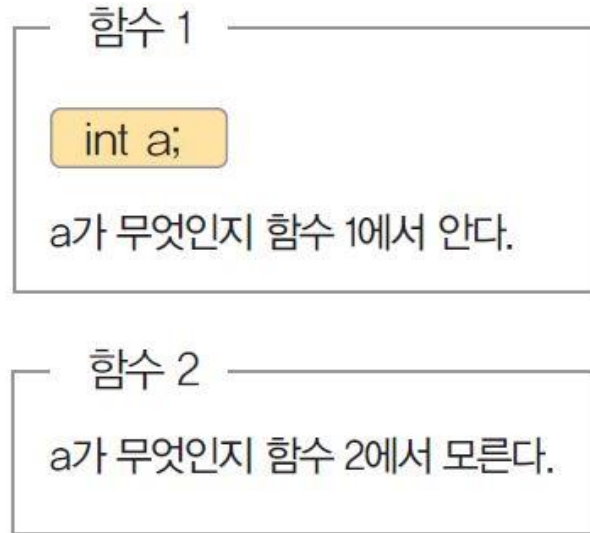
시드를 설정하는 가장 일반적인 방법은
현재의 시각을 시드로 사용하는 것이다.
현재 시각은 실행할 때마다 달라지기 때문이다.

6.2. 함수와 변수의 관계 - 2)지역변수와 전역변수

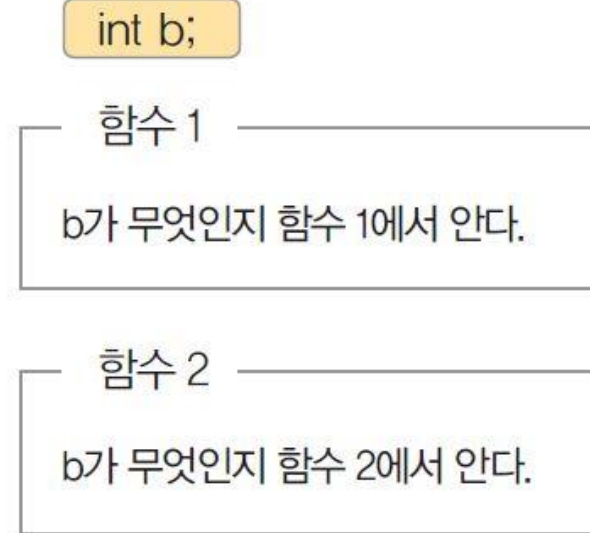
■ 지역변수와 전역변수의 이해

- 지역변수 : 한정된 지역(local)에서만 사용되는 변수
- 전역변수 : 프로그램 전체(global)에서 사용되는 변수

❶ 지역변수의 생존 범위



❷ 전역변수의 생존 범위



- ❶에서 a가 '함수 1' 안에 선언됨. 그러므로 a는 '함수 1' 안에서만 사용될 수 있고, '함수 2'에서는 a의 존재를 모름.
- ❷는 전역변수 b를 보여줌. b는 함수(함수 1, 함수 2) 안이 아니라 함수 바깥에 선언되어 있으므로 모든 함수에서 b의 존재를 알게 됨.

```
01 #include <stdio.h>
02
03 int a = 100; ----전역변수 a를 선언하고 초기값을 대입한다.
04
05 void func1( )
06 {
07     int a = 200; ----지역변수 a를 선언하고 초기값을 대입한다.
08     printf ("func1( )에서 a의 값== > %d\n", a); ----지역변수를 출력한다.
09 }
10
11 int main( )
12 {
13     func1( ); ----함수를 호출한다.
14     printf ("main( ) 에서 a의 값== > %d\n", a); ----전역변수를 출력한다.
15 }
```

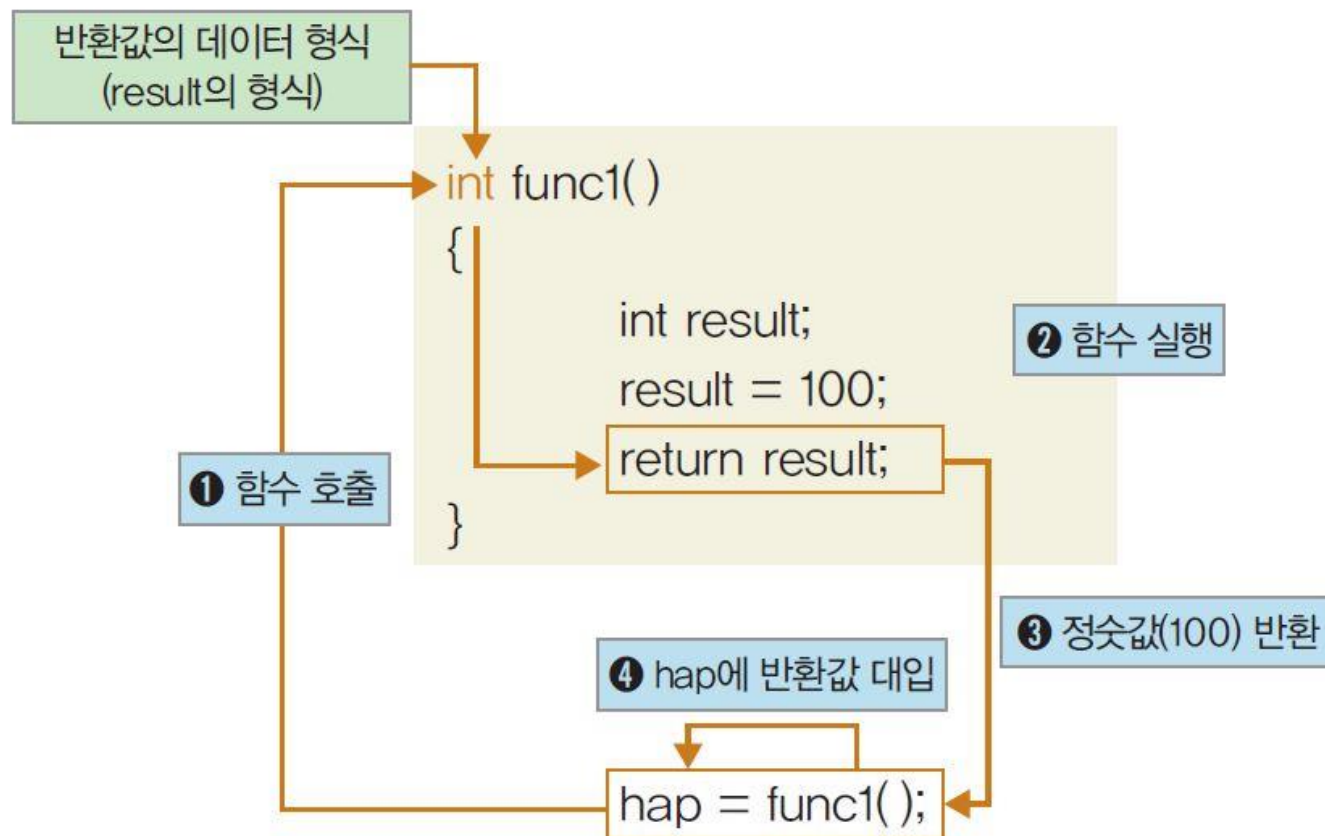
func1()에서 a의 값== > 200
main() 에서 a의 값== > 100

