

C 프로그래밍

함수

목차

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



모듈의 개념

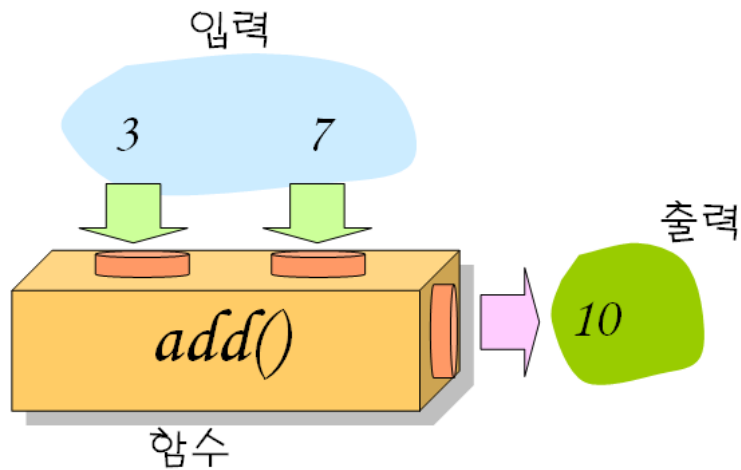


- 모듈(module)
 - 하나의 독립된 작업을 처리하는 프로그램의 일부분
- 모듈러 프로그래밍의 장점
 - 각 모듈들을 독자적으로 개발 가능
 - 다른 모듈과 독립적으로 변경 가능
 - 유지 보수 용이
 - 모듈의 재사용 가능
- C에서의 모듈==함수



함수의 개념

- 특정한 작업을 수행하는 독립적인 부분
- 함수 호출(function call)
 - 함수를 호출하여 사용하는 것
 - 입력을 받아서 수행 결과를 출력 또는 반환

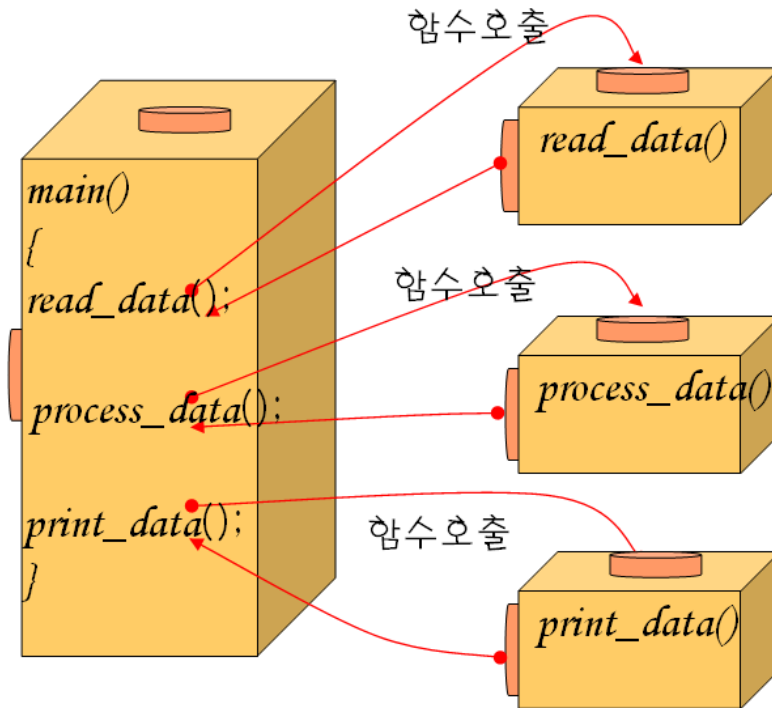


함수는 이름을 가지며 입력을 받아서 특정한 작업을 실행하고 결과를 반환합니다.



