함수

```
자료형 함수명(매개변수) {}
```

```
#include<stdio.h>
void add(int a, int b) {
    printf("add: %d\n", a + b);
}
int minus(int a, int b) {
    return a - b;
}
int mult(int a, int b) {
    a *= b;
    return a;
}
double div(int a, int b) {
    return (double)a / b;
}

int main()
{
    int a = 9, b = 5;
    add(a, b);
    printf("minus: %d\n", minus(a, b));
    printf("mult: %d\n", mult(a, b));
    printf("div: %lf\n", div(a, b));
    return 0;
}
```

output:

```
add: 14
minus: 4
mult: 45
div: 1.800000
```

void형 함수에는 반환값이 존재할 수 없으므로 return; 말고는 사용할 수 없다.

int, char, double등 다른 자료형은 반환값이 존재할 수 있으므로 return 1;과 같이 함수의 결과로 데이터를 반환할 수 있다.

Call by value

```
#include<stdio.h>
void add(int a, int b) {
    printf("add: %d\n", a + b);
}
int minus(int a, int b) {
    return a - b;
}
int mult(int a, int b) {
    a *= b;
    return a;
}
double div(int a, int b) {
    return (double)a / b;
}

int main()
{
    int a = 9, b = 5;
    add(a, b);
    printf("minus: %d\n", minus(a, b));
    printf("mult: %d\n", mult(a, b));
    printf("div: %lf\n", div(a, b));
    return 0;
}
```

output:

```
add: 14
minus: 4
mult: 45
div: 1.800000
```

방금 위에서 쓴 이 코드가 바로 Call by value 방식을 사용하는 함수 호출 방식이다.
자바같은 언어들을 비롯한 꽤 많은 언어들이 Call by value만을 사용하는 경우가 많다.
뒤에서 설명할 Call by reference 또는 Call by address 방식은 편리함과 동시에 보안상 취약해지기 때문이다.
Call by value는 오직 저장되어 있는 값만을 전달하기 때문에 함수가 끝난 이후 메인문에서는 어떠한 변화도 생기지 않는 특징이 존재한다.

Call by address

```
#include<stdio.h>
void add(int* a, int b) {
    *a += b;
    b += *a;
}
int main()
{
    int a = 5, b = 3;
    add(&a, b);
    printf("%d %d", a, b);
    return 0;
}
```

output:

```
8 3
```

Call by address는 매개 변수로 일반적인 값이 아닌 주소값을 넘겨주는 방식을 말한다.
주소값을 넘겨주기 때문에, 매개 변수로 포인터 변수를 사용해야 한다.
C++에 참조자(reference)가 표준에 추가되면서, Call by address는 Call by value와 동일하다는 시각이 존재한다.
그 이유는 Call by value와 Call by address 모두 결국 매개 변수에 할당하므로 변수를 하나 더 생성하기 때문이다.
Call by value와 달리, Call by address는 포인터 사용하므로 함수에서 바꾼 값이 메인문에서도 바뀐다는 특징이 존재한다.

Call by reference

```
#include<iostream>
using namespace std;
void add(int& a, int& b) {
    a += b;
    b += a;
}
int main()
{
    int a = 5, b = 3;
    add(a, b);
    printf("%d %d", a, b);
    return 0;
}
```

output:

```
8 11
```

*위 코드는 C++ 코드이므로 확장자를 무조건 .c가 아닌 .cpp로 실행해야 한다.
C++로 넘어오면서 참조자라는 개념이 새로 생겼다. 포인터는 주소값을 다른 변수에 저장시키는 것에 반면, 참조자는 주소 자체를 따로 메모리 할당 없이 가져오므로 조금 더 편리하고 안전하다는 특징이 존재한다.

*참조자와 포인터에 대한 더 깊은 내용은 이후 활동에 다룰 예정이다.