함수

20200517

함수의 기초

```c
#include <stdio.h>

int main(void)
{
    printf("Hello World! \n");
    return 0;
}
```

함수를 쓰는 이유??

만약, main 함수 내 코드가 엄청 길고,

불필요한 반복 작업들이 들어가 있다면??

	반환 값의 유무	
전달 인자의 유무	전달 인자 O 반환 값 O	전달 인자 O 반환 값 X
	전달 인자 X 반환 값 O	전달 인자 X 반환 값 X

	반환 값의 유무	
전달 인자의 유무	전달 인자 O 반환 값 O	전달 인자 O 반환 값 X
	전달 인자 X 반환 값 O	전달 인자 X 반환 값 X

```c
int Add(int a, int b) {

    int y = a + b;
    return y;
}
```

```c
#include <stdio.h>

int main(void)
{
    printf("Hello World! \n");
    return 0;
}
```

```c
int Add(int a, int b) {

    int y = a + b;
    return y;
}

int main(void) {
    int res;
    res = Add(3,4);
    printf("Add 함수를 통한 덧셈의 결과: %d", res);
    return 0;
}
```

	반환 값의 유무	
전달 인자의 유무	전달 인자 O 반환 값 O	전달 인자 O 반환 값 X
	전달 인자 X 반환 값 O	전달 인자 X 반환 값 X

"쉽게 말해서, 그냥 그 함수 안에서 끝낸다는 느낌"

Practice 1 #인자 전달과 반환 값

아래와 같이 출력 결과가 나오도록 코드를 구성하시오.

(단, main 함수에는 변수 선언 및 함수 선언만 허용함)

구성 : 메뉴 구성 / 변수 받기 / 결과 처리 / 결과 출력

```
출력 결과
두 개의 정수를 입력하시면 덧셈 결과가 출력됩니다.
자! 그럼 두 개의 정수를 입력하세요.
12 24
덧셈 결과 출력: 36
```

return의 의미

1. 값을 반환한다.
2. 함수를 빠져나간다.

```
int Add(int a, int b) {

    int y = a + b;
    return y;
}
```

```
void NoReturnType(int num) {

    if(num<0)
        return;
}
```

1

```c
#include <stdio.h>

int Add(int a, int b) {

    int y = a + b;
    return y;
}

int main(void) {
    int res;
    res = Add(3,4);
    printf("Add 함수를 통한 덧셈의 결과: %d", res);
    return 0;
}
```

선언과

2

```c
#include <stdio.h>

int main(void) {
    int res;
    res = Add(3,4);
    printf("Add 함수를 통한 덧셈의 결과: %d", res);
    return 0;
}

int Add(int a, int b) {

    int y = a + b;
    return y;
}
```

3

```c
#include <stdio.h>

int Add(int a, int b);

int main(void) {
    int res;
    res = Add(3,4);
    printf("Add 함수를 통한 덧셈의 결과: %d", res);
    return 0;
}

int Add(int a, int b) {

    int y = a + b;
    return y;
}
```

Summary

1. 함수는 크게 4가지!! (전달 인자 & 반환 값)
2. return의 또 다른 기능 -> 함수를 빠져나간다!!
3. 함수 선언 규칙 꼭 알기!!