함수들의 연결

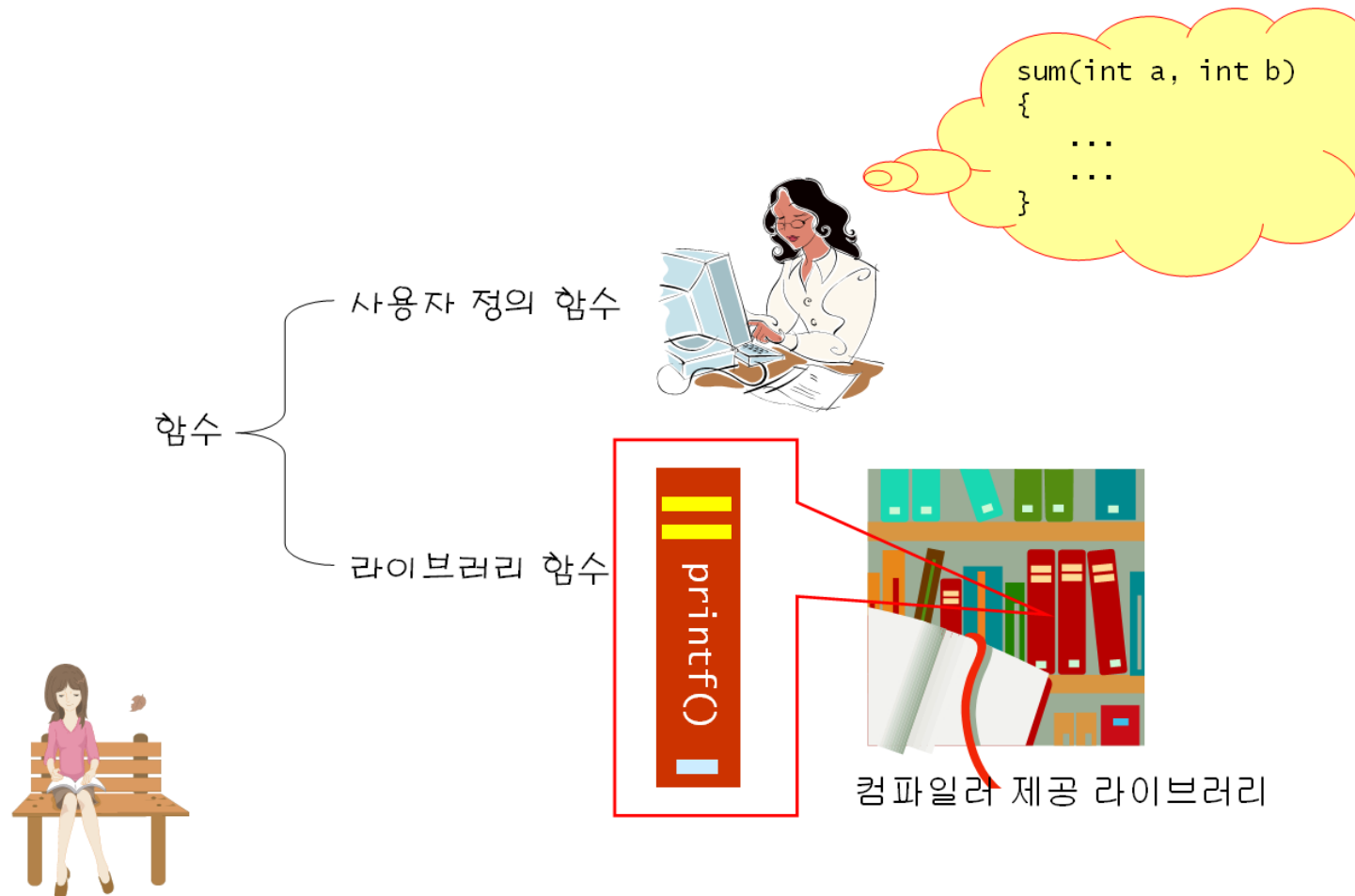
- 프로그램은 여러 개의 함수들로 구성
- 함수 호출을 통하여 함수간 연결
- `main()` : 제일 먼저 호출되는 시작 함수



각 함수들은 함수 호출을 통하여 서로 결합되어 프로그램을 구성합니다.