3. 함수의 반환값과 매개변수

■ 반환값 유무에 따른 함수 구분

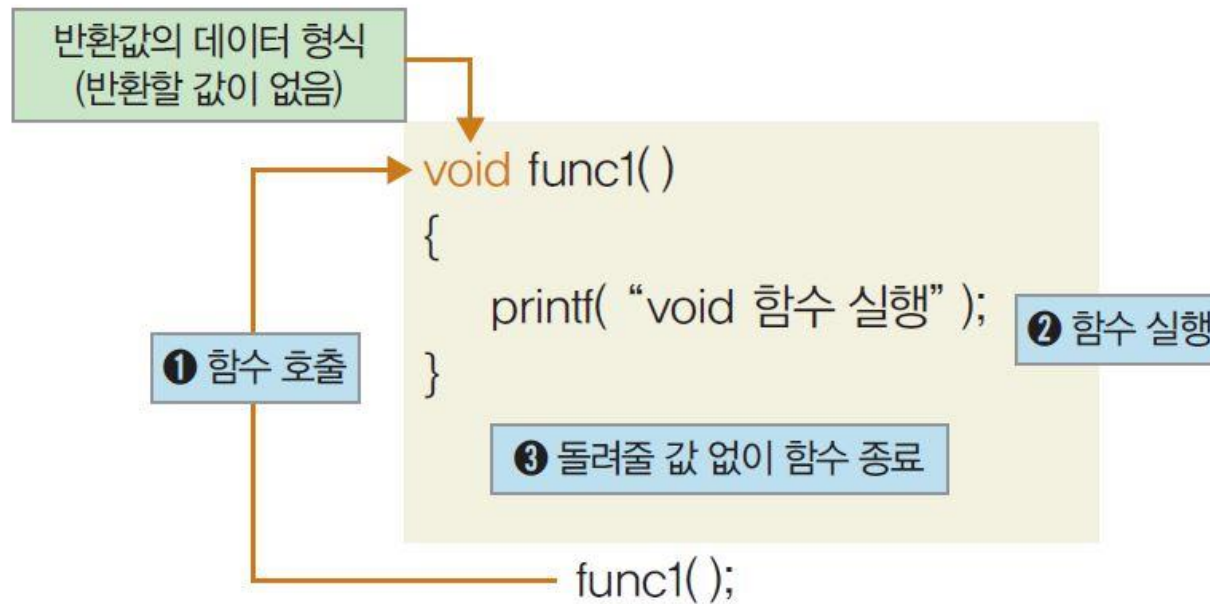
● 반환값이 있는 함수

- 함수를 실행한 결과값은 함수의 데이터형을 따름
- 'int 함수 이름()'으로 정의했다면 결과도 정수형 변수나 정숫값이어야 함.
 - 'return 정수형 변수;' 또는 'return 정수;'로 표현해야 함



3. 함수의 반환값과 매개변수

- 반환값이 없는 함수
 - 함수를 실행한 결과로 돌려줄 것이 없는 경우
 - 함수의 데이터형을 void로 표시 : void 형 함수를 호출할 때는 함수 이름만 쓴다.



3. 함수의 반환값과 매개변수 - 반환값 유무에 따른 함수 비교

```

01  #include <stdio.h>
02
03  void func1( )      ---void 형 함수이므로 반환값이 없다.
04  {
05      printf("void 형 함수는 돌려줄게 없음.\n");
06  }
07
08  int func2( )      ---int 형 함수므로 반환값이 있다.
09  {
10      return 100;
11  }
12
13  int main( )
14  {
15      int a;
16
17      func1( );      ---void 형 함수를 호출한다.
18
19      a = func2( );  ---int 형 함수를 호출한다.
20      printf("int 형 함수에서 돌려준 값 == > %d\n", a);
21  }

```

```

C:\Windows\system32\cmd.exe
void 형 함수는 돌려줄게 없음.
int 형 함수에서 돌려준 값 ==> 100

```

3. 함수의 반환값과 매개변수

■ 매개변수 전달 방법

- 값으로 전달(call by value)
 - 숫자나 문자 등의 값 자체를 함수에 넘겨주는 방법
 - 원개 값을 전달한 곳에는 아무런 영향을 미치지 않음
- 주소(또는 참조)의 전달
 - 주소값(address)을 함수에 넘겨주는 방법

3. 함수의 반환값과 매개변수 - 매개변수 전달 방법: 값으로 전달

```

01  #include <stdio.h>
02
03  void func1(int a)
04  {
05      a = a + 1;
06      printf("전달받은 a == > %d\n", a);
07  }
08
09  void main( )
10  {
11      int a=10;
12
13      func1(a);
14      printf("func1( ) 실행 후의 a == > %d\n", a);
15  }

```

---전달받은 a 값을 1 증가시킨 후 출력한다.

---변수 a를 선언한다.

---a 값을 매개변수로 넘겨 함수를 호출한다.

---함수를 호출한 후 a 값을 출력한다.

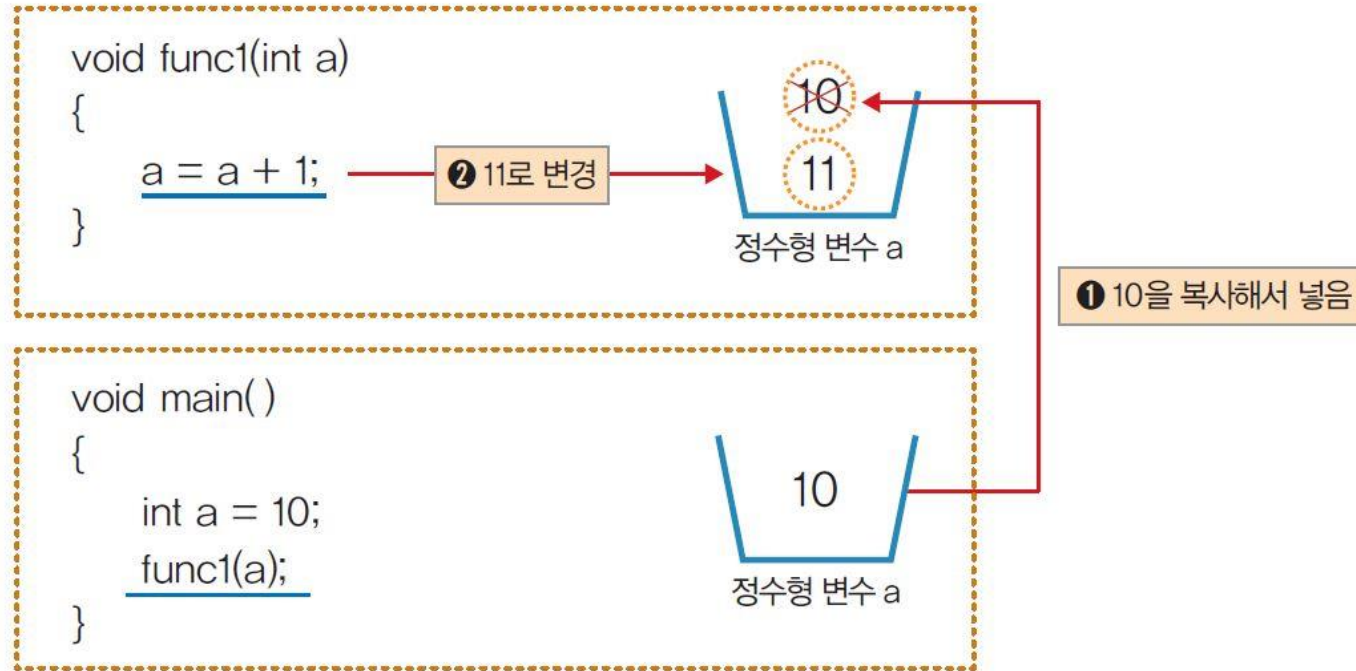
```

C:\Windows\system32\cmd.exe
전달받은 a ==> 11
func1(<) 실행 후의 a ==> 10