변수에 대하여

C언어에서 변수가 메모리에 들어가는 과정

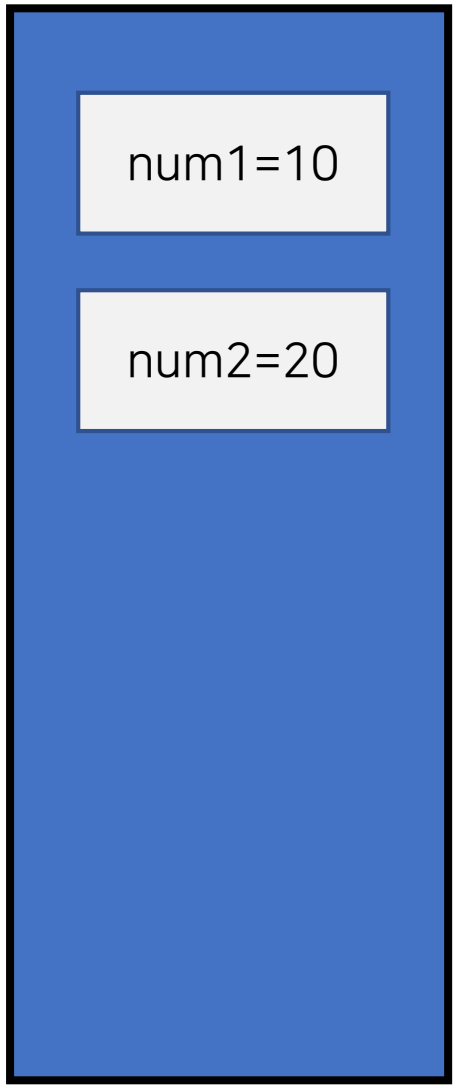
```c
#include <stdio.h>

int main(void){

    int num1 = 10;
    int num2 = 20;

    printf("합계: %d", num1+num2);

    return 0;
}
```



중괄호 내에 선언되는 변수는 모두

지역변수 이다.

지역 변수의 특징

1. 중괄호 내에 선언되는 변수는 모두 지역변수이다.
2. 지역변수는 해당 지역을 벗어나면 자동으로 소멸한다.
3. 지역변수는 선언된 지역 내에서만 유효하기 때문에, 선언된 지역이 다르면 이름이 같아도 문제가 되지 않는다. (다른 변수로 취급)

지역변수에 대한 이해

```c
#include <stdio.h>

int SimpleFuncOne(void) {
    int num = 10; // 이후부터 SimpleFuncOne의 num 유효
    num++;
    printf("SimpleFuncOne num: %d\n", num);
    return 0; // SimpleFuncOne의 num이 유효한 마지막 문장
}

int SimpleFuncTwo(void) {
    int num1 = 20; // 이후부터 num1 유효
    int num2 = 30; // 이후부터 num2 유효
    num1++;
    num2--;
    printf("num1 & num2: %d %d\n", num1, num2);
    return 0; // num1, num2가 유효한 마지막 문장
}

int main(void) {
    int num = 17; // 이후부터 main의 num 유효
    SimpleFuncOne();
    SimpleFuncTwo();
    printf("main num: %d\n", num);
    return 0; // main의 num이 유효한 마지막 문장
}
```

반복문에서의 지역변수도 똑같다

```c
#include <stdio.h>

int main(void) {
    int cnt;
    for (cnt = 0; cnt < 3; cnt++) {
        int num = 0;
        num++;
        printf("%d번째 반복, 지역변수 num은 %d.\n", cnt + 1, num);
    }
    if (cnt == 3) {
        int num = 7;
        num++;
        printf("if문 내에 존재하는 지역변수 num은 %d. \n", num);
    }
    return 0;
}
```

