1. 하노이탑 문제를 순환방법이 아니라 반복적인 방법으로 구현하는 방법?

인터넷을 검색해본 결과 반복을 이용한 코드는 크게 많이 없었습니다. 그 몇몇의 코드 또한 비슷한 방법을 사용하였으며 대부분 스택을 사용하였습니다. 제가 사실 구조체를 현재 공부 중에 있어 코드를 해석하는데 많은 어려움을 겪었지만 그 중 가장 효율적이라 생각했던 코드를 제 나름의 생각대로 해석해보겠습니다. 이 코드에서는 큰 무한 while문 안에 남은 원반 개수가 1이 될 때까지 작은 while문을 돌려 목적지, 임시공간, 출발지, 원반 순으로 스택에 쌓고 원반의 개수를 1씩 삭제하였습니다. 또한 임시 공간과 목적지의 역할을 바꿔가면서 임시 공간을 거쳐 목적지로 옮기는 방법을 이용했습니다. 이 작은 while문 아래에는 스택이 비어있지 않을 때 역순으로 빼주며 다시 원반의 개수를 하나 줄입니다. 그런 후 임시공간과 출발지의 역할을 교환합니다. 그리고 스택을 비우면 while문을 탈출하는 식으로 코드가 짜여 있습니다.

저는 이 활동을 하며 수업 중 듣는 것만으로도 한 번에 이해가 갔던 재귀적 방법에 비해 반복적 방법은 인터넷을 보고 계속 반복해서 생각을 해도 코드를 이해하기 쉽지 않았습니다. 뿐만 아니라 코드의 길이도 재귀적 방법이 반복적 방법에 비해 훨씬 짧고 효율적으로 나왔습니다. 이를 통해 보통의 경우에는 반복적 방법이 좋으나 어떠한 경우에는 재귀적 방법이 오히려 좋을 수 있겠다는 사실을 깨달았습니다.

1. 재귀적인 해결 방법(알고리즘)을 반복적인 해결 방법(알고리즘)으로 변환하는 일반적인 방법이 가능하고 있는가?

저는 최근 재귀적 방법과 반복적 방법의 코드를 모두 보여주는 수업을 들으며 재귀적 방법이란 자기 자신을 ‘다시’ 호출하여 문제를 해결하는 방법이고 반복적 방법이란 명령어를 이용하여 ‘다시’ 반복적으로 수행하는 방법이라는 것을 알게 되었습니다. 또한 두 방법 모두 ‘다시’라는 단어와 연관되어 있다는 것을 알게 되었습니다. 그렇기에 저는 둘 사이의 변환이 쉬울 줄 알았습니다. 하지만 인터넷에 검색해본 결과 쉽지 않은 경우도 있다는 것을 알게 되었습니다. 인터넷의 말을 빌리자면 순환에는 순환 호출이 끝에서 이루어지는 순환을 꼬리 순환, 그렇지 않은 것을 머리 순환이라고 합니다. 꼬리 순환의 경우, 반복문을 사용하여 쉽게 반복적 방법으로 쉽게 변환이 가능하다고 합니다. 반면, 머리 순환의 경우 혹은 하노이탑 문제와 같이 여러 부분에서 자기 자신을 호출하는 경우에는 쉽게 반복적인 코드로 바꿀 수 없다고 합니다. 다만 명시적인 스택을 만들어서 순환을 반복적 방법으로 바꿀 수 있다고 합니다.

저는 이 활동을 통해 인터넷에서 다양한 정보를 찾아보고 함으로써 재귀적 방법과 반복적 방법을 언제 사용해야 효율적인 프로그램을 짤 수 있는지 감이 오게 되었고 재귀적 방법으로 알고리즘을 작성할 때 가능하다면 꼬리 순환의 방법으로 작성해 재귀적 방법으로의 변환 난이도도 생각하는 것이 좋을 것 같다는 생각을 하게 되었습니다.