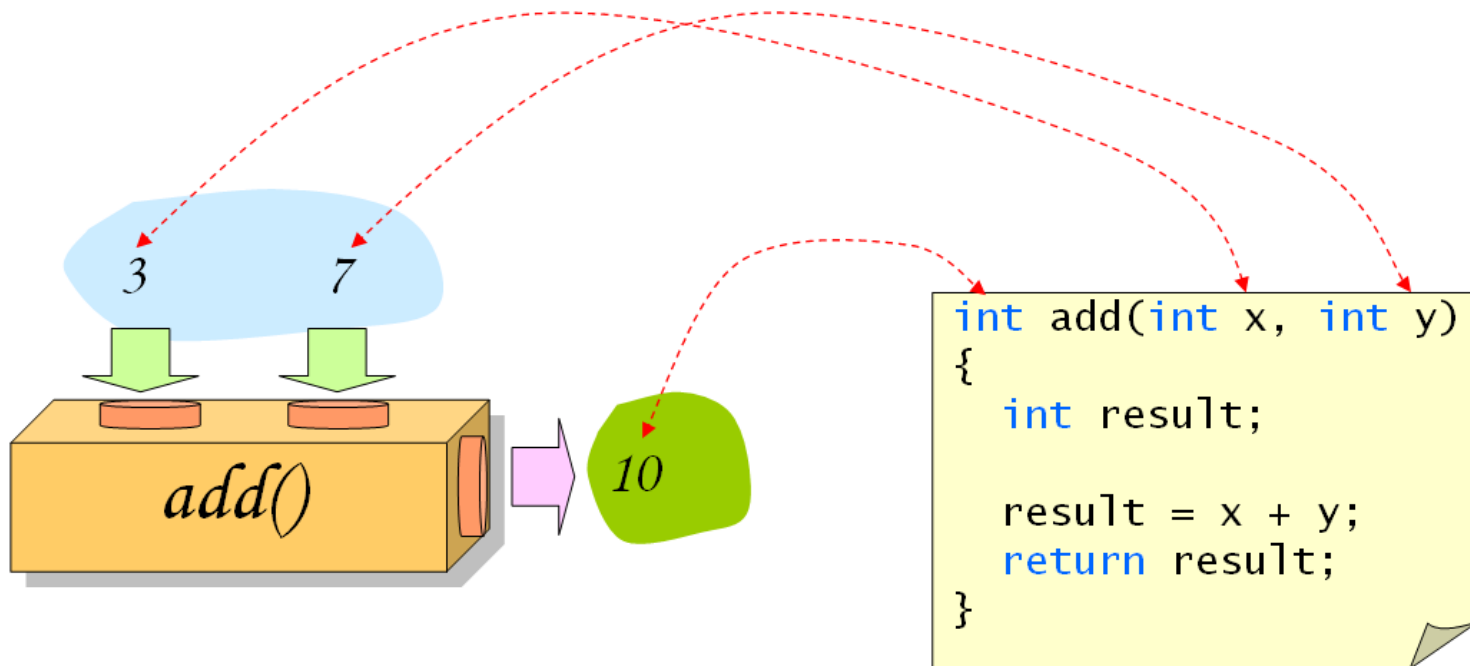


함수의 종류

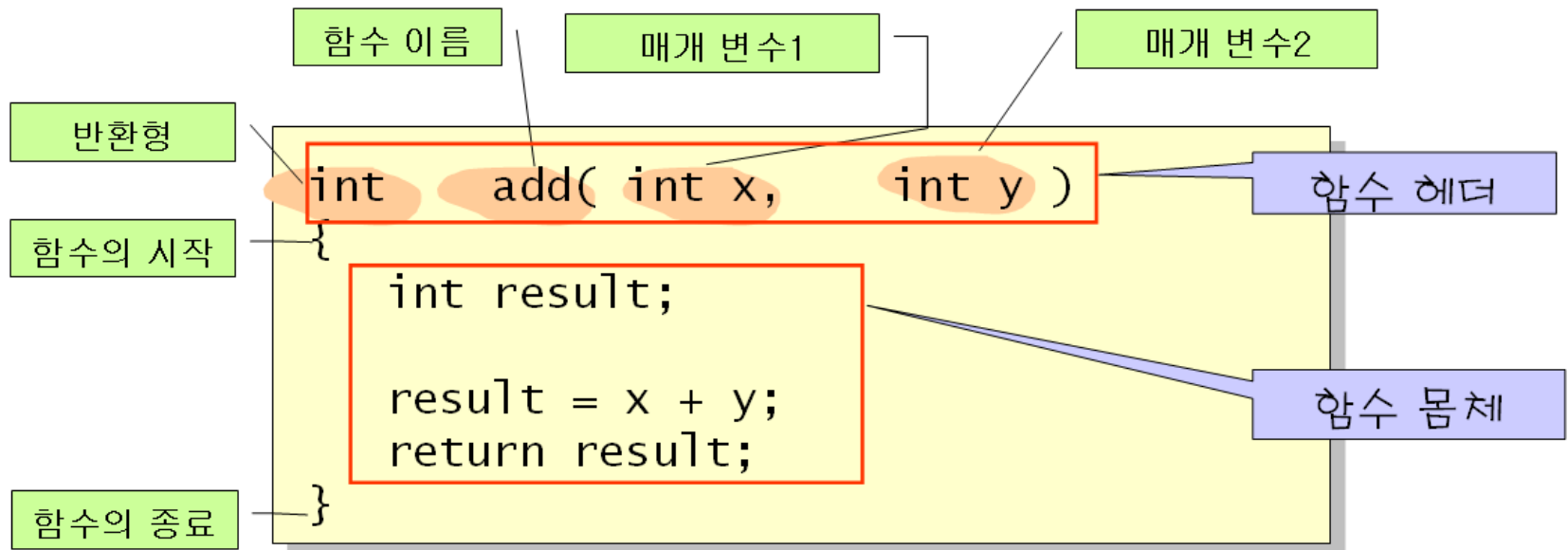


함수의 정의

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




함수의 구조



예제 #1

- 정수의 제곱값을 계산하는 함수
 - 함수로 전달되는 값에 대한 제곱을 구한 다음, 연산결과 반환

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




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



예제 #2

- 두 개의 정수 중에서 큰 수를 계산하는 함수
 - 함수로 전달되는 두 수 중에 큰 수를 판단한 후, 큰 수를 다시 반환

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



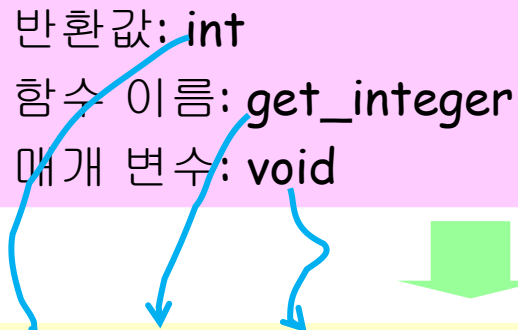
```
int get_max(int x, int y)
{
    if( x > y ) return(x);
    else return(y);
}
```