Practice 2 #지역변수 가리기

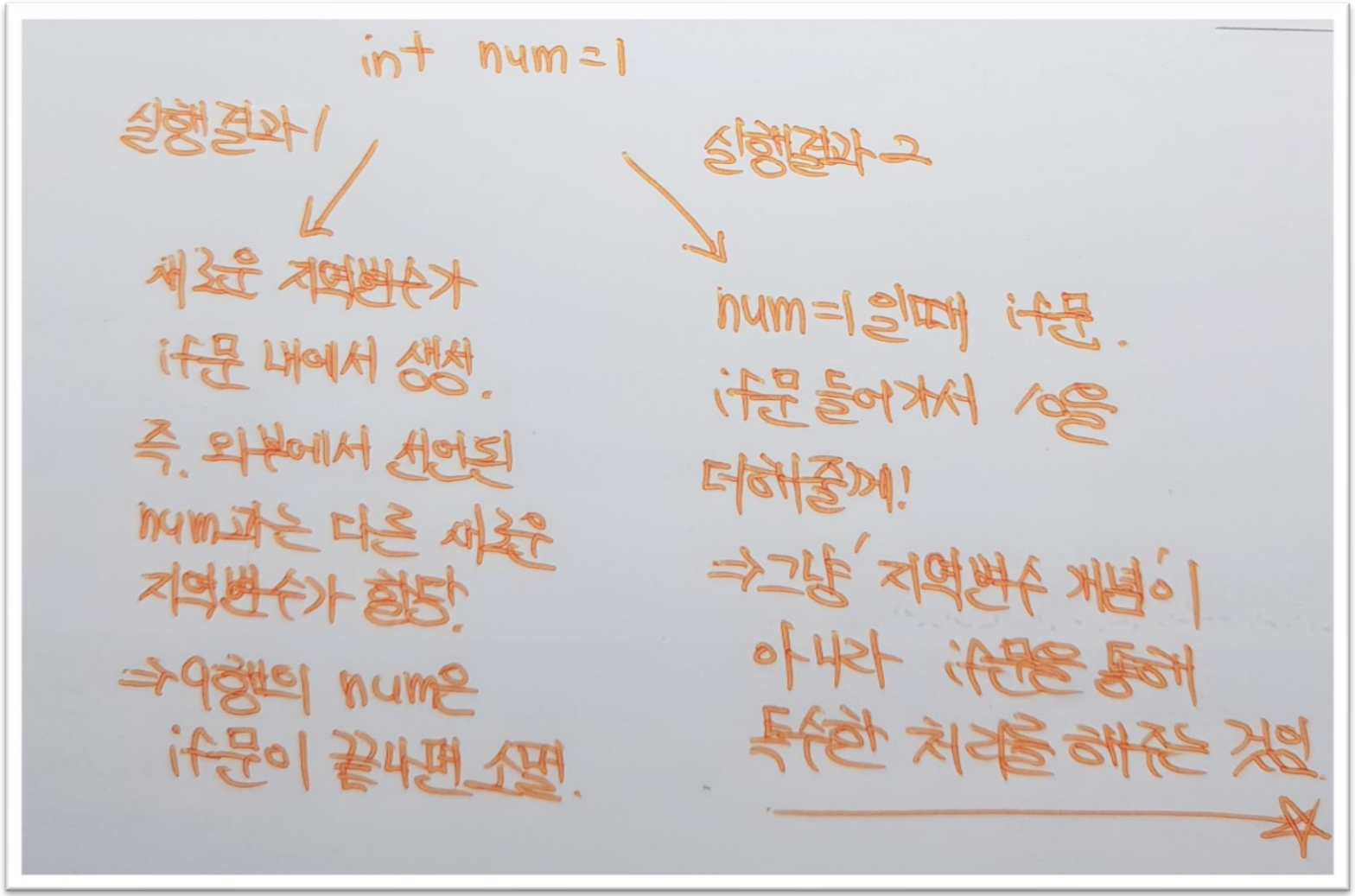
아래 코드의 결과를 분석하시오.

(추가로, int num = 7 이라는 코드를 주석처리 했을 때는 출력 결과가 어떻게 나오는지까지 분석하시오)

```c
#include <stdio.h>

int main(void) {
    int num = 1;
    if (num == 1) {
        int num = 7; // 이 행을 주석처리 하고 실행결과 확인하자!
        num += 10;
        printf("if문 내 지역변수 num: %d\n", num);
    }
    printf("main 함수 내 지역변수 num: %d\n", num);
    return 0;
}
```

Practice 2 분석 #지역변수 가리기



Summary

1. 함수는 크게 4가지!! (전달 인자 & 반환 값)
2. return의 또 다른 기능 -> 함수를 빠져나간다!!
3. 함수 선언 규칙 꼭 알기!!
4. 지역변수 특징 꼭 기억하기 (괄호 기준)

