프로그래밍연습 Lab 6

함수

[TA] 강성민, 김기현, 최석원, 최지은, 표지원
Department of Computer Science and Engineering
Seoul National University, Korea
2022/10/19

이번 장에서 학습할 내용

- 모듈
- 함수의 개념, 필요성
- 함수의 구조
- 함수의 호출과 반환
- 여러 가지 함수
- 함수를 사용하는 이유

1. 모듈

- **모듈(module)**
  - 독립되어 있는 프로그램의 일부분

- **모듈러 프로그래밍**
  - 모듈 개념을 사용하는 프로그래밍 기법

- **모듈러 프로그래밍의 장점**
  - 각 모듈들은 독자적으로 개발 가능
  - 다른 모듈과 독립적으로 변경가능
  - 유지보수가 용이
  - 모듈의 재사용 가능

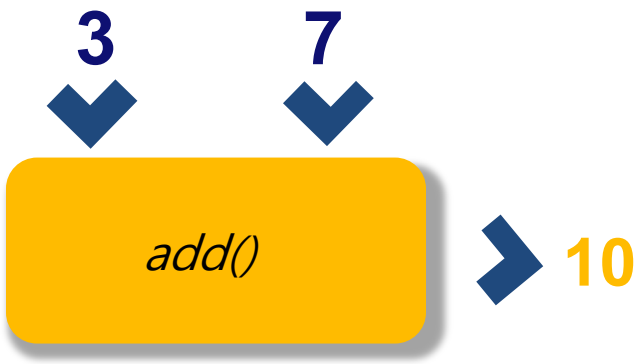
- C에서는 **모듈==함수**

2. 함수의 개념 및 필요성

- **함수(function)**
  - 특정한 작업을 수행하는 독립적인 부분

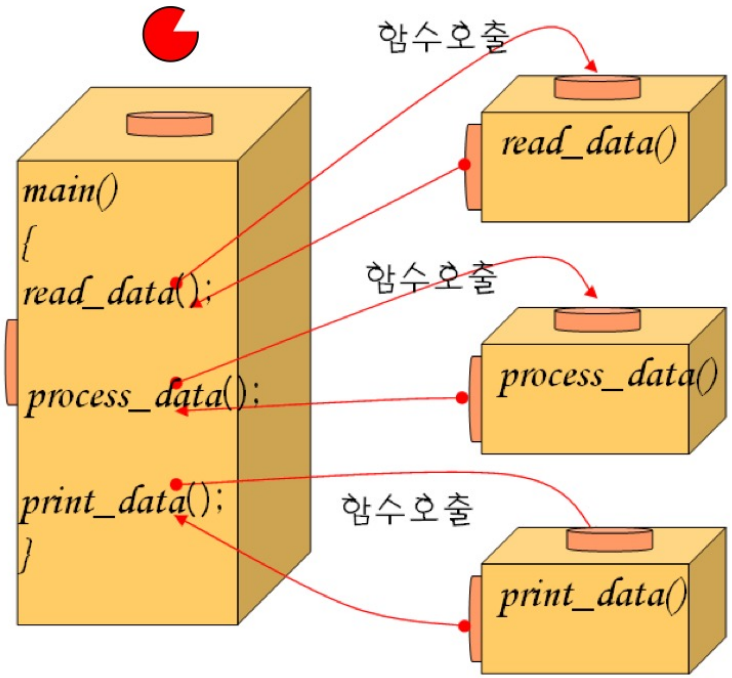
- **함수 호출(function call)**
  - 함수를 호출하여 사용하는 것.

- **함수는 이름을 가지며, 입력을 받아 특정 작업을 실행하고 결과를 반환함**



2. 함수의 개념 및 필요성

- **함수의 연결**
  - 프로그램은 **여러 개의 함수들로** 이루어짐
  - 함수 **호출**을 통하여 서로 연결됨
  - 제일 먼저 호출되는 함수는 **main()**



2. 함수의 개념 및 필요성

- **함수의 종류**
  - 사용자 정의 함수
    - ex) add_two_num(int a, int b) { … }
  - 라이브러리 함수
    - ex) printf()
    - 컴파일러에서 지원되는 함수이다.