```

3. 함수의 반환값과 매개변수

- 값으로 전달

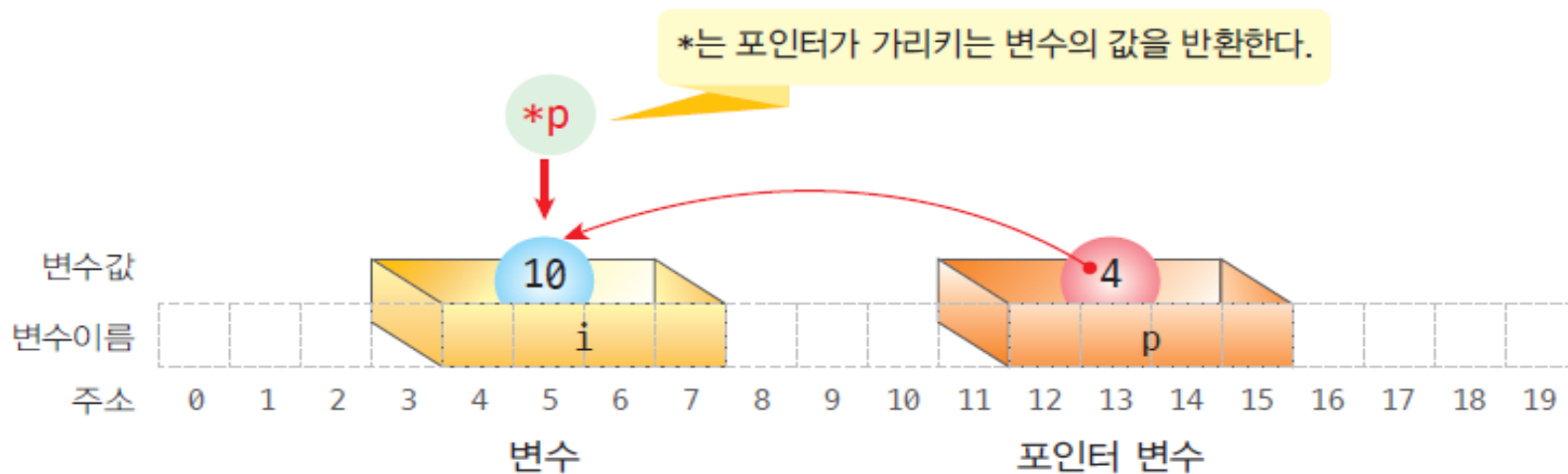


- `main()` 함수에서 `func1(a)` 호출. `func1()` 함수의 `a`에 10을 복사해서 넣음.
- `func1()` 함수에서는 `a` 값을 1 증가시켜서 11로 바꿈.
- `main()`의 `a`는 변경되지 않고 10을 유지함.

3. 함수의 반환값과 매개변수 - 매개변수 전달 방법: 주소로 전달

47 |

```
01 #include <stdio.h>
02
03 void func1(int *a)      ---매개변수로 주솟값(포인터)을 받는다.
04 {
05     *a = *a + 1;        ---a가 가리키는 곳의 실제 값 +1을 수행한다.
06     printf("전달받은 a == > %d\n", *a); ---a가 가리키는 곳의 실제 값을 출력한다.
07 }
08
09 void main( )
10 {
11     int a=10;           ---a를 10으로 초기화한다.
12
13     func1(&a);          ---함수를 호출할 때 a의 주소를 전달한다.
14     printf("func1( ) 실행 후의 a == > %d\n", a); ---함수를 호출한 후 a 값을 출력한다.
15 }
```