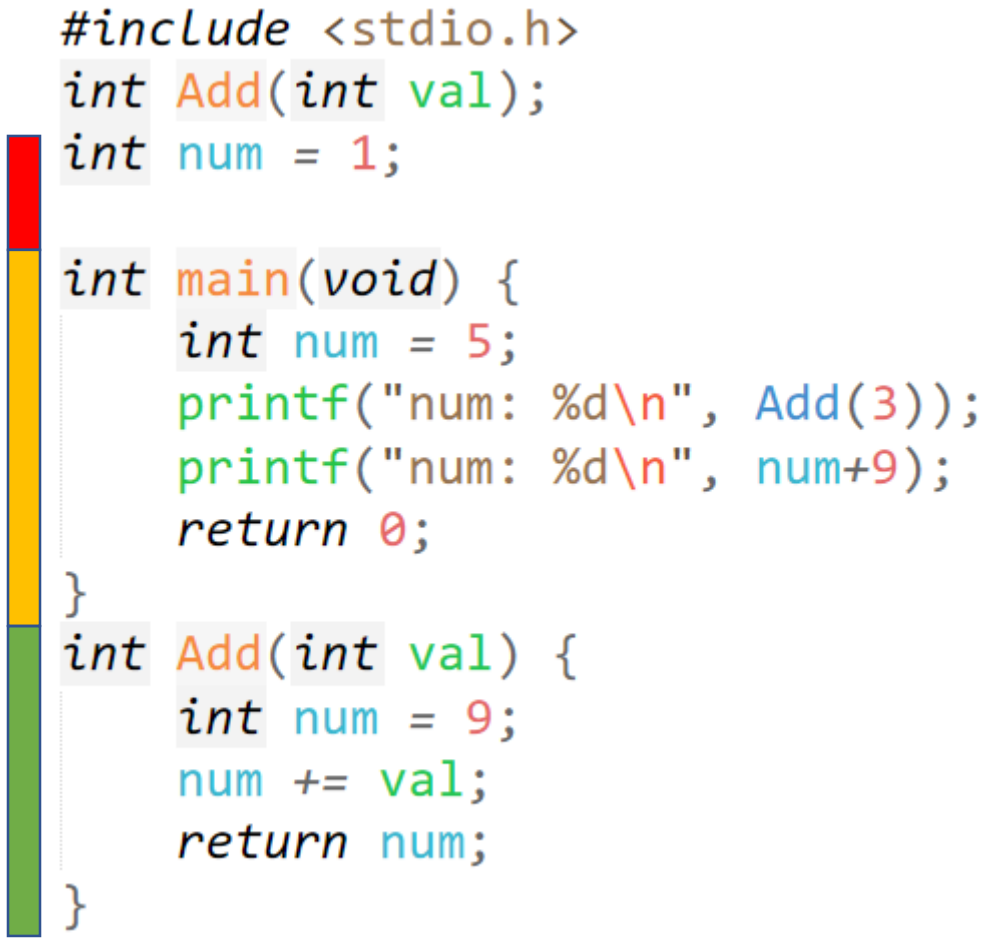
전역변수의 이해와 선언방법

```c
#include <stdio.h>
void Add(int val);
int num; // 전역변수는 기본 0으로 초기화됨

int main(void) {
    printf("num: %d\n", num);
    Add(3);
    printf("num: %d\n", num);
    num++; // 전역변수 num의 값 1 증가
    printf("num: %d\n", num);
    return 0;
}

void Add(int val) {
    num += val; // 전역변수 num의 값 val만큼 증가
}
```

전역변수의 이해와 선언방법



```c
#include <stdio.h>
int Add(int val);
int num = 1;

int main(void) {
    int num = 5;
    printf("num: %d\n", Add(3));
    printf("num: %d\n", num+9);
    return 0;
}
int Add(int val) {
    int num = 9;
    num += val;
    return num;
}
```