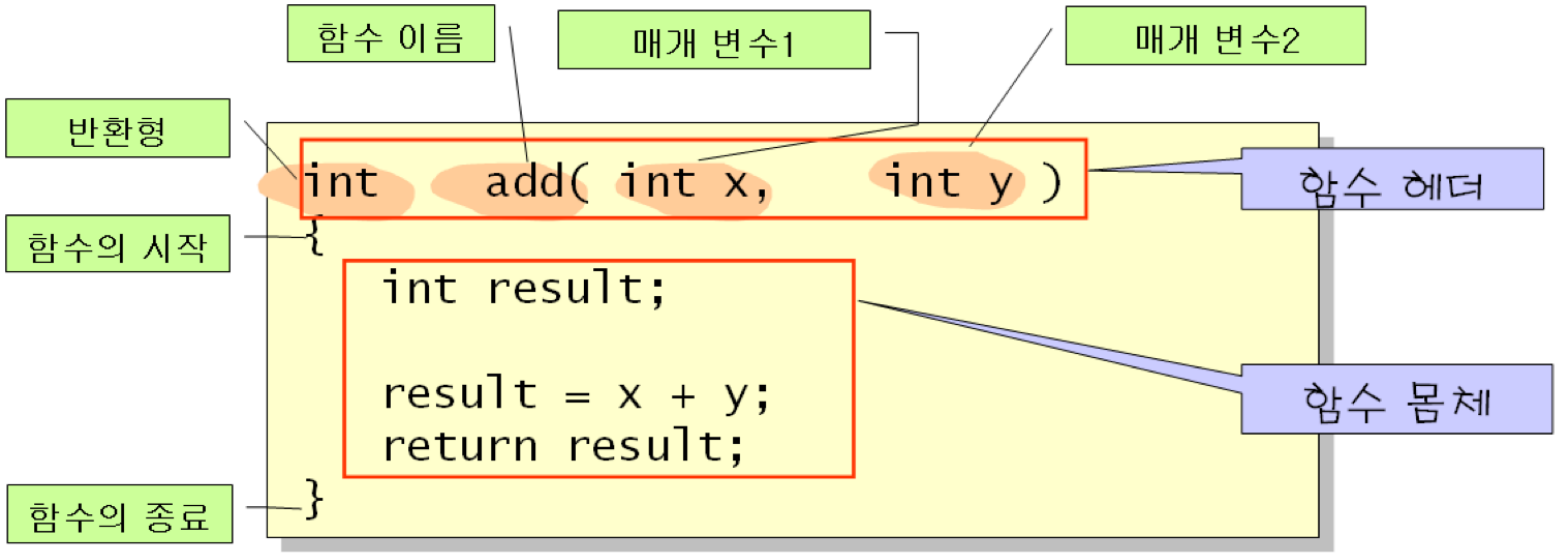
2. 함수의 개념 및 필요성

- 코드가 중복되는 것을 막을 수 있음
- 한 번 작성된 함수는 **재사용** 가능함
- 전체 프로그램을 모듈로 나눌 수 있음
  - 개발 과정이 쉬워짐
  - 체계적
  - 쉬운 유지보수

```c
#include <stdio.h>
void print_star() {
    int i;
    for(i=0;i<10;i++)
        printf("* \n");
}
int main(void) {
    print_star();
    print_star();

    return 0;
}
```