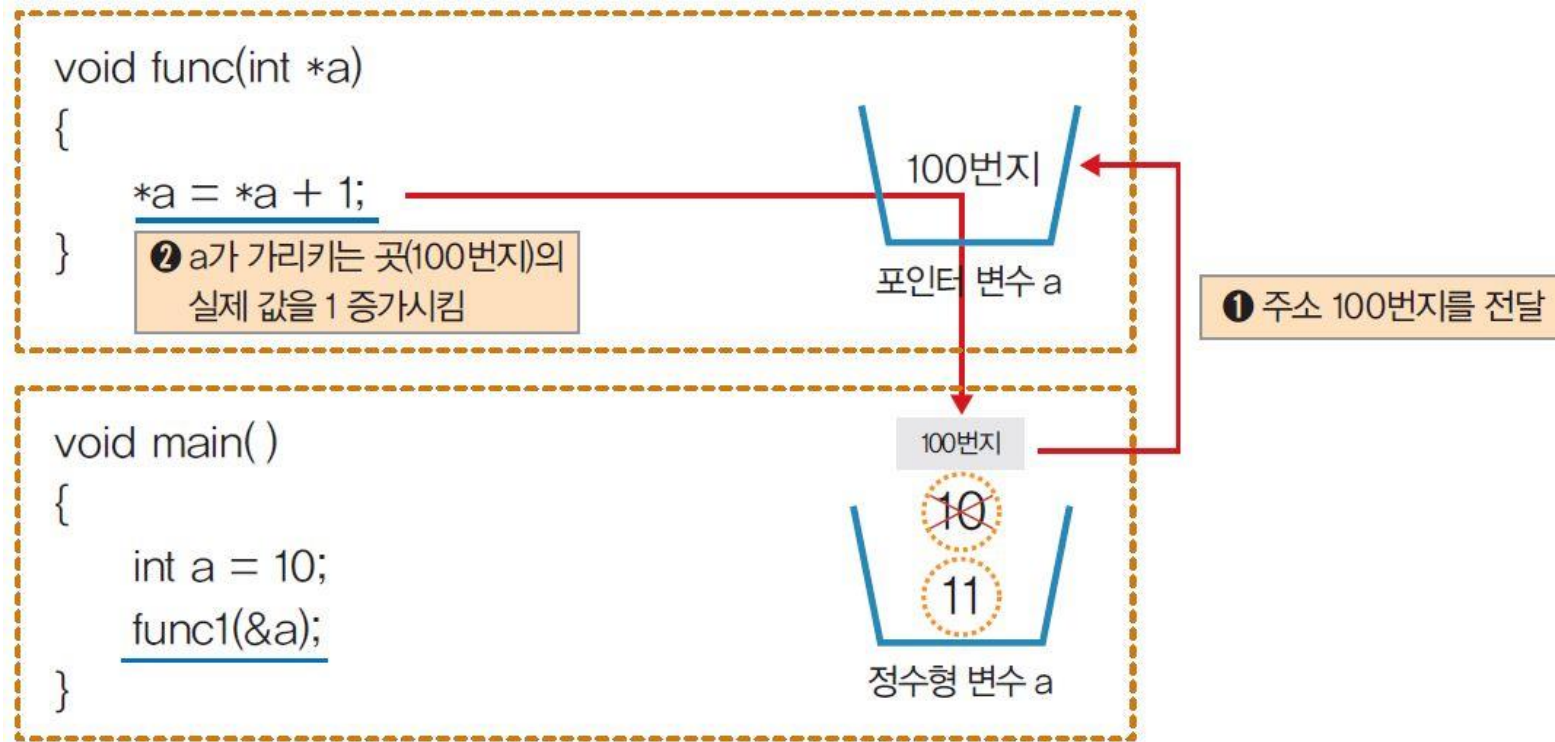


C:\#Window

```
전달받은 a ==> 11
func1(<) 실행 후의 a ==> 11
```

3. 함수의 반환값과 매개변수

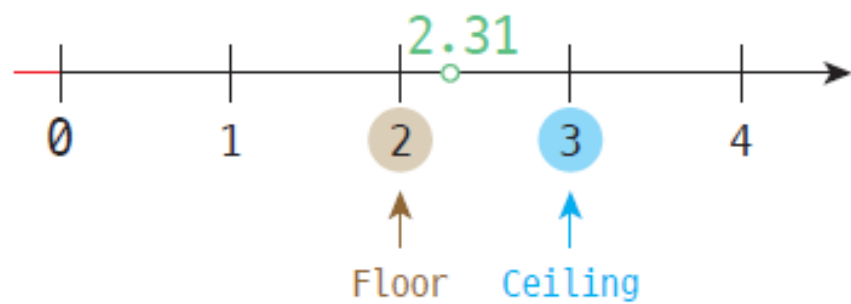
- 주소(또는 참조)로 전달



<code>int *a;</code>	⇒ 주소를 저장할 수 있는 포인터 변수를 선언한다.
<code>int b = 10;</code>	⇒ 정수형 변수이다.
<code>a = &b;</code>	⇒ a에 b의 주소를 대입한다.
<code>*a = 20;</code>	⇒ a가 가리키는 곳의 실제 값을 20으로 변경한다.

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()와 ceil() 함수



$\lfloor x \rfloor$
floor(x)

$\lceil x \rceil$
ceil(x)

```
#include <stdio.h>
#include <math.h>
int main( )
{
    float a = 2.99f;
    printf("%.2f : %d\n", a, (int)a);
    printf("round: %.2f : %d\n", a, (int)(a+0.5f));
    printf("ceil: %.2f : %d\n", a, (int)ceil(a));
    printf("floor: %.2f : %d\n", a, (int)floor(a));
    return 0;
}
```

- 함수가 자기 자신을 다시 호출하는 것을 재귀호출(recursive call)이라 한다.
- 재귀 호출에 사용되는 함수를 재귀함수라고 한다.
- 재귀 함수를 사용하면 반복적이거나 연결성을 가지는 결과를 출력 할 수 있다.
- 변수를 최대한 줄여서 프로그램의 오류를 줄일 수 있다.

```
#include <stdio.h>

int f(int n)
{
    int fact = 1;
    if(n==1)
        fact = 1;
    else
        fact = f(n-1)*n;
    return fact;
}

void main()
{
    int n;
    scanf_s("%d",&n);
    printf("%d\n",f(n));
}
```

```
f(10)
=f(9)*10
=(f(8)*9)*10
=((f(7)*8)*9)*10
=(((f(6)*7)*8)*9)*10
=(((((f(5)*6)*7)*8)*9)*10
=((((((f(4)*5)*6)*7)*8)*9)*10
=((((((((f(3)*4)*5)*6)*7)*8)*9)*10
=(((((((((((f(2)*3)*4)*5)*6)*7)*8)*9)*10
=((((((((((((((1*2)*3)*4)*5)*6)*7)*8)*9)*10
=3628800
```

```
#include <stdio.h>

int f(int n)
{
    int sum = 0;
    if(n==1)
        sum = 1;
    else
        sum = f(n-1)+n;
    return sum;
}

void main()
{
    int n;
    scanf_s("%d",&n);
    printf("%d\n",f(n));
}
```

```
f(10)
=f(9)+10
=(f(8)+9)+10
=((f(7)+8)+9)+10
=(((f(6)+7)+8)+9)+10
=(((((f(5)+6)+7)+8)+9)+10
=((((((f(4)+5)+6)+7)+8)+9)+10
=((((((((f(3)+4)+5)+6)+7)+8)+9)+10
=((((((((((f(2)+3)+4)+5)+6)+7)+8)+9)+10
=(((((((((((((1+2)+3)+4)+5)+6)+7)+8)+9)+10
=55
```

```
#include <stdio.h>

int f(int n)
{
    if(n>0){
        f(n-1);
        printf("%d",n);
        printf("*");
    }
}

void main()
{
    int n;
    scanf_s("%d",&n);
    f(n);
}
```

```
5
*****
```

```

01  #include <stdio.h>
02  char star[20];
03
04  int f(int n) {
05      if (n > 0) {
06          f(n - 1);
07          star[n]='*';
08          printf("%s\n",star+1);
09      }
10  }
11
12  void main() {
13      int n;
14      scanf_s("%d", &n);
15      f(n);
16  }

```

```

5
*
**
***
****
*****

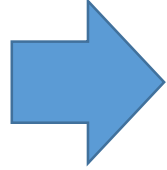
```

n값의 변화에 대한 star배열의 변화

n=1	<div> <div>0</div> <div>*</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> </div> <div> star <div>0</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
n=2	<div> <div>0</div> <div>*</div> <div>*</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> </div> <div> star <div>0</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
n=3	<div> <div>0</div> <div>*</div> <div>*</div> <div>*</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> </div> <div> star <div>0</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
n=4	<div> <div>0</div> <div>*</div> <div>*</div> <div>*</div> <div>*</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> </div> <div> star <div>0</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
n=5	<div> <div>0</div> <div>*</div> <div>*</div> <div>*</div> <div>*</div> <div>*</div> <div>0</div> <div>0</div> <div>0</div> <div>0</div> </div> <div> star <div>0</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>

- 5행에서 n이 음수일 경우에 대해서는 아무것도 하지 않고 재귀호출 종료
- 7행에서 f(n)을 그리기 위해서 f(n-1)을 재귀호출, 즉 이미f(n-1)이 그려져 있다고 가정
- 8행은 '*'모양 n개를 그리기 위해서 문자열의 n번째 공간에 '*'를 저장
- 9행은 서식지정자 "%s"를 이용하여 star[1]번 방의 내용부터 문자열 형태로 출력