예제 #3

- 사용자로부터 한 개의 정수를 받아서 반환
 - 함수 내에서 사용자 입력을 받은 다음, 해당 값을 반환

반환값: `int`
함수 이름: `get_integer`
매개 변수: `void`



```
int get_integer(void)
{
    int n;

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

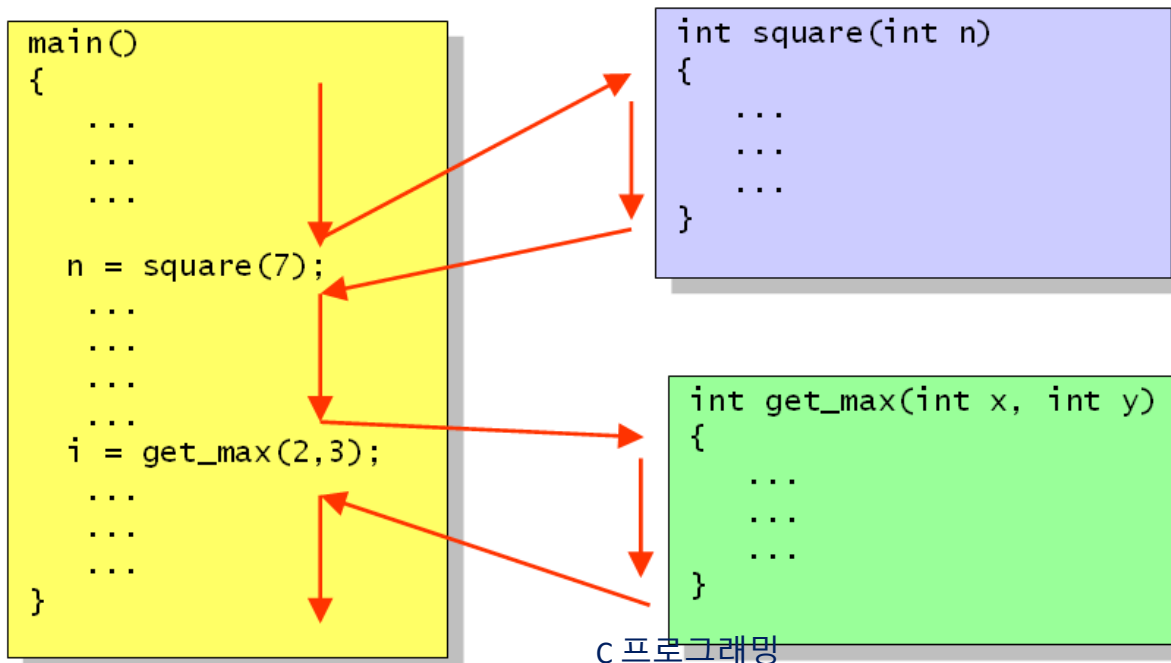
    return n;
}
```



함수 호출과 반환

● 함수 호출(function call):

- 함수를 사용하기 위하여 함수의 이름을 적어주는 것
- 함수 안의 문장들이 순차적으로 실행
- 문장의 실행이 끝나면 호출한 위치로 되돌아 감
- 결과값을 전달