static 변수

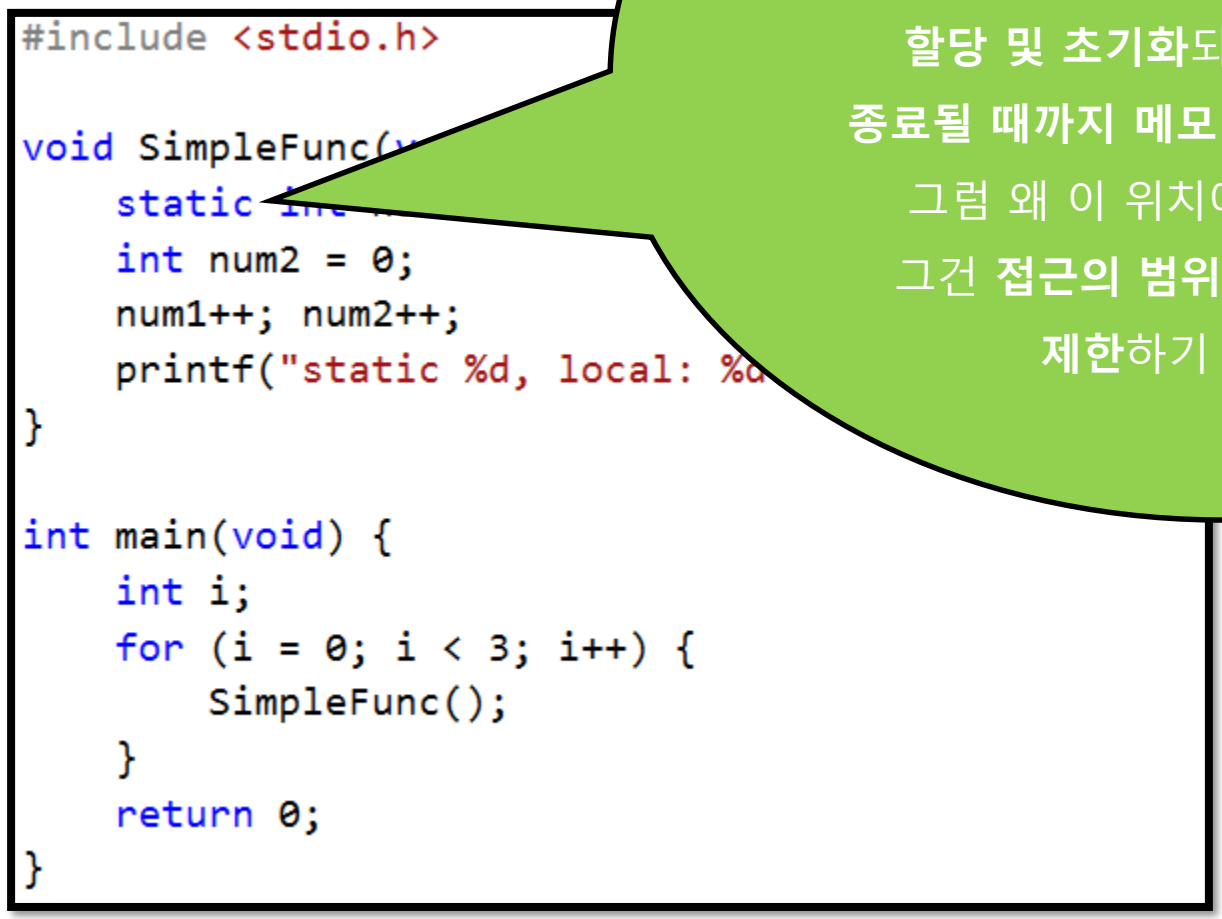
선언된 함수 내에서만 접근이 가능함
(지역변수 특성)
+
딱 1회 초기화되고 프로그램 종료 시까지
메모리 공간에 존재
(전역변수 특성)

```c
#include <stdio.h>

void SimpleFunc(void) {
    static int num1 = 0;
    int num2 = 0;
    num1++; num2++;
    printf("static %d, local: %d \n", num1, num2);
}

int main(void) {
    int i;
    for (i = 0; i < 3; i++) {
        SimpleFunc();
    }
    return 0;
}
```

static 변



난 사실 **전역변수**랑 성격이 같아.
초기화하지 않으면 **전역변수처럼**
0으로 초기화되고, **프로그램 시작과 동시에**
할당 및 초기화되어서 **프로그램이**
종료될 때까지 메모리 공간에 남아있지!
그럼 왜 이 위치에 선언되었냐고?
그건 **접근의 범위를 SimpleFunc로**
제한하기 위해서야!