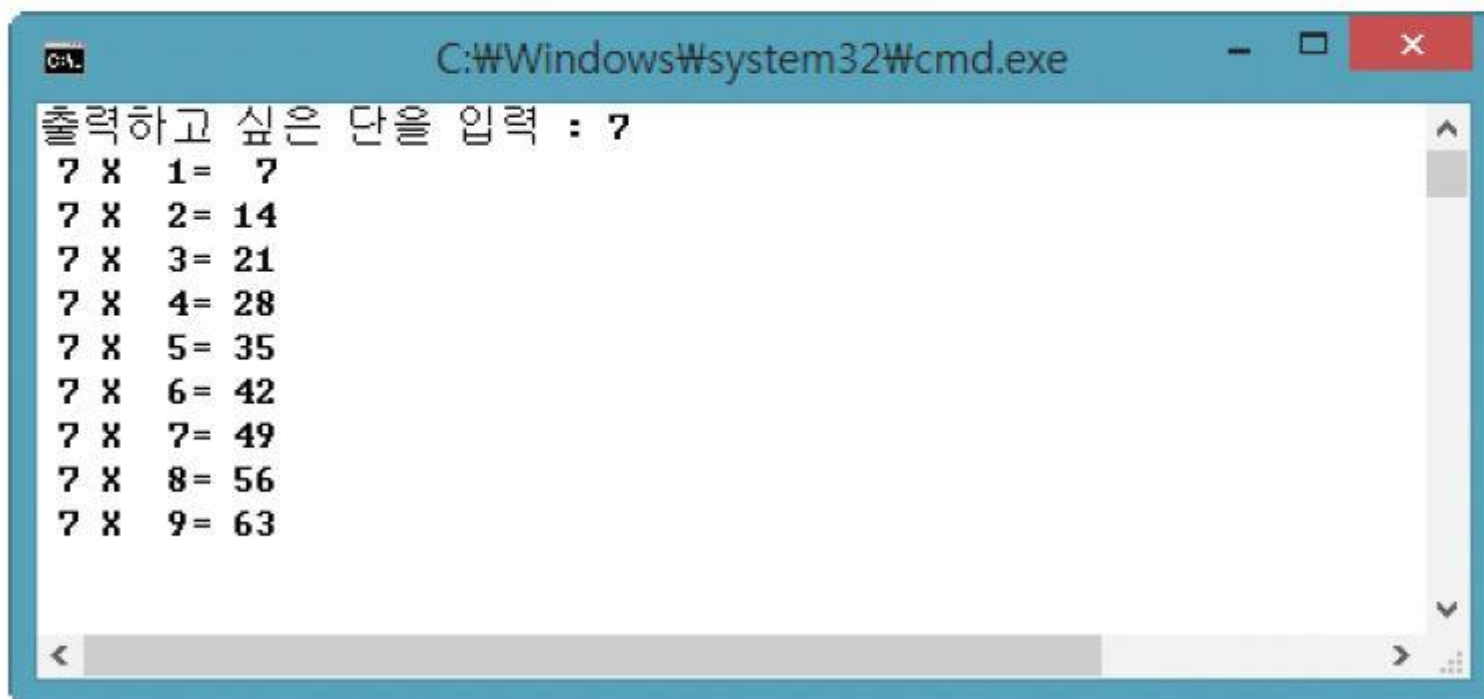

```
01  #include <stdio.h>
02  char star[20];
03
04  int f(int n) {
05      if (n > 0) {
06          f(n - 1);
07          star[n]='*';
08          printf("%s\n",star+1);
09      }
10  }
11
12  void main() {
13      int n;
14      scanf_s("%d", &n);
15      f(n);
16  }
```



```
int f(int n) {
    f(n - 1);
    printf("***** .... *****"); -----n개의 별모양 출력
}
```

예제 설명 함수를 사용하여 구구단을 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

출력하고 싶은 단을 입력 : 7
7 x 1= 7
7 x 2= 14
7 x 3= 21
7 x 4= 28
7 x 5= 35
7 x 6= 42
7 x 7= 49
7 x 8= 56
7 x 9= 63
```

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

```
02
```

```
03 void gugu(int dan)
```

```
04 {
```

```
05     int i;
```

```
06
```

```
07     for (i=1; i<=9; i++)
```

```
08     {
```

```
09         printf("%2d X %2d= %2d \n", dan, i, dan*i);
```

```
10     }
```

```
11 }
```

```
12
```

```
13 int main( )
```

```
14 {
```

```
15     int input;
```

```
16
```

```
17     printf("출력하고 싶은 단을 입력 : ");
```

```
18     scanf_s("%d", &input);
```

```
19
```

```
20     gugu(input);
```

```
21 }
```

---gugu() 함수를 정의한다(매개변수는 정수형 dan이다).

-----1~9를 반복하며 매개변수로 받은 dan의 단을 출력한다.

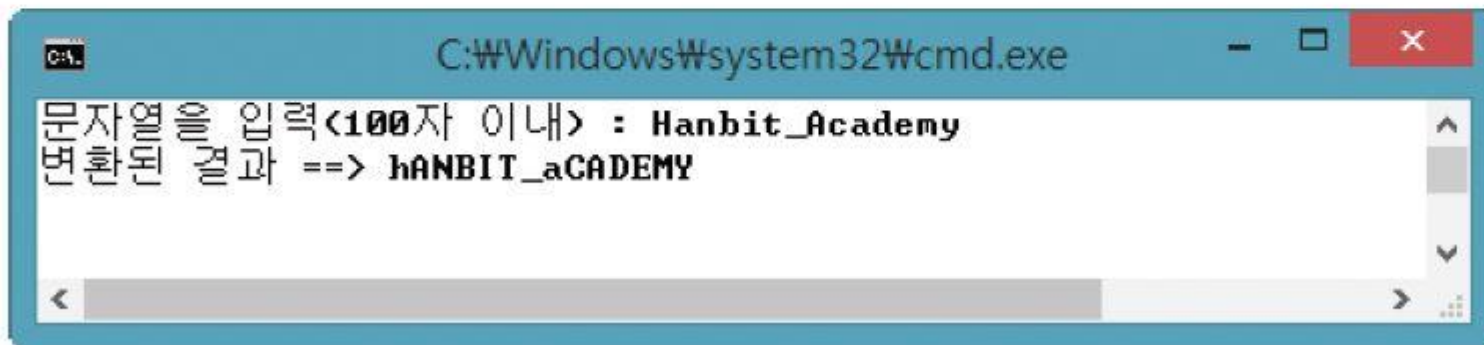
---출력할 단을 입력한다.

---구구단을 계산하고 출력할 함수를 출력한다.

예제 설명 대문자는 소문자로, 소문자는 대문자로 변환하는 프로그램이다.

- ① 대문자 변환 방법: 소문자에서 대·소문자 차이를 뺀다.
- ② 소문자 변환 함수: 대문자에서 대·소문자 차이를 더한다.

실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력<100자 이내> : Hanbit_Academy
변환된 결과 ==> hANBIT_aCADEMY
```