- 화씨 온도를 섭씨 온도로 변경하는 함수 생성

$$c = \frac{5.0}{9.0} (f - 32)$$

- 함수 `f_to_C()` 정의
 - 사용자로부터 화씨 온도 입력 받음
 - 입력 받은 값에 대해 섭씨 온도 계산 후
 - 섭씨 온도 출력

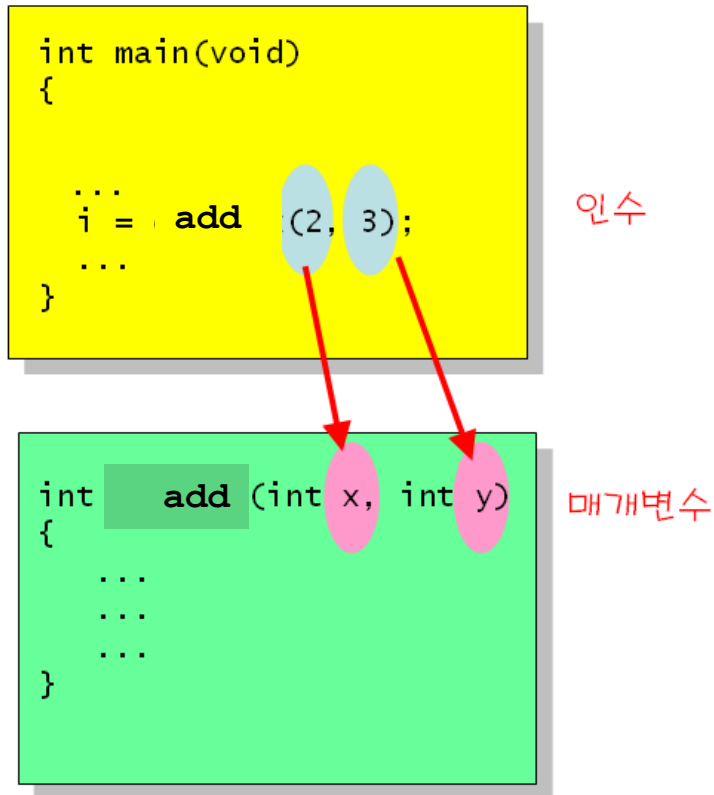
인수와 매개 변수



- 인수(argument)
 - 함수 호출 시, 넘겨 주는 값
 - 실인수, 실매개 변수
- 매개 변수(parameter)
 - 함수 정의부에서 인수를 전달 받기 위한 변수
 - 변수 선언 형식
 - 형식 인수, 형식 매개 변수



인수와 매개 변수



```
#include <stdio.h>
int add(int x, int y)
{
    return (x + y);
}

int main(void)
{
    // 2와 3이 add()의 인수가 된다.
    add(2, 3);

    // 5와 6이 add()의 인수가 된다.
    add(5, 6);
    return 0;
}
```

반환값



- 호출된 함수가 호출한 곳으로 작업의 결과값을 전달하는 것
- 인수는 여러 개가 가능하나 반환값은 하나만 가능

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



함수 원형을 사용하지 않는 예제



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

```
// 함수 정의
```

```
int compute_sum(int n)
```

```
{
```

```
    int i;
```

```
    int result = 0;
```

```
    for(i = 1; i <= n; i++)
```

```
        result += i;
```

```
    return result;
```

```
}
```

```
int main(void)
```

```
{
```

```
    int n, sum;
```

```
    printf("정수를 입력하시오: ");
```

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

```
    sum = compute_sum(n);           // 함수 사용
```

```
    printf("1부터 %d까지의 합은 %d입니다. \n", n, sum);
```

```
    return 0;
```

```
}
```



정수를 입력하시오: 10

1부터 10까지의 합은 55입니다.





- 화씨 온도를 섭씨 온도로 변경하는 함수 생성
- 함수 `f_to_C(temp)` 정의
 - 사용자로부터 입력 받은 화씨 온도를 전달 받음
 - `Main()` 함수에서 화씨 온도 입력 받은 후, 입력 받은 값을 함수 호출과 함께 전달
 - 전달 받은 값에 대해 섭씨 온도 계산 후
 - 섭씨 온도 반환
 - `Main()` 함수에서 섭씨 온도 출력



함수 원형 (Function Prototype)

- 컴파일러에게 함수에 대하여 미리 알리는 것

**반환형 함수이름(매개변수1, 매개변수
2, ...);**

// 정수의 제곱을 계산하는 함수 예제

#include <stdio.h>

int square(int n); // 함수 원형

int main(void)

{

int i, result;

for(i = 0; i < 5; i++)

{

result = square(i); // 함수 호출

printf("%d \n", result);

}

return 0;

}

int square(int n) // 함수 정의

{

return(n * n);

}



함수 원형 예제



```
#include <stdio.h>
// 함수 원형
int compute_sum(int n);
```

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

```
    int n, sum;
```

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

```
    sum = compute_sum(n);           // 함수 사용
```

```
    printf("1부터 %d까지의 합은 %d입니다. \n", n, sum);
```

```
}
```

```
int compute_sum(int n)
```

```
{
```

```
    int i;
```

```
    int result = 0;
```

```
    for(i = 1; i <= n; i++)
```

```
        result += i;
```

```
    return result;
```

```
}
```



