4.10 재귀

개요

함수를 사용할 때 어디에서 호출했나요? main 안에서 프로그램을 작성하면서 필요한 순간에 호출하여 사용합니다. 그런데 main 역시 함수라는걸 기억하시나요? main이라는 함수 안에서 또 다른 함수를 사용한 것입니다. 이 사실을 알게 되었을 때, 우리는 **함수가 본인 스스로를 호출해서 사용할 수 있는지**에 대해 의문을 가질 수 있습니다. 이에 대한 대답은 할 수 있다 이며, 이러한 것을 재귀(Recursion)라고 부릅니다.

핵심개념

* 재귀
* 스택

재귀의 사용

시그마라는 것을 아시나요? 시그마는 주어진 수 n 부터 1까지의 연속된 수를 모두 합한 값입니다. 시그마를 지금까지 우리가 배운 개념을 이용하여 프로그래밍 해보면 〈코드 1〉과 같습니다. 3행부터 6행까지는 양의 정수가 아닐 경우에 무한반복을 피하기 위해서 작성된 부분입니다. 8행부터 11행까지가 실질적으로 함수가 구현된 부분입니다. sum 변수에 i를 n까지 1씩 늘려가며 더하면 시그마 값이 반환됩니다.

그렇다면 동일한 기능의 프로그램을 재귀를 이용하여 작성한다면 어떻게 될까요? 〈코드 2〉를 보면 for문이 없는 것을 확인할 수 있습니다. else 안에 작성된 부분이 이 함수의 핵심입니다. **반환값 안에 자기 자신(함수)가 호출**됩니다. 9행의 코드는 결과적으로 n + (n-1) + (n-2)+…+ 1 + 0이 되어 n부터 1까지의 합이 반환됩니다.

```
1  int sigma(int n)
2  {
3      if(n < 1)
4      {
5          retrun 0;
6      }
7      int sum = 0;
8      for(int i=1; i<=n; i++)
9      {
10         sum += i;
11     }
12     return sum;
13 }
```

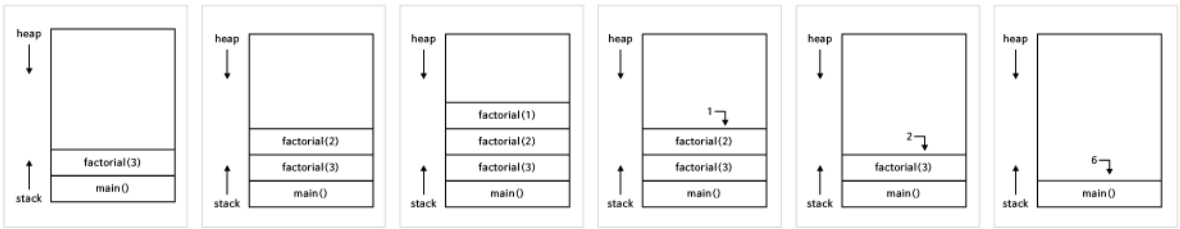
◀ 〈코드 1〉

```
1  int sigma(int n)
2  {
3      if(n<=0)
4      {
5          return 0;
6      }
7      else
8      {
9          return (n + sigma(n-1));
10     }
11 }
```

◀ 〈코드 2〉

스택

재귀 함수에서 동일한 함수를 계속해서 호출될 때마다 **함수를 위한 메모리가 계속해서 할당**됩니다. 함수가 호출될 때 마다 사용되는 메모리를 스택이라고 부릅니다. 컴퓨터가 일을 처리하는데 관리를 하는 역할인 운영체제는 함수를 실행할 수 있도록 일정량의 바이트를 주고, 그 공간에 함수의 변수나 다른 것들을 저장할 수 있도록 합니다. 그래서 재귀함수를 이용하다 보면 함수가 종료되지 않고, 함수가 계속해서 호출되는 경우가 발생하기도 합니다. 이 경우 스택 공간은 초과되고 프로그램 충돌이 발생됩니다. 그렇기 때문에 재귀를 사용할 때는 과도하게 스택 메모리가 사용되지 않도록 주의해야 합니다. 메모리 사용 문제 때문에 재귀는 매우 유의 해야 하지만, 특정 자료구조를 다룰 때 매우 유용하게 사용됩니다.



▲ 〈재귀함수 factorial이 스택메모리에 할당되고 종료되는 과정〉