



## [11주차 2강] 함수(3)



# 학습 내용

## 8.3 함수 호출과 반환 (연습)



# 학습 목표

8.3 함수 활용 방법을 연습한다.





## 8.3 함수 호출과 반환 (연습)





[연습문제1] N을 입력 받아, 1부터 N까지의 합을 구하고,  
합이 홀수이면 합\*2를 출력하고, 합이 짝수이면 합/2를 출력하기

✓ 함수 호출을 불필요하게 반복하지 않도록 유의!!

실행 결과

10 → N  
110



- ✓ sum() 함수가 불필요하게 여러 번 수행되는 코드
- ✓ 바람직하지 않은 코드

```
int sum(int);  
  
int sum(int n) {  
    int s = 0;  
    for (int i = 1; i <= n; ++i)  
        s += i;  
    return s;  
}
```

```
int main() {  
    int N;  
  
    scanf("%d", &N);  
  
    if (sum(N) % 2)  
        printf("%d\n", sum(N) * 2);  
    else  
        printf("%d\n", sum(N) / 2);  
  
    return 0;  
}
```



- ✓ 불필요하게 여러 번 수행되지 않도록 개선한 코드
- ✓ 함수 호출 결과를 변수에 저장하여 사용

```
int sum(int);

int sum(int n) {
    int s = 0;
    for (int i = 1; i <= n; ++i)
        s += i;
    return s;
}
```

```
int main() {
    int N, S;

    scanf("%d", &N);

    S = sum(N);
    if (S % 2)
        printf("%d\n", S * 2);
    else
        printf("%d\n", S / 2);

    return 0;
}
```



[연습문제2] 양의 정수 N과 M을 입력 받아,  
2~N 사이의 소수를 구하여 각 줄에 M개씩 출력하기

- ✓ 코드를 작성하기 전에 먼저 논리 구조를 구상하자.
- ✓ 어떻게 함수를 구성할 것인가?
- ✓ 일을 독립적인 단위로 나누기

실행 결과

50 5 → N, M

□ 2 3 5 7 11

□ 13 17 19 23 29

□ 31 37 41 43 47

- ✓ 두 가지 방식으로 생각해보자.





## [연습문제2]

## 방식 A

- ✓ 소수를 먼저 고려하고, 출력 모양은 그 다음에
- ✓ 2부터 50까지의 각 수에 대해 소수인지 검사하고,
- ✓ 소수 5개가 출력되면 '개행 문자' 출력

## 실행 결과

50 5 → N, M

☐ 2 3 5 7 11

☐ 13 17 19 23 29

☐ 31 37 41 43 47



```
void print_prime(int, int);  
int is_prime(int);
```

```
int main() {  
    int n, m;  
  
    scanf("%d%d", &n, &m);  
    print_prime(n, m);  
    return 0;  
}
```

```
int is_prime(int x) {  
    int i;  
    for (i = 2; i < x && x%i != 0; ++i)  
        ;  
    return (i == x);  
}
```

```
void print_prime(int n, int m) {  
    int x, cnt = 0;  
  
    for (x = 2; x <= n; ++x) {  
        if (is_prime(x)) {  
            printf(" %d", x);  
            cnt++;  
            if (cnt % m == 0)  
                printf("\n");  
        }  
    }  
}
```



## [연습문제2]

### 방식 B

- ✓ 출력 모양을 먼저 고려하고, 소수는 그 다음에
- ✓ 한 줄에 수를 5개씩 출력하되
- ✓ 출력되는 수는 현재 소수의 다음 소수

### 실행 결과

50 5 → N, M

□ 2 3 5 7 11

□ 13 17 19 23 29

□ 31 37 41 43 47



```
void print_prime(int, int);  
int is_prime(int);  
int next_prime(int);
```

```
int main() {  
    ... // 방식 A와 동일  
}
```

```
int next_prime(int x) {  
    do {  
        ++x;  
    } while (!is_prime(x));  
    return x;  
}
```

```
int is_prime(int x) {  
    ... // 방식 A와 동일  
}
```

```
void print_prime(int n, int m) {  
    int i, x;  
  
    x = 2;  
    while (x <= n) {  
        for (i=0; i<m && x<=n; ++i) {  
            printf(" %d", x);  
            x = next_prime(x);  
        }  
        printf("\n");  
    }  
}
```



[연습문제3] N개의 문자를 하나씩 입력 받아,  
각 문자가 영어 소문자/대문자/숫자 인지 판별하고,  
영문자이면 추가로 자음/모음을 판별하기

✓ (참고) 문자 입력 시 사용되는 개행문자 처리에 유의

실행 결과

3 → N  
A  
대문자 모음  
b  
소문자 자음  
9  
숫자



```
int is_upper(char);
int is_lower(char);
int is_digit(char);
int is_vowel(char);
void print_vowcon(char);
void print_class(char);

int main() {
    int N;
    char ch, tmp;
    scanf("%d", &N);
    for (int i = 0; i < N; ++i) {
        scanf("%c%c", &tmp, &ch);
        print_class(ch);
    }
    return 0;
}
```

```
void print_class(char ch) {
    if (is_upper(ch)) {
        printf("대문자 ");
        print_vowcon(ch);
    }
    else if (is_lower(ch)) {
        printf("소문자 ");
        print_vowcon(ch);
    }
    else if (is_digit(ch))
        printf("숫자\n");
}

void print_vowcon(char ch) {
    if (is_vowel(ch))
        printf("모음\n");
    else
        printf("자음\n");
}
```



```
int is_upper(char ch) {
    return (ch >= 'A' && ch <= 'Z');
}
int is_lower(char ch) {
    return (ch >= 'a' && ch <= 'z');
}
int is_digit(char ch) {
    return (ch >= '0' && ch <= '9');
}
int is_vowel(char ch) {
    if (ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' ||
        ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U')
        return 1;
    return 0;
}
```

# 학습 정리

- 함수를 불필요하게 여러 번 수행하지 않도록 유의하자!
- 함수 구성은 일을 독립적인 단위로 어떻게 나누냐에 따라 달라짐
- 함수의 필요성 : 함수는 코드 작성의 효율성과 가독성을 높여줌