정수를 입력하시오: 10
1부터 10까지의 합은 55입니다.



```
sum = compute_sum(3.5);
```

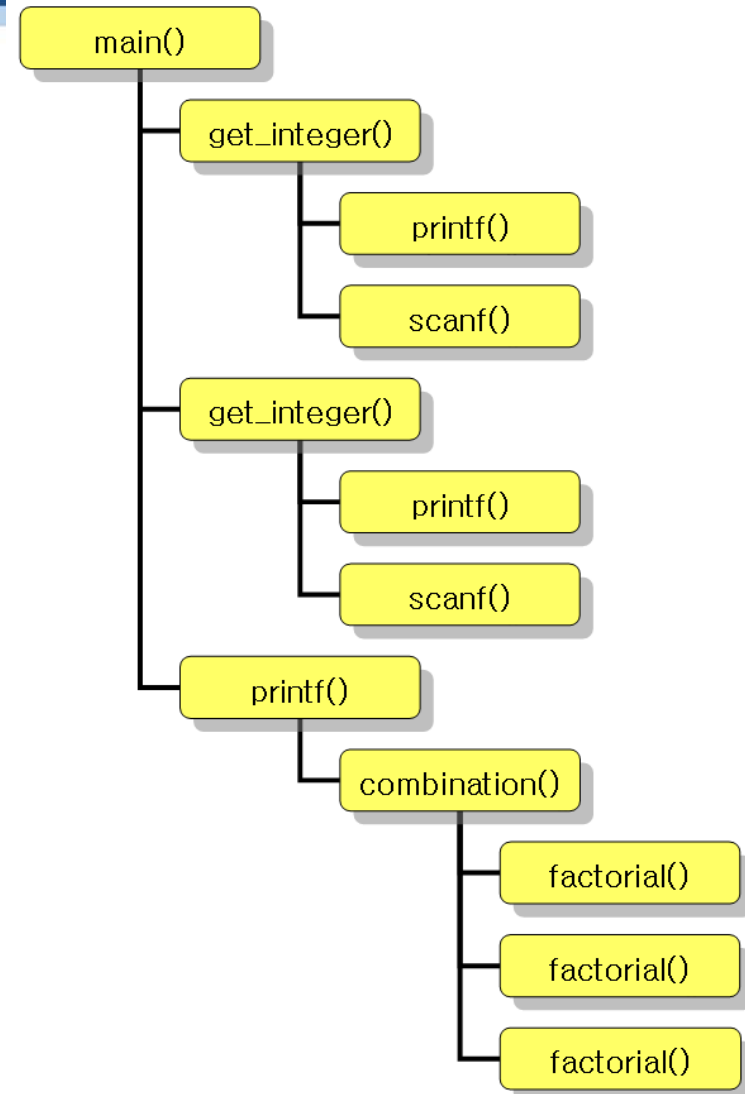


조합(combination) 계산 함수

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

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

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



예제



```
#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 프로그래밍



예제



```
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;
}
```

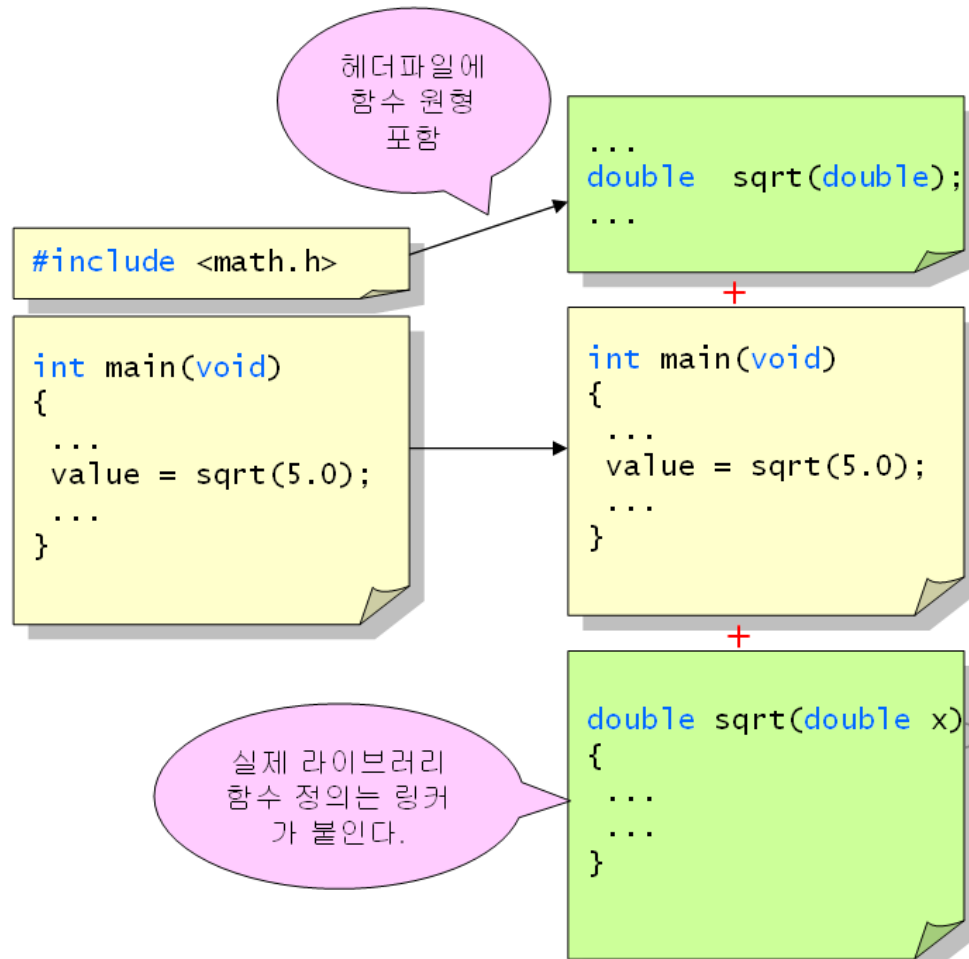
정수를 입력하시오: 10
정수를 입력하시오: 3
 $C(10, 3) = 120$



라이브러리 함수

● 컴파일러에서 제공하는 함수

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



난수 생성 라이브러리 함수

- rand()
 - 난수를 생성하는 함수
 - 0부터 RAND_MAX (32767)까지의 난수를 생성



난수 생성 라이브러리 함수



// 난수 생성 프로그램

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

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

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

// n개의 난수를 화면에 출력한다.

