

객체지향프로그래밍

Assignment Report 1-2

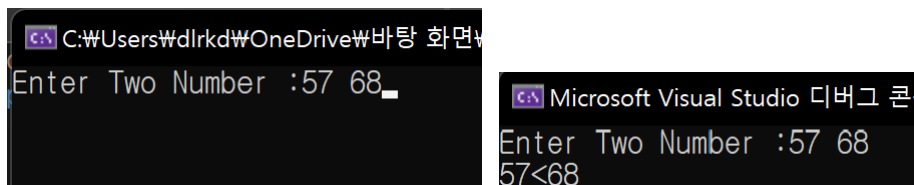
2019202050 이강현

문제 1

2개의 수를 입력 받아 두 수의 크기를 구분하는 프로그램 구현.

두 수를 입력 받고 $>$, $=$, $<$ 부호를 이용해 그 값들의 대소를 비교한다.

필요한 개념: 조건문을 사용하여 값의 대소비교를 할 수 있으며 콘솔을 이용한 입출력을 할 수 있어야 할 것 같다.



<입력>

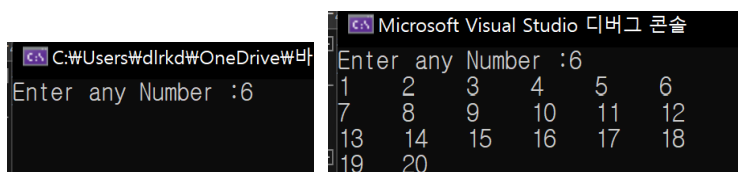
<출력>

고찰: 조건문을 활용하여 두수를 입력받고 비교하는 프로그램이다. 문제에선 단순히 크다,작다,같다만을 답으로 제시하여 간단하게 코드를 완성할 수 있었으나 조건문의 기본적인 활용방법과 사용예시를 알 수 있는 문제이다.

문제 2

숫자 $N(0 < N < 20)$ 을 입력 받아 1부터 20까지 N 번째 수에서 바꾸며 출력하는 프로그램을 구현.

필요한 개념: 반복문을 이용하여 20까지 출력할 수 있는 환경을 구현하고 조건문을 활용하여 개행시기를 정해야 한다. 조건은 나머지를 활용하자.



<입력>

<출력>

고찰: 코드를 작성하면서 문제에서 요구하는 결과값은 얻을 수 있었으나 값의 정렬이 제대로

되지 않아 곤란을 겪었다. 이와 같은 문제를 해결하기 위해 <iomanip>라이브러리 안의 setf와 setw를 활용하여 좌측정렬과 폭을 설정하도록 하여 깔끔한 인터페이스를 만들 수 있었다.

문제 3

3x3의 보드에서 Command를 입력 받아 Node가 움직이는 프로그램을 구현. 보드는 모두 '0'으로 채워져 있으며 Node는 'x'로 표기. 프로그램이 시작 시 Node는 좌상단 끝(0.0)에 위치하고 있으며 Command의 입력에 의해 움직임. 프로그램은 시작 시 보드 및 Node를 출력하고 Command를 입력받을 때마다 재출력하는 방식으로 동작. Command는 아래와 같으며 만약 Node가 명령어에 의해 보드의 범위를 넘어가려 하면 입력을 무시.

필요한 개념: 우선 입력값을 2번 이상 받아야 하고 받을 때마다 보드를 출력해야 하므로 조건문과 반복문을 사용해야함을 알 수 있다. 프로그램의 시작시 노드의 위치는 정해져 있으므로 그것을 기준으로 코드를 작성하고 입력값은 영문 소문자를 이용하여 받으므로 char 변수를 잘 활용하여 코드를 작성하면 좋을 듯 하다. Node가 보드를 벗어날 때는 입력을 무시하여 전에 있던 노드의 위치를 다시 재출력한다면 사용자가 제대로 된 값을 입력하는데 도움을 줄 수 있을 것이다.

<입력>

```
X          0          0
0          0          0
0          0          0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) :
```

```

Microsoft Visual Studio 디버그 콘솔
X      0      0
0      0      0
0      0      0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) : s
0      0      0
X      0      0
0      0      0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) : d
0      0      0
0      X      0
0      0      0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) : w
0      X      0
0      0      0
0      0      0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) : a
X      0      0
0      0      0
0      0      0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) : w
X      0      0
0      0      0
0      0      0
Enter Move Command(a: Left, d: Right, w: Up, s: Down, q: Finish) : q
C:\Users\Wlrdk\OneDrive\바탕 화면\과제\이강현\프로그래밍\코드\Assignment2\WA
16668개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...

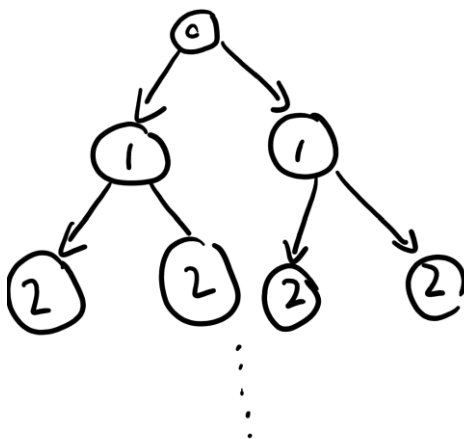
```

<출력>

고찰: 보드 판을 1차원 배열로 선언하고 3열로 출력하면 up키를 입력했을때는 -3 down키를 입력했을때는 +3, left는 -1, right는 +1로 생각하고 보드판을 넘어갈 때 입력을 무시하는 조건만 달아주었다. 이러한 알고리즘으로 보드판을 작성하면 1차원 배열의 길이만 늘려 선언한다면 그 이상의 크기의 보드판에도 적용할 수 있다. 2차원배열을 이용하여 코드를 짰다면 $n \times n$ 의 보드판이라고 가정했을 때 n 의 나머지가 0일 때 개행을 하는 연산을 하지 않아도 될 것 같다는 좋은점이 있을 것 같다.