3. 함수의 구조

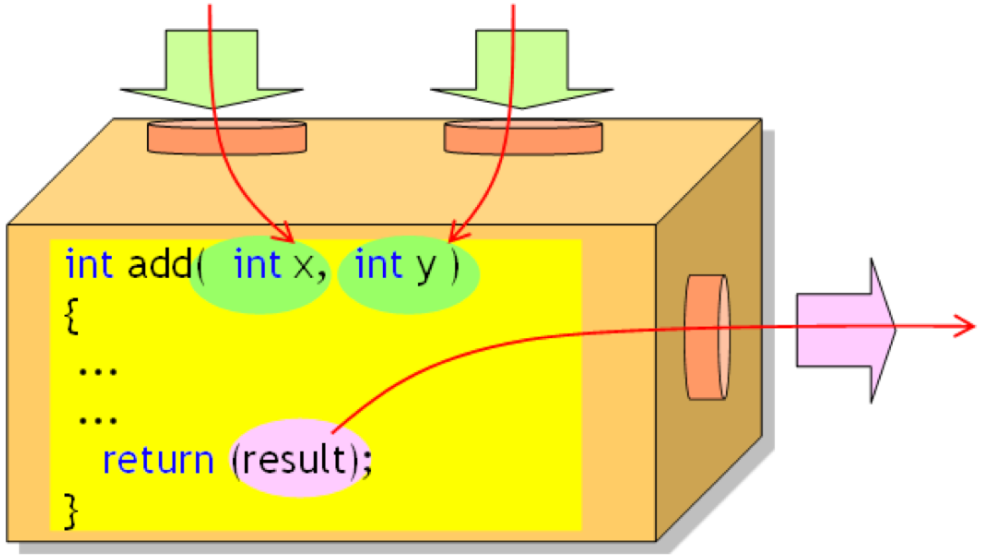
- **함수의 구조**
  - 반환형(return type)
  - 함수 헤더(function header)
  - 함수 몸체(function body)



3. 함수의 구조

- 반환형

```
int square()                    //int형의 값을 반환함
double compute_average()//double 형의 값을 반환함
void set_cursor_type()   //반환값이 없는 함수
```



3. 함수의 구조

- **매개변수**
  - 외부에서 전달되는 데이터가 저장되는 변수

```
int square(int n)                                  //정수를 제곱
double compute_average(double x, double y )        //평균 계산
void get_cursor_type(void)                         //커서의 타입 반환
```

예제 1.정수의 제곱값 계산하기

- **함수 작성 시 고려사항**
  - 함수를 프로그램을 이루는 부품이라고 생각하자.
  - 입력을 받아 작업한 후 결과를 생성해 보자.
- **Given**
  - 반환값: int
  - 함수 이름: square
  - 매개변수: int n

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

예제 2.두 정수 중 큰 수를 계산하는 함수

- **Given**
  - 반환값: int
  - 함수 이름: get_max
  - 매개변수: int x, int y

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

예제 3.팩토리얼 계산

- **Given**
  - 반환값: int
  - 함수 이름: factorial
  - 매개변수: int n
  - => int factorial(int n) { … } 활용
- **Hint**
  - 출력할 변수 result (자료형 long)를 1로 초기화
  - for문을 이용하여

Do it yourself!

