함수

1. 강의

▼ 함수의 정의

프로그램에서 자주 사용되는 코드 블록을 따로 한 번만 만들어 두고 필요할 때 마다 불러서 사용하는 기능

▼ 함수 정의 형식

```
리턴형 함수명 (데이터형 매개변수명 [, 데이터형 매개변수명, ...])
{
  함수가 처리할 내용;
}

예제> int GetSum (int num) {
        int i, sum;
        for (i=1, sum =0, ; i <=num; i++){
          sum += i;
        }
        return sum;
      }
```

▼ 함수의 리턴형

- 함수 리턴값의 형을 말한다. int, double, void . 등
- 예> retrun sum 에서 sum 이 int 형이면, 함수 헤더부에도 int로 적어야 한다.
- void : 리턴 값이 없음을 의미
- 리턴형이 생략되었을 경우, c 는 자동으로 int 형 리턴으로 간주 한다.

▼ 함수 이름

- 이름도 식별자 이므로 식별자 만드는 규칙에 따른다.
- 영문자, 숫자, 밑줄만을 사용
- 일반적으로 동사+목적어의 형식
- 어떤 기능인지 알 수 있는 이름

▼ 매개변수

- 함수의 기능을 수행하는데 필요한 값을 넘겨주기 위한 변수

- 함수 이름 다음에 ( ) 안에 데이터형과 이름을 지정.
- 매개변수가 여러개이면 " , " 로 구분
- 매개변수의 개수에는 제한이 없다.
- 예: double GetMax (double a, double b, double c)
- int Add(int a, b) → error
- 함수의 매개변수도 함수 안에서 선언된 변수로 사용 된다.

▼ 함수 헤더와 바디

- 헤더 : 자료형 이름(매개변수...) 를 적는 부분
- 바디 : 함수가 처리할 내용을 적는 부분
- 변수의 자료 형식 선언시에는 함수 바디의 시작 부분에 선언

▼ 함수의 예

```c
//리턴문이 없는 함수
//리턴문이 없으면 함수의 끝에 왔을때 함수가 호출 된 곳으로 리턴한다. (ex1)
void PrintHello(void){
  printf("Hello World\n");
}

//리턴문이 있지만 중간에 있어, return문 다음에는 실행되지 않는다. (ex2)
void PrintHello(void){
  printf("Hello World\n");
  return;
  printf("Good Bye\n");
}

//리턴값은 없고, 매개변수는 있는 경우 (ex3)
void PrintSum(int a, int b){
  printf("합계 : %d \n", a + b);
}

//리턴과 매개변수 모두 있는 경우 (ex4, ex4_scanf)
int GetFactorial(int num) {
  int i;
  int fact = 1;
  for (i=1; i<=num; i++)
    fact *= i;
  return fact;
}
```

▼ 함수의 호출

- 함수이름 (함수 인자)
- 함수이름 ( )
- 인자란 함수 호출시 함수에 직접 넘겨 주는 값을 말한다.
- 즉, 함수 호출에 쓰여 있을땐 인자, 함수정의에 쓰여 있을땐 매개변수로 구분된다. (ex5, ex5_total)

▼ 처리과정

메인 함수에서 함수 호출 → 정의된 함수로 인자 전달 → 함수의 코드 수행 → 메인의 함수를 호출한 곳으로 돌아옴 → 메인 함수에서의 다음 줄 수행 (예제, ex6)

▼ 함수 호출시 주의 사항

- 함수의 인자로 수식을 사용할 경우 수식을 계산한 후, 계산 결과를 함수에 전달한다.

  (ex6에서, 10 대신, 1+3, GetFactorial(3) 을 대신 대입해 보기)
- 함수를 호출할때 넘겨주는 인자의 개수와 매개변수의 개수는 같아야 한다. (ex7)
- 함수 호출시 이름이 틀리면 컴파일 에러 발생

▼ 함수의 선언

- 함수가 정의 되기 전에 호출 하면 경고나 에러 메세지를 발생시킨다. (ex8)
- 경고인 경우에 실행파일이 만들어져 프로그램을 수행시켜도 쓰레기 값이 들어가게 된다.
- 따라서 함수를 호출하기 전에 항상 선언 되어 있거나 정의 되어 있어야 한다.
- 정의하는 함수가 많을 경우 함수 호출보다 함수 선언이 우선되었는지를 따져봐야 하기 때문에 함수의 정의 순서를 정하는 것이 복잡해진다. → 함수 선언 사용
- 함수 선언 형식

```
리턴형 함수명 (데이터형 매개변수명 [...]);
예제>
  void PrintHello (void);
  void PrintSumAverage (int a, int b);
  int  GetFactorial (int);
```

▼ 지역변수

- 함수 내부에서 선언된 변수로, 함수 안에서만 사용 할 수 있다.
- 함수가 리턴 될때, 자동으로 없어진다.
- 하나의 함수 안에서 선언된 지역 변수는 다른 함수에서 사용 할 수 없다.
- 서로 다른 함수 에서 같은 이름을 사용하더라도 두 변수는 서로 다른 변수이다. (ex9)
- 블럭문 안에서 선언된 변수의 경우에는 블럭문 안에서만 유효하고, 같은 함수 여도 블럭문 밖에서는 유효하지 않다. (ex10)
- 같은 함수를 여러번 호출해도 지역변수는 매번 초기화 되었다가 처리된후 해제 된다. (ex11)
- 함수의 매개변수도 지역변수이다.
- 지역변수도 선언할 때 초기화 하지 않으면, 기본적으로 쓰레기 값을 가진다.

▼ 전역변수

- 메인 함수 밖에 선언 한다.
- 전역변수는 모든 함수에서 사용 가능 (ex12)
- 전역변수는 프로그램이 시작될때 한 번만 생성되고, 프로그램이 종료될 때 해제된다.
- 전역변수를 함수들 사이에 선언하면 컴파일 에러가 발생한다. → 전역변수는 메인함수 위에 선언!
- 전역변수는 초기화 하지 않았을 경우 자동으로 0 으로 초기화 된다. (ex13)

▼ 변수의 영역 규칙

- 같은 함수에서 같은 변수명에 대해 여러번 선언하면 컴파일 에러 발생
- 같은 함수에서 블럭을 지정해 블럭 범위가 다를 때는 같은 변수명을 여러번 선언 할 수 있다.
- 하나의 변수명에 블럭별로 선언이 여러번 이면 가장 가까운 변수가 사용 된다. (ex14)
- 지역변수와 전역변수의 이름이 같은 경우 지역변수가 먼저 사용 된다. → 전역변수 이름을 지을때, g_num 과 같은 형식으로 지정하자

▼ 함수의 인자 전달

- 값에 의한 전달 : 인자를 함수 정의에 있는 매개변수로 복사해서 전달.

- 포인터에 의한 전달 : 변수의 값이 아닌 주소를 전달하는 방식
- 포인터에 의한 전달이 필요한 경우(ex15)
- ex15의 예제에서 a, b 의 값은 x, y에 복사 되어 Swap 함수 안으로 전달 된다. x, y 값은 Swap 함수 안에서 변경 된다. 그러나 a, b 는 x, y와 서로 다른 변수이므로 x, y의 변화가 a, b에 영향을 끼치지 못한다.

2. 실습