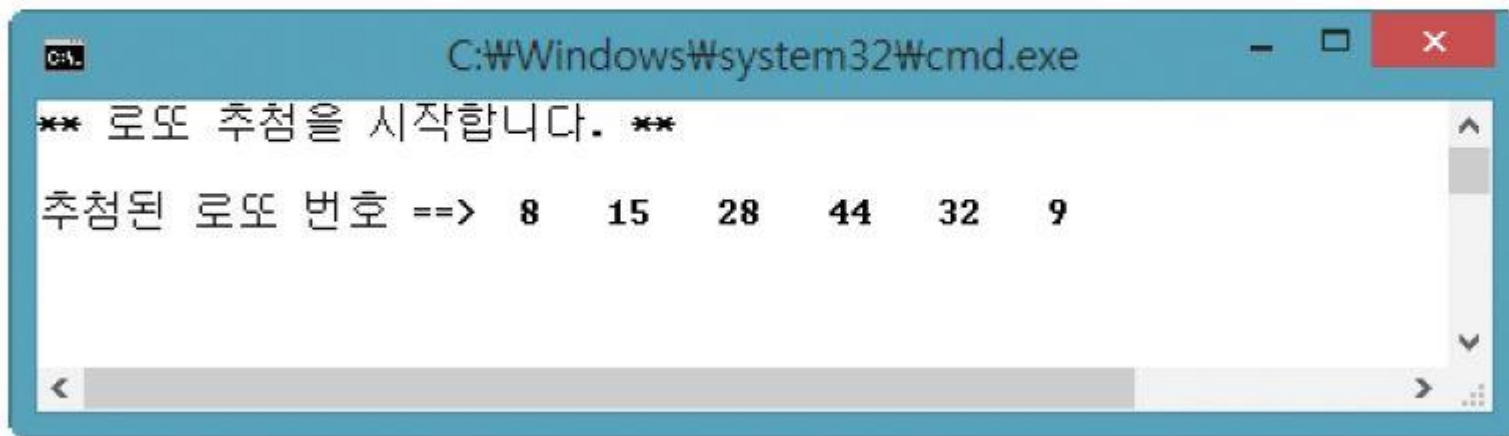
```

01  #include <stdio.h>
02
03  char upper(char ch) -----대문자로 변환하는 함수이다.
04  { return ch - ('a' - 'A'); }
05
06  char lower(char ch) -----소문자로 변환하는 함수이다.
07  { return ch + ('a' - 'A'); }
08
09  int main( )
10  {
11      char in[100], out[100];
12      char ch;
13      int i = 0;
14
15      printf("문자열을 입력(100자 이내) : ");
16      scanf_s("%s", in, sizeof(char)); -----문자열을 입력받는다.
17
18      do { -----문자열이 널이 아닌 동안 반복한다.
19          ch = in[i]; -----문자형 배열에서 한 문자만 추출한다.
20          if(ch >= 'A' && ch <= 'Z')
21              out[i] = lower(ch);
22          else if(ch >= 'a' && ch <= 'z') -----문자가 대문자이면 lower()함수를,
23              out[i] = upper(ch); -----소문자이면 upper()함수를 호출한다.
24          else -----숫자나 기호 등은 그대로 사용한다.
25              out[i] = ch;
26          i++;
27      } while (ch != '\0');
28
29      out[i] = '\0'; -----출력 문자열의 맨 뒤에 널 문자를 추가한다.
30      printf("변환된 결과 == > %s\n",out);
31  }

```

예제 설명 1~45 중에서 숫자 6개를 자동으로 뽑는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

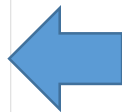
** 로또 추첨을 시작합니다. **

추첨된 로또 번호 ==>  8   15   28   44   32   9
```

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains two lines of text: the first line is "** 로또 추첨을 시작합니다. **" and the second line is "추첨된 로또 번호 ==> 8 15 28 44 32 9". The window has a standard Windows title bar with minimize, maximize, and close buttons.

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <time.h>
04
05 int getNumber( ) {
06     return rand( ) % 45 + 1;
07 }
08
```

```
if(dup == 'N')
    lotto[i++] = num;
else
    dup = 'N';
```



```
if(dup == 'N'){
    lotto[i] = num;
    i++;
}
else{
    dup = 'N';
}
```

```
09 int main( )
10 {
11     int lotto[6] = {0,}; ----추첨된 숫자를 담을 배열이다.
12     int i, k, num; ----반복 변수 i, k와 뽑힌 숫자를 담을 변수 num이다.
13     char dup = 'N'; ----이미 뽑힌 숫자인지 체크하기 위한 변수이다.
14
15     printf("** 로또 추첨을 시작합니다. ** \n\n");
16     srand((unsigned)time(NULL)); ----rand() 함수를 초기화하는 함수이다. 이 행이 없으면 늘 같은 숫자가 뽑힌다.
17
18     for (i=0; i<6; i++) { ----다른 숫자 6개가 뽑힐 때까지 반복한다(18~29행).
19         num = getNumber( ); ----다른 숫자가 뽑히면 18행에서 i를 1 증가시킨다.
20         ----로또 숫자를 1개 뽑는다.
21         for(k=0; k<6; k++){
22             if (lotto[k] == num) ----뽑은 숫자가 이미 뽑은 숫자와 동일한지 체크하고,
23                 dup = 'Y'; ----동일하면 중복 확인 변수에 'Y'를 대입한다.
24         }
25         if(dup == 'N') ----뽑은 숫자가 처음 뽑혔다면 로또 배열에 넣고
26             lotto[i++] = num; i(뽑힌 개수)를 1 증가시킨다(25~28행).
27         else ----아니면 다시 중복 확인 변수에 'N'을 대입한다.
28             dup = 'N';
29     }
30
31     printf("추첨된 로또 번호 == > ");
32     for (i = 0 ; i < 6 ; i++) { ----뽑힌 로또 숫자 6개를 출력한다.
33         printf("%d ", lotto[i]);
34     }
35
36     printf("\n\n");
37 }
```



```

01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <time.h>
04
05 int getNumber() {
06     return rand() % 45 + 1;
07 }
08
09 int main() {
10     int lotto[6] = {0,};
11     int i, k, num;
12     char dup = 'N';
13
14     printf("*** 로또 추첨을 시작합니다. ** \n\n");
15     srand((unsigned) time(NULL));
16
17     for (i = 0; i < 6;) {
18         num = getNumber();
19
20         for (k = 0; k < 6; k++){
21             if (lotto[k] == num)
22                 dup = 'Y';
23         }
24         if (dup == 'N')
25             lotto[i++] = num;
26         else
27             dup = 'N';
28     }
29
30     printf("추첨된 로또 번호 == > ");
31     for (i = 0; i < 6; i++) {
32         printf("%d ", lotto[i]);
33     }
34
35     printf("\n\n");
36 }