```
void get_random( int n )
```

```
{
```

```
    int i;
```

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

```
        printf( " %6d\n", rand() );
```

```
}
```

```
int main( void )
```

```
{
```

// 일반적으로 난수 발생기의 시드(seed)를 현재 시간으로 설정한다.

// 현재 시간은 수행할 때마다 달라지기 때문이다.

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

```
get_random( 10 );
```

```
return 0;
```

```
}
```



21783

14153

4693

13117

21900

19957

15212

20710

4357

16495



함수 사용의 장점

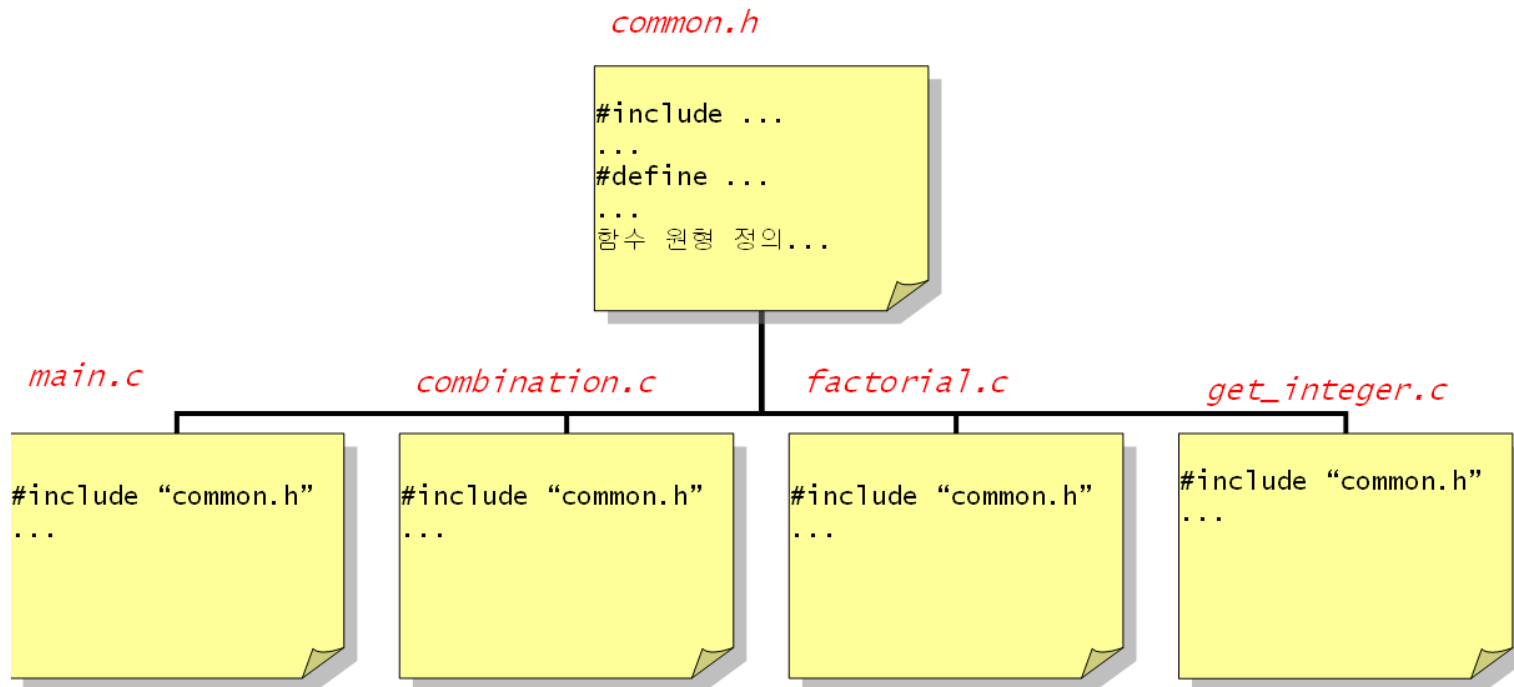


- 소스 코드의 중복 제거
 - 한번 만들어진 함수를 여러 번 호출하여 사용
- 한번 작성된 함수를 다른 프로그램에서 재사용
- 복잡한 문제를 단순한 부분으로 분해하여 해결



다중 소스 프로그램

- 함수 원형 정의는 헤더 파일에 들어 있고 여러 파일에서 헤더 파일을 포함



다중 소스 프로그램 예제

common.h