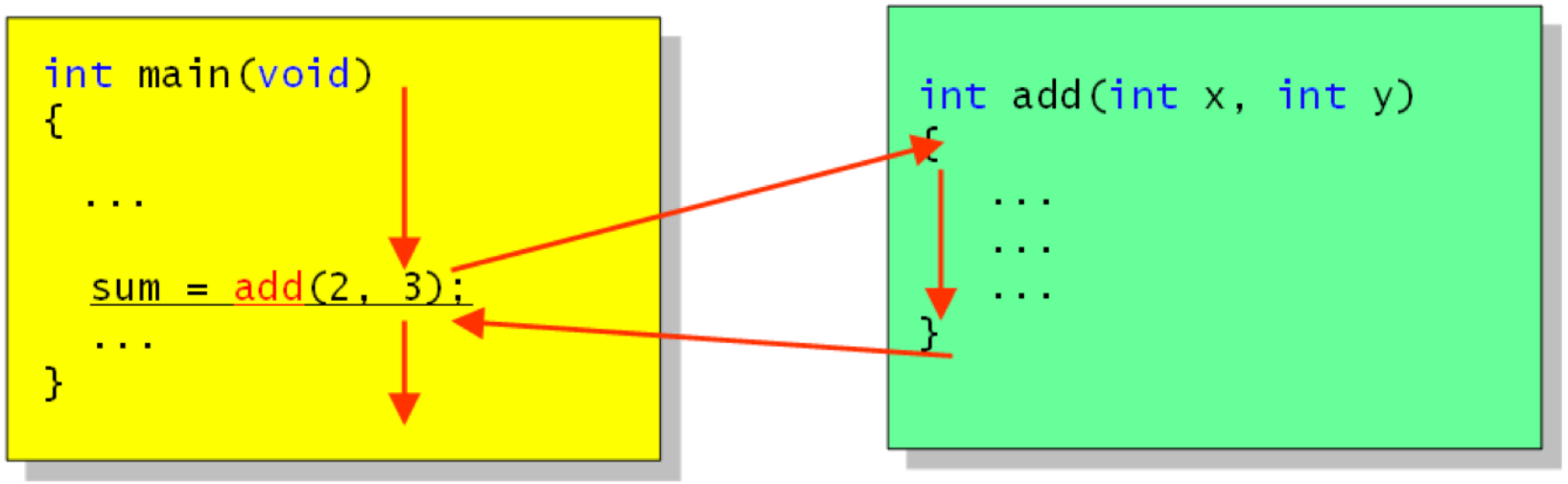
예제 3.팩토리얼 계산

- **Given**
  - 반환값: int
  - 함수 이름: factorial
  - 매개변수: int n

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

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

4. 함수 호출과 반환

- **함수 호출(function call)**
  - 함수를 사용하기 위해 함수의 이름을 적어주는 것
  - 함수 안의 문장들이 순차적으로 실행
  - 문장의 실행이 끝나면 호출한 위치로 되돌아감
  - 결과값 전달 가능



4. 함수 호출과 반환

- **함수 원형(function prototyping)**
  - 컴파일러에게 함수에 대해 미리 알리는 것.
  - 형식: **반환형 함수이름(매개변수 1, 매개변수 2,...);**

```
int compute_sum(int n);
int main(void)
{
        int sum;
        sum = compute_sum(100);
        printf("sum=%d \n", sum);
}
int compute_sum(int n)
{
        int i;
        int result = 0;
        for(i = 1; i <= n; i++)
                result += i;
        return result;
}
```

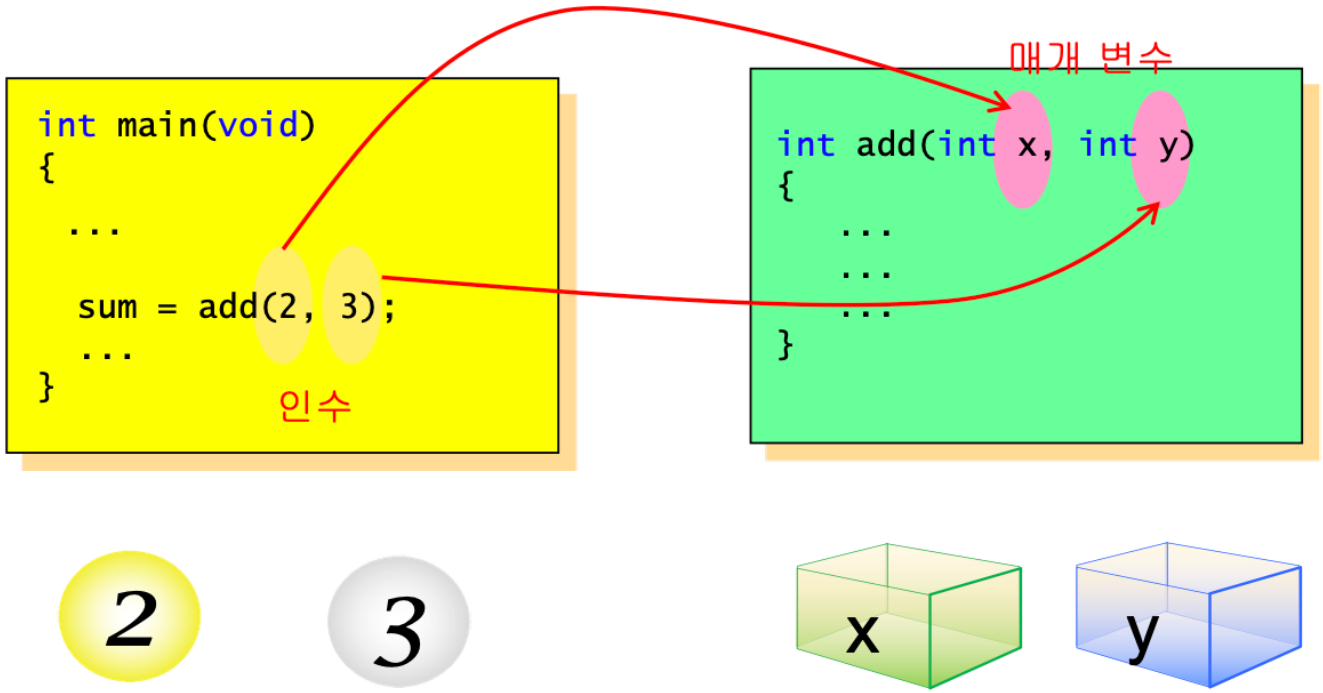
- (예)
- int get_integer(void);
- int combination(int n, int r);