```

```

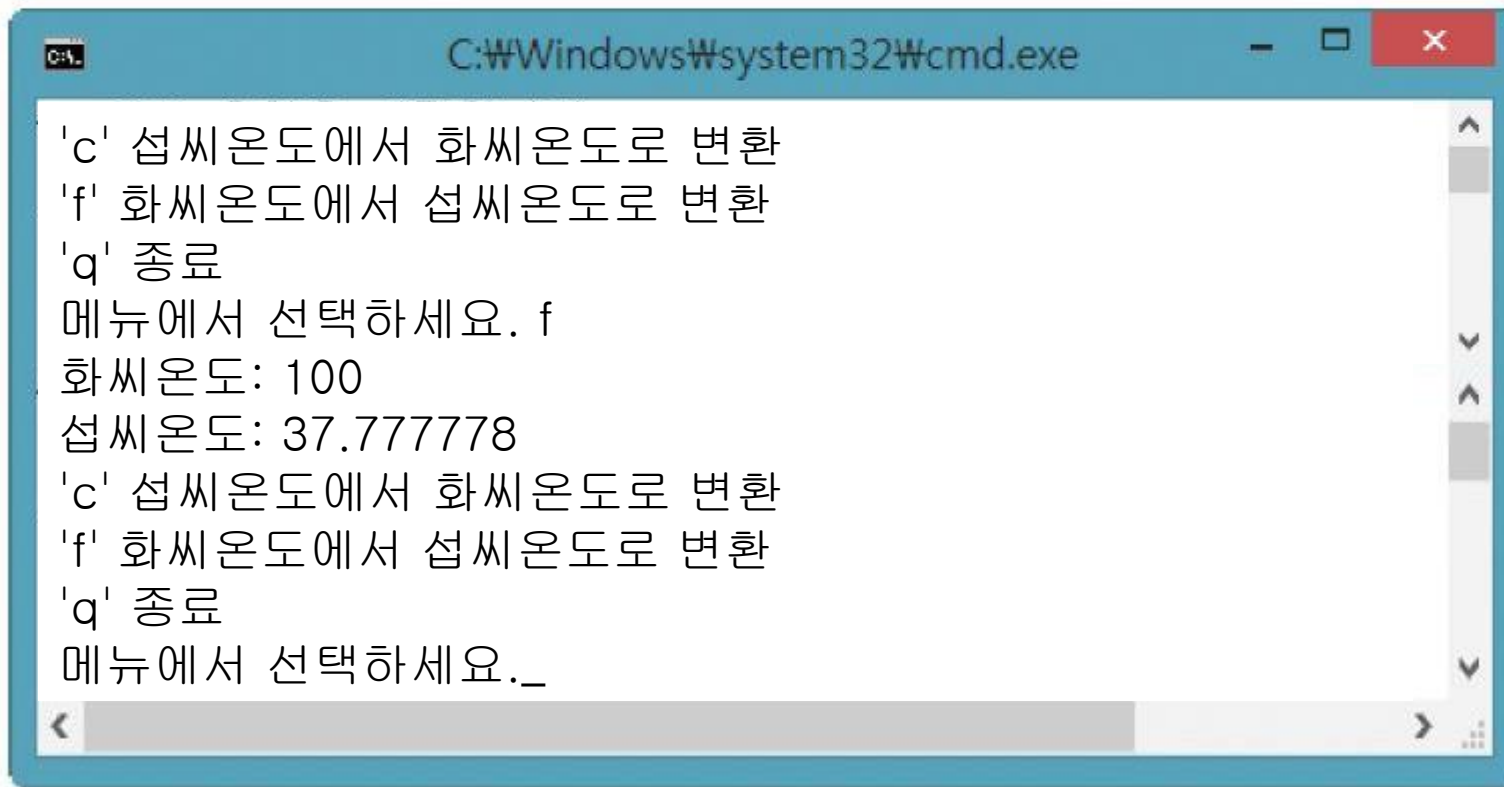
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <time.h>
04
05 int getNumber( ) {
06     return rand( ) % 45 + 1;
07 }
08 void getLottoNumber(int* lotto){
09     int num, i,k, dup='N';
10
11     for (i=0;i<6;) {
12         num = getNumber( );
13         for(k=0; k<6; k++){
14             if (lotto[k] == num)
15                 dup = 'Y';
16         }
17         if(dup == 'N')
18             lotto[i++] = num;
19         else
20             dup = 'N';
21     }
22 }
23 int main( ){
24     int lotto[6] = {0,};
25     int i, k; //int* ptr = &lotto[0];
26
27     printf("*** 로또 추첨을 시작합니다. ** \n\n");
28     srand((unsigned)time(NULL));
29     getLottoNumber(&lotto[0]); // getLottoNumber(ptr);
30
31     printf("추첨된 로또 번호 == > ");
32     for (i= 0 ; i < 6 ; i++) {
33         printf("%d ", lotto[i]);
34     }
35     printf("\n\n");
36 }

```

예제 설명

메뉴에서 c(섭씨→화씨 변환), f(화씨→섭씨), q(종료)를 입력받아 섭씨와 화씨를 변환하는 프로그램을 작성해 보자

실행 결과



```
C:\Windows\system32\cmd.exe

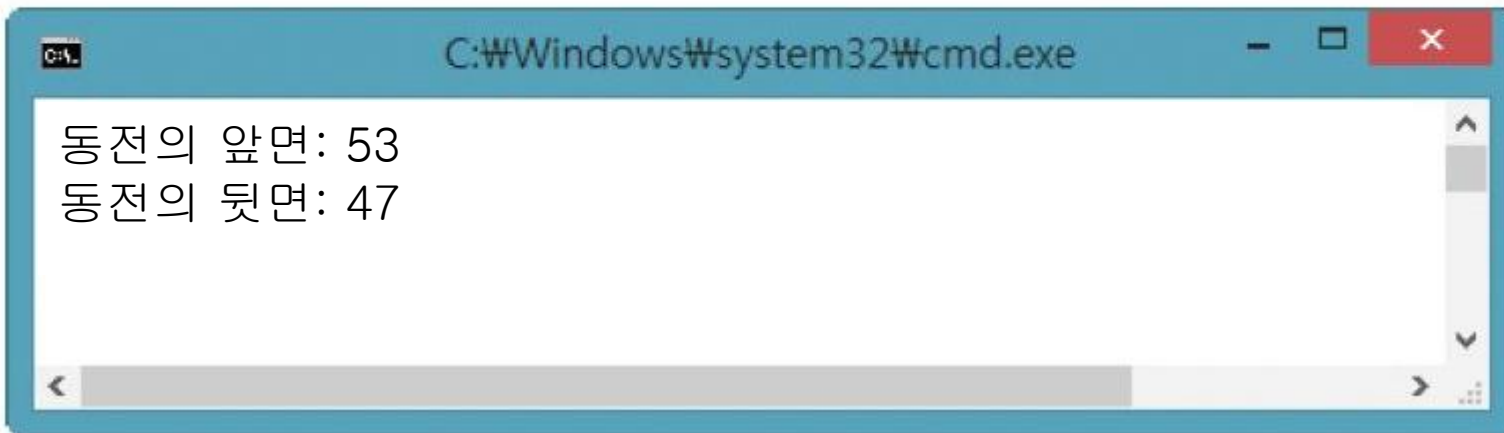
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
메뉴에서 선택하세요. f
화씨온도: 100
섭씨온도: 37.777778
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
메뉴에서 선택하세요._
```

```
01 #include <stdio.h>
02
03 void printOptions()
04 {
05     printf(" 'c' 섭씨온도에서 화씨온도로 변환\n");
06     printf(" 'f' 화씨온도에서 섭씨온도로 변환\n");
07     printf(" 'q' 종료\n");
08 }
09 double C2F(double c_temp)
10 {
11     return 9.0 / 5.0 * c_temp + 32;
12 }
13
14 double F2C(double f_temp)
15 {
16     return (f_temp - 32.0) * 5.0 / 9.0;
17 }
```

```
18 int main(void)
19 {
20     char choice;
21     double temp;
22     while (1) {
23         printOptions();
24         printf("메뉴에서 선택하세요.");
25         choice = getchar();
26         if (choice == 'q') break;
27         else if (choice == 'c') {
28             printf("섭씨온도: ");
29             scanf_s("%lf", &temp);
30             printf("화씨온도: %lf \n", C2F(temp));
31         }
32         else if (choice == 'f') {
33             printf("화씨온도: ");
34             scanf_s("%lf", &temp);
35             printf("섭씨온도: %lf \n", F2C(temp));
36         }
37         getchar(); // 엔터키 문자를 삭제하기 위하여 필요!
38     }
39     return 0;
40 }
```

예제 설명 동전을 100번 던져서 앞면이 나오는 횟수와 뒷면이 나오는 횟수를 출력한다.