```
// 헤더 파일
#include <stdio.h>

#define MAX_INPUT 30

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

main.c



```
// 수학적 조합값을 구하는 예제
#include "common.h"
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;
}
```



다중 소스 프로그램 예제

combination.c



// 수학적 조합값을 계산

#include "common.h"

int combination(int n, int r)

{

 return (factorial(n)/(factorial(r) * factorial(n-r)));

}

factorial.c



// 팩토리얼 계산

#include "common.h"

int factorial(int n)

{

 int i;

 long result = 1;

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

 result *= i; // result = result * i

 return result;

}



다중 소스 프로그램 예제



get_input.c

// 사용자로부터 정수를 입력받는 함수 정의
`#include "common.h"`

```
int get_integer(void)
{
    int n;

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

    return n;
}
```



방향키 이동하기



● 커서 이동하기

```
#include<stdio.h>
#include<windows.h>
void gotoxy(int x, int y)
{
    COORD Cur = {x,y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Cur);
}
int main()
{
    gotoxy(30,11);
    printf("Hello! C");

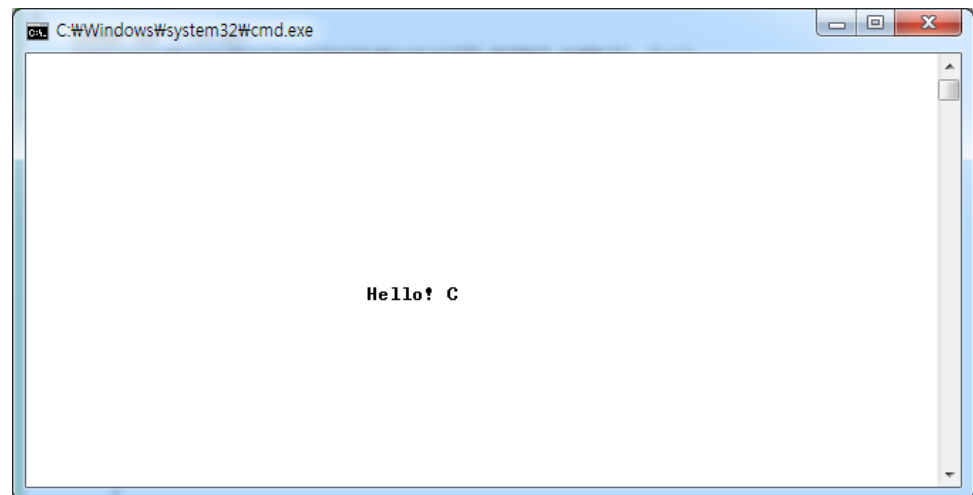
    return 0;
}
```

(0,0)

(79,0)

(0,24)

(79,24)



방향키 이동하기

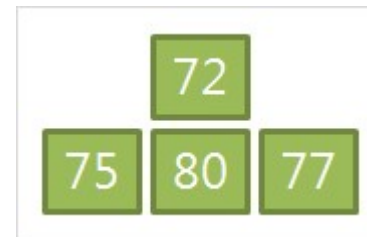


● getch()

- 키보드로 입력한 한 문자를 읽은 후, 해당 문자 반환
 - getchar()과 유사
- 사용자가 입력한 문자가 화면에 표시되지 않음
- `#include <conio.h>`

● 확장키 사용

- 아스키 코드에 등록되지 않은 키
 - function key (F1, F2 등), 방향 키, insert 키, home 키 등
- 방향키 = 224 + 해당 키 값(↑:72, ↓:80, ←: 75, →:77)
- 방향키 입력
 - `int ch;`
 - `ch = getch();`



```

#include<stdio.h>
#include<windows.h>
#include<conio.h>
void gotoxy(int x, int y)
{
    COORD Cur = {x,y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Cur);
}
int main()
{
    int x=40,y=12,ch;
    while(1)
    {
        gotoxy(x,y);
        printf("*");
        ch = getch();
        if(ch == 224)
            ch=getch();

        switch(ch) {
        case 72:
            y--;
            break;

        case 80:
            y++;
            break;

        case 75:
            x--;
            break;

        case 77:
            x++;
            break;

        }

    }
}

```