- (예)
- int get_integer(void);
- int combination(int, int);

자료형만 적어주어도 됨!

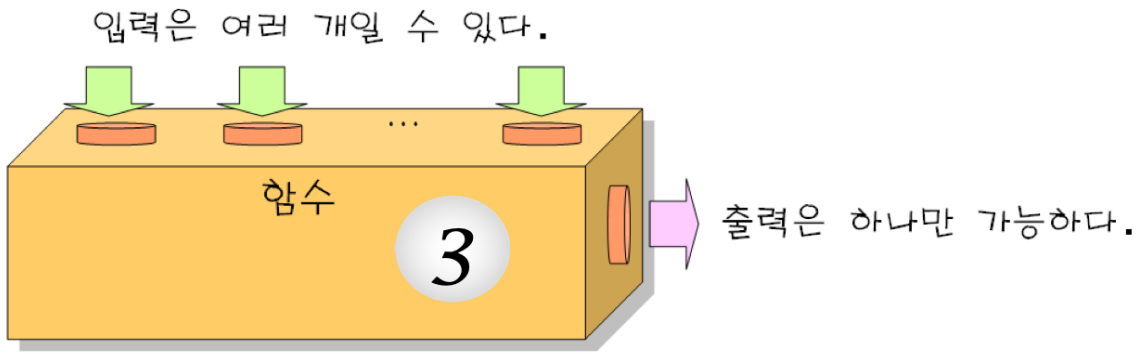
4. 함수 호출과 반환

- **인수(argument)**
  - 실인수, 실매개 변수라고도 함
- **매개 변수(parameter)**
  - 형식 인수, 형식 매개 변수라고도 함



4. 함수 호출과 반환

- **반환값(return value)**
  - 호출된 함수가 호출한 곳으로 작업의 결과값을 전달하는 것
- **인수는 여러 개가 가능하나 반환값은 하나만 가능**
  - 다중 리턴을 지원하는 Python과 다름




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

4. 함수 호출과 반환

- **함수 원형과 헤더파일**
  - 보통 헤더파일에 함수 원형이 선언되어 있음.
  - 사용자 정의 함수는 맨 위에 선언

```c
#include <stdio.h>
int get_max(int x, int y);

int main(void) {
    int a, b;
    printf("두 개의 정수 입력: ");
    scanf("%d %d", &a, &b);
    printf("두 수 중 큰 수는 %d \n", get_max(a,b));
    return 0;
}
int get_max(int x, int y) {
    if(x>y) return(x);
    else return(y);
}
```

```
/***
*stdio.h - definitions/declarations for
standard I/O routines
*
*
****/

...
_CRTIMP int __cdecl printf(const char
*, ...);
...
_CRTIMP int __cdecl scanf(const char
*, ...);
...
```

stdio.h

4. 함수 호출과 반환

- 예제1(정수의 제곱을 계산하는 함수) 전체코드

```c
#include <stdio.h>
int square(int n);

int main(void) {
    int result;
    result = square(5);
    printf("%d ", result);
}

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

4. 함수 호출과 반환

- 예제2(두 정수를 입력받아 큰 수를 찾는 함수) 전체코드를 작성해보자.