실행 결과



```
C:\Windows\system32\cmd.exe

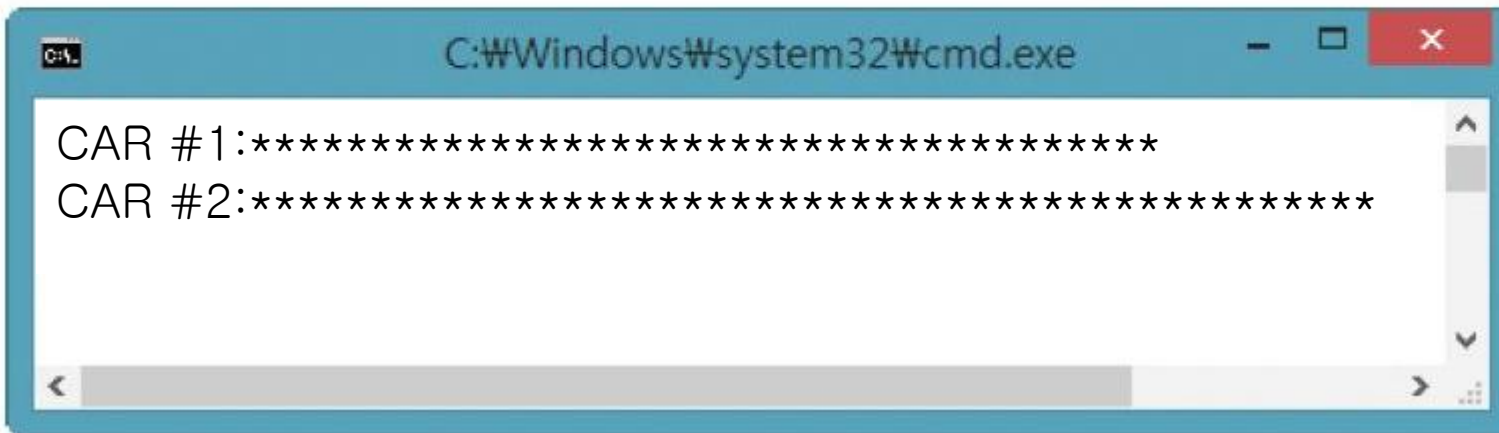
동전의 앞면: 53
동전의 뒷면: 47
```

The image shows a screenshot of a Windows command prompt window. The title bar indicates the path 'C:\Windows\system32\cmd.exe'. The window contains two lines of text: '동전의 앞면: 53' (Front of the coin: 53) and '동전의 뒷면: 47' (Back of the coin: 47). The window has a standard Windows interface with a blue title bar and a scroll bar on the right.

```
01 #include <stdlib.h>
02 #include <stdio.h>
03 #include <time.h>
04
05 int coin_toss( void );
06 int main( void )
07 {
08     int toss;
09     int heads = 0;
10     int tails = 0;
11     srand((unsigned)time(NULL));
12
13     for( toss = 0; toss < 100; toss++ ){
14         if(coin_toss( ) == 1)
15             heads++;
16         else
17             tails++;
18     }
19     printf( "동전의 앞면: %d \n", heads );
20     printf( "동전의 뒷면: %d \n", tails );
21     return 0;
22 }
23
24 int coin_toss( void )
25 {
26     int i = rand() % 2;
27     if(i == 0)
28         return 0;
29     else
30         return 1;
31 }
```

예제 설명 난수를 이용하여서 자동차 게임을 작성

실행 결과



```
C:\Windows\system32\cmd.exe

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

The screenshot shows a Windows command prompt window with the title bar 'C:\Windows\system32\cmd.exe'. The window contains two lines of text: 'CAR #1:*****' and 'CAR #2:*****'. The text is displayed in a monospaced font. The window has a standard Windows interface with a title bar, minimize, maximize, and close buttons, and a scroll bar on the right side.

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.

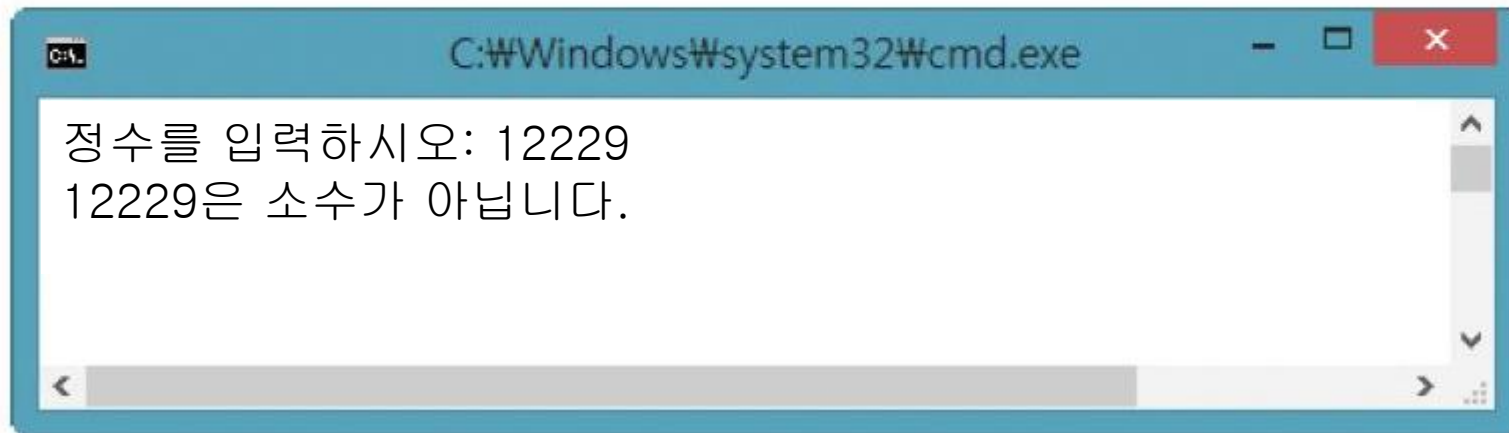
```
}
```

```
01 #include <stdlib.h>
02 #include <stdio.h>
03 #include <time.h>
04 void disp_car(int car_number, int distance);
05 int main(void)
06 {
07     int i;
08     int car1_dist=0, car2_dist=0;
09     srand((unsigned)time(NULL));
10
11     for( i = 0; i < 6; i++ ) {
12         car1_dist += rand() % 100;
13         car2_dist += rand() % 100;
14         disp_car(1, car1_dist);
15         disp_car(2, car2_dist);
16         printf("-----\n");
17         getch();
18     }
19     return 0;
20 }
21
22 void disp_car(int car_number, int distance)
23 {
24     int i;
25     printf("CAR #d:", car_number);
26     for( i = 0; i < distance/10; i++ ) {
27         printf("*");
28     }
29     printf("\n");
30 }
```

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

예제 설명 정수를 입력받아 소수를 찾는 프로그램을 작성하여 보자

실행 결과



[예제] 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

사용자로부터 정수를 입력받아서 변수 n 에 저장한다.

약수의 개수를 0으로 초기화한다.

`for($i=1$; $i \leq n$; $i++$)`

n 을 i 로 나누어서 나머지가 0인지 본다.

 나머지가 0이면 약수의 개수를 증가한다.

약수의 개수가 2이면 정수 n 은 소수이다.

```
01 #include <stdio.h>
02
03 int is_prime(int);
04 int get_integer(void);
05
06 int main()
07 {
08     int n, result;
09     n = get_integer();
10     result = is_prime(n);
11
12     if ( result == 1 )
13         printf("%d은 소수입니다.\n", n);
14     else
15         printf("%d은 소수가 아닙니다.\n", n);
16
17     return 0;
18 }
```

```
19 int get_integer(void)
20 {
21     int n;
22     printf("정수를 입력하시오: ");
23     scanf_s("%d", &n);
24
25     return n;
26 }
27
28 int is_prime(int n)
29 {
30     int divisors = 0, i;
31     for ( i = 1 ; i <= n ; i++ )
32     {
33         if ( n%i == 0 )
34             divisors++;
35     }
36
37     return (divisors == 2);
38 }
```