문제 4

양의 정수 N 을 입력 받아 $\sum_{i=0}^N (i + 2^i)$ 을 계산하는 프로그램을 구현. 프로그램은 시작 시 숫자 N 을 입력 받으며 숫자에 따른 계산결과를 출력. 이 때 반복문(for, while, do while) 등은 사용하지 않고 함수의 호출을 이용해 구현한다.

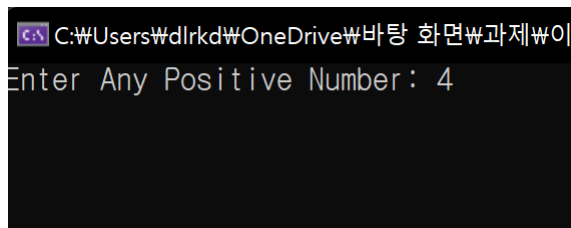


다음과 같은 함수 호출구조를 가진다.

필요한 개념: 다음과 같은 함수호출 개념으로 반복문을 사용하지 않고 코드를 작성하려면

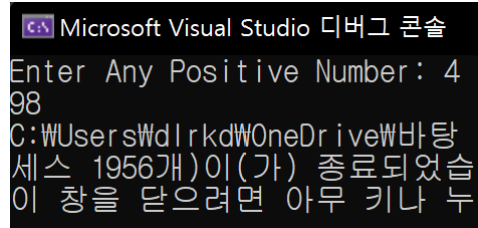
재귀함수의 이용과 재귀함수에서 함수를 호출할 때 두개의 함수를 호출해 나간다면 위와 같은 그림의 함수호출 구조를 가질 수 있을 것이다.

<입력>



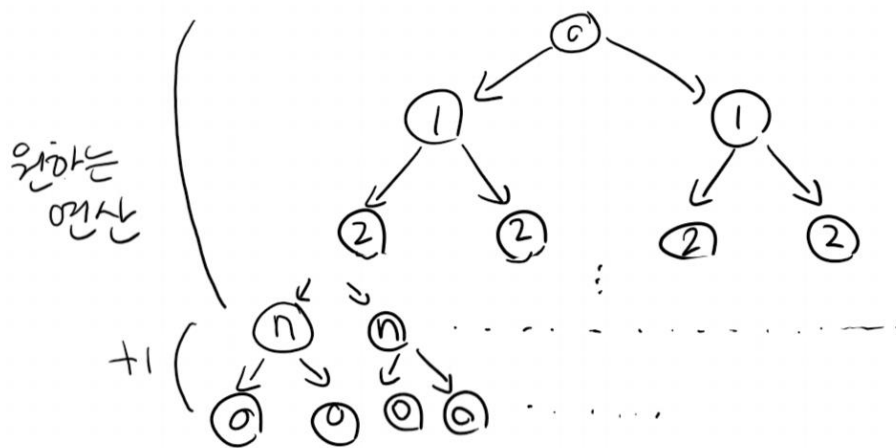
```
C:\Users\Wdlrkd\OneDrive\바탕 화면\과제\이  
Enter Any Positive Number: 4
```

<출력>



```
Microsoft Visual Studio 디버그 콘솔  
Enter Any Positive Number: 4  
98  
C:\Users\Wdlrkd\OneDrive\바탕  
세스 1956개)이(가) 종료되었습  
이 창을 닫으려면 아무 키나 누
```

고찰: 이 문제는 코드의 길이는 그리 길지 않았지만 함수의 호출구조를 명확히 이해하고 있어야 하는 문제여서 문제해결방향을 정하는데 시간이 많이 들었다. 2의 i승을 구하는 함수를 만들고 i를 -1해가며 곱해주는 함수 총 두가지의 함수를 정하여 중첩시켜 구하는 코드도 짜보았지만 함수의 반환값을 $n+f(i-1)+f(i-1)$ 의 꼴로 호출해나가니 하나의 함수로 훨씬 효율적이고 간결하게 코드를 구성할 수 있었다. 한가지 문제는 일반적인 재귀함수의 사용은 i값을 입력받고 점점 줄여나가는 방식으로 사용해야 반환값을 0으로 사용하게 되어 편리한데 계산식이 복잡해지니 반환값을 설정하는 것 어려움을 느꼈다. 그에 대한 해결책으로 입력받은 연산횟수보다 한번더 연산을 시행하고 그에대한 반환값을 0으로 설정하면 결국 더해도 0이므로 원하는 값을 얻을 수 있었다.



다 더하면 결국 한 번 더 한 연산 값들은
모두 0 이므로 원하는 값을 얻을 수 있다.