```
두 개의 정수 입력: 2 3
두 수 중 큰 수는 3
```

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

4. 함수 호출과 반환

- 예제2(두 정수를 입력받아 큰 수를 찾는 함수) 전체코드

```c
#include <stdio.h>
int get_max(int x, int y);

int main(void) {
    int a, b;
    printf("두 개의 정수 입력: ");
    scanf("%d %d", &a, &b);
    printf("두 수 중 큰 수는 %d \n", get_max(a,b));
    return 0;
}
int get_max(int x, int y) {
    if(x>y) return(x);
    else return(y);
}
```

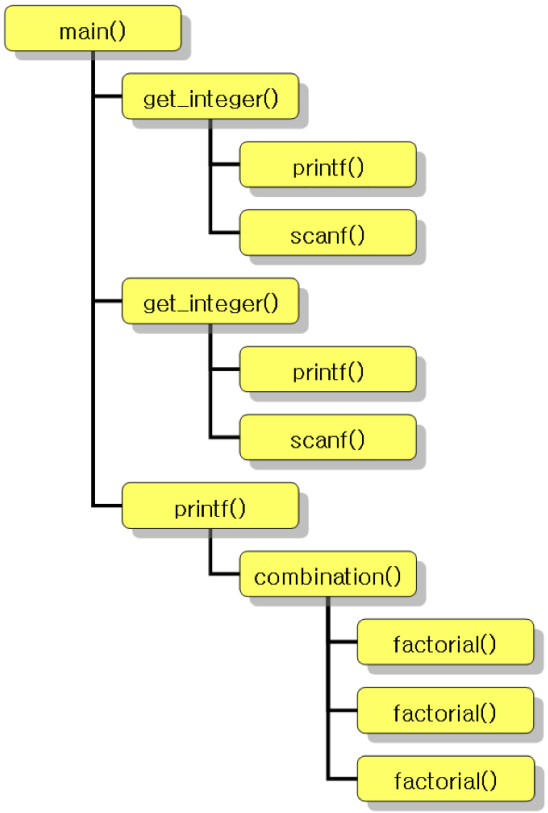
4. 함수 호출과 반환

- **예제 3활용- 조합(combination) 계산 프로그램**
  - 예제 3에서 만들었던 **factorial()** 함수와, 사용자로부터 값을 입력 받아 반환하는 **get_integer()** 함수를 호출하여 조합을 계산해 보자.

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

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

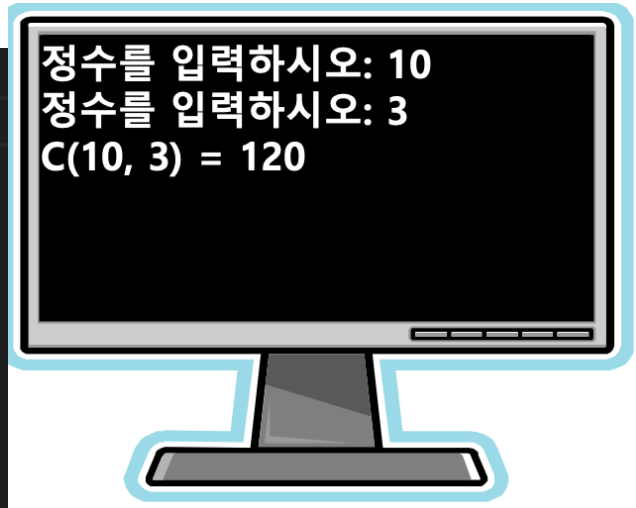
```c
int get_integer(void) {
    int n;
    printf("정수를 입력하시오: ");
    scanf("%d", &n);
    return n;
}
```



4. 함수 호출과 반환

- 조합 계산 프로그램 전체 코드

```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)));
}
```



4. 함수 호출과 반환

- 조합 계산 프로그램 전체 코드

```
21  int get_integer(void) {
22      int n;
23      printf("정수를 입력하시오: ");
24      scanf("%d", &n);
25      return n;
26  }
27
28  int factorial(int n) {
29      int i;
30      long result = 1;
31
32      for(i=1; i<=n; i++)
33          result *= i;
34      return result;
35  }
```

5. 여러 가지 함수

- **라이브러리 함수**
  - 컴파일러에서 제공하는 함수
  - 예시
    - 표준 입출력
    - 문자열 처리
    - 오류 처리
    - 수학 연산
    - 시간 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_MAX 까지의 난수를 생성

5. 여러 가지 함수

- 유틸리티 함수

함수	설명
exit(int status)	exit()를 호출하면 호출 프로세스를 종료시킨다.
int system(const char *command)	system()은 문자열 인수를 운영 체제의 명령어 셸에게 전달하여서 실행시키는 함수이다.

```
#include <stdlib.h>
#include <stdio.h>
int main( void )
{
        system("dir");
        printf("아무 키나 치세요\n");
        getch();
        system("cls");
        return 0;
}
```

```
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 507A-3B27
c:₩source₩chapter02₩hello₩hello 디렉터리
2011-11-28 오후 04:32 <DIR> .
2011-11-28 오후 04:32 <DIR> ..
2011-11-16 오전 11:01 20 binary.bin
...
4개 파일 5,296 바이트
3개 디렉터리 69,220,450,304 바이트 남음
아무 키나 치세요
```

5. 여러 가지 함수

- 표준 라이브러리인 math.h 안에 정의된 함수들이다.
  - #include <math.h>

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)	$\sqrt{x}$

5. 여러 가지 함수

- 표준 라이브러리인 math.h 안에 정의된 함수들이다.

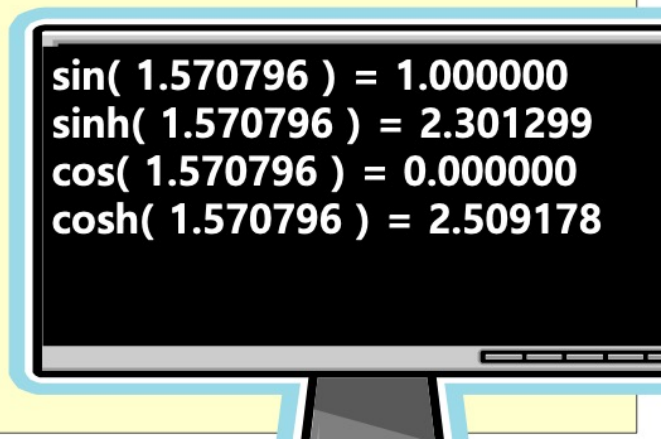
- abs(int x), fabs(double x)
  - abs(-9) // 9를 반환
  - fabs(-3.67) // 3.67을 반환
- pow(double x, double y)
  - 인수 x의 y-거듭제곱인 x^y 을 계산한다.
  - pow(2.0, 3.0); // 8.0을 반환
- sqrt(double x)
  - 주어진 수의 제곱근을 구한다. 만약에 음수가 입력되면 오류가 발생한다.
  - sqrt(9.0); // 3.0을 반환
- ceil(double x)
  - ceil은 x보다 작지 않은 가장 작은 정수를 반환
  - ceil(-2.9); // -2.0을 반환
  - ceil(2.9); // 3.0을 반환
- floor(double x)
  - floor()는 x보다 크지 않은 가장 큰 정수를 반환한다.
  - floor(-2.9); // -3.0을 반환
  - floor(2.9); // 2.0을 반환

5. 여러 가지 함수

- math.h 활용 예시

```
// 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 = sinh( x );
        printf( "sinh( %f ) = %f\n",x, y );
        y = cos( x );
        printf( "cos( %f ) = %f\n", x, y );
        y = cosh( x );
        printf( "cosh( %f ) = %f\n",x, y );
}
```

여러 수학 함수들을 포함하는 표준 라이브러리



예제4. 로또 번호 생성하기(randomnum.c)

- 1~45 사이의 랜덤한 로또 번호 6개 만들어보자.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
        int i;
        for(i = 0; i < 6; i++)
                printf("%d ", rand());

        return 0;
}
```

- 실행 시 0~32767 사이의 정수로 출력됨.

"범위를 지정하려면?"
-> 나머지 연산자를 이용

- **printf("%d ", 1+(rand()%45));** 로 바꾸어 실행

예제4. 로또 번호 생성하기(randomnum.c)

- **실행 할 때마다 같은 결과가 출력됨.**
  - 매번 같은 seed값을 두고 계속하여 난수를 생성하기 때문.

seed

"매번 다른 결과를 얻고싶다면?"

예제4. 로또 번호 생성하기(randomnum.c)

- **실행 할 때마다 같은 결과가 출력됨.**
  - 매번 같은 seed값을 두고 계속하여 난수를 생성하기 때문.
- 랜덤함수에 seed값을 부여하는 **srand 함수** 사용
  - 숫자를 넣어도 되지만, 주로 현재의 시각을 seed로 사용함.

```
#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.

예제5. 자동차 게임(car_game.c)

- 난수를 이용하여서 자동차 게임을 작성
- 사용자가 키를 누를 때마다 1초씩 주행하도록 하자.
- 주행 거리는 난수로 결정된다. (rand, srand 사용)
- **HINT:** main 함수에서 disp_car()를 호출하여 인수를 전달하여 주자.

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

Algorithm

1. 난수 발생기를 초기화한다
2. for(i=0; i<주행시간; i++)

    난수를 발생하여서 자동차1의 주행거리에 누적한다.

    난수를 발생하여서 자동차2의 주행거리에 누적한다.

    disp_car()를 호출하여 자동차1의 distance 만큼 *를 출력한다.

    disp_car()를 호출하여 자동차2의 distance 만큼 *를 출력한다.

예제5. 자동차 게임(car_game.c)

```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;
}
```

```c
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");
}
```

예제6. 한수(hansoo.c)

- **문제**
  - 어떤 양의 정수 X의 각 자리가 **등차수열**을 이룬다면, 그 수를 **한수**라고 한다. 입력 n을 받았을 때, 1 이상 n이하의 한수의 개수를 출력하는 프로그램을 작성하시오.
    - 등차수열: 연속된 두 개의 수의 차이가 일정한 수열
    - **1000보다 작은 자연수 n**에 대해서만 고려
    - 두 자리 수 까지는 모두 한수로 취급

1	➔ 비교할 다른 자리 없음	➔ 1개 (한자리 수는 모두 한수)
52	➔ 비교할 다른 자리 없음	➔ 52개 (두자리 수는 모두 한수)
123	➔ (3 - 2) == (2 - 1) 한수 성립	➔ 101개 (100이후 한수 == 111, 123)
112	➔ (2 - 1) != (1 - 1) 한수 성립X	➔ 100개 (100이후 한수 == 111)

예제6. 한수(hansoo.c)

- 정답 코드

```c
#include <stdio.h>
int count_hansoo(int n);

int main(void) {
    int input, result;
    printf("원하는 수를 입력하시오 N: ");
    scanf("%d", &input);            //입력받기
    result = count_hansoo(input); //함수호출
    printf("%d \n", result);        //결과출력
}
```

```c
int count_hansoo(int n) {
    int cnt, i, one, ten, hund;
    cnt=0;

    if(n < 100)
        return n;
    else{
        for(i=100;i<=n;i++) {
            one = (i%100)%100;    //  1의 자리의 수
            ten = (i%100)/10;     // 10의 자리의 수
            hund = (i%100);       //100의 자리의 수
            if ((ten-one)==(hund-ten)){
                cnt+=1;
            }
        }
        return (99+cnt);
    }

}
```

THANK YOU!