提出日　令和2年06月10日

アルゴリズムとデータ構造　第2回課題レポート

学籍番号（A19117）
氏名（永尾優磨）

【課題】

1. プログラミングにおける「再帰」について説明せよ。
2. 再帰を使って 1 から n までの整数の総和を求めるプログラムを作 れ。ただし，クラス名を Sum，総和を求めるメソッドを CalcSum とせよ。
3. 再帰を使って 1 から n までの階乗を求めるプログラムを作れ。た だし，クラス名を Fact，階乗を求めるメソッドを CalcFact とせよ。 課題 1（1 から n までの総和を求めるプログラム）にほん の少し手を加えるだけで完成する。ただし，課題 1 のファイルは残しておくこと。（レポートとして提出してもらう ため）
4. 再帰を使って n 枚のハノイの塔を解くプログラムを作れ。ただし、 以下の点に留意されたい。
   - “どの円盤” を “どこからどこへ” 動かしたのかがわかるよう に結果を出力すること出力例（ただし、A≡開始棒、B≡目的棒、C≡作業棒とする）
   - 1 の円盤を A から B に移す

【作成したプログラム】

```
public class Sum {
    public static void main(String[] args) {
        CalcSum(10);
    }

    public static int CalcSum(int n) {
        if (n > 1) {
            n = CalcSum(n - 1) + n;
            System.out.println(n);

        } else {
            System.out.println(n);

        }

        return n;
    }
}
```

課題2のプログラム

```
public class Fact {
    public static void main(String[] args) {
        CalcFact(10);
    }

    public static int CalcFact(int n) {
        if (n > 1) {
            n = CalcFact(n - 1) * n;
            System.out.println(n);

        } else {
            System.out.println(n);

        }

        return n;
    }
}
```

課題3のプログラム

```
public class hanoi {
    public static void main(String[] args) {
        hanoi(3, "A", "B", "C");
        // A＝開始棒、B＝目的棒、C＝作業棒とする
    }

    static void hanoi(int n, String start, String goal, String task) {
        if (n > 1) {
            hanoi(n - 1, start, task, goal);
            // 開始棒から作業棒への移動
            System.out.println(n + ":" + start + " -> " + goal);
            hanoi(n - 1, task, goal, start);
            // 作業棒から目的棒への移動
        } else {
            System.out.println(n + ":" + start + " -> " + goal);

        }

    }
}
```

課題4のプログラム

【プログラムの解説】

再帰について

再帰関数を簡単に言うと、「自分で自分を呼び出す関数」のことです。
メリットとしては、解こうとする問題が再帰的な性質を備えていると素直にプログラムを説くことが出来ます。処理を終わらせるために、処理条件の記述が必要。今回の課題4だとn=1の時に、結果をprint出力して終了する。となっている。
再帰関数の使用例として、

1. データ処理（複数のデータをソートしたり、繰り返し処理を行う場合）
2. 複雑な問題の解決（今回のハノイの塔のような問題）

などが挙げられる。
今回の課題では、簡単な挿話の計算を行った後に再帰関数で簡単に解くことが出来るハイノの塔を解く。

課題2について

S(1)=1
S(2)=S(1)+2
S(3)=S(2)+3
という連続した式をプログラムに起こしたものである。初めてreturn文を用いて出力した。if文を用いて、引数nが1よりも大きいときに、合計を計算するようにコーディングした。再帰を繰り返していき、n=1になった時の出力を1とした。これによって、再帰の無限ループを防いで、プログラムとして、成立させた。
S(1)=1より、最初の時点を1として、出力できるようにした。
returnで引数として参照できるようにした理由は、計算して、値を返す必要が出てきた時に、すぐに使えるようにするためのものである。

課題3について

課題2で作成したプログラム中にある　`n = CalcSum(n - 1) + n;`の演算子を＋から＊に変更するだけで、課題のプログラムを完成させることが出来た。

課題4について

最初のif文に入ったときは、開始棒から、作業棒に移動させた。そして、作業棒から目的棒への移動および出力を行った。

1. 開始状態
2. 1枚目～n-1枚目を作業棒へ移動
3. n枚目を目的棒へ移動
4. 1枚目～n-1枚目を目的棒へ移動

という流れでコードを書いた。
if(n>1)という条件にした理由は、1枚目～n-1枚目までを他の棒へ移動したと考えたとき、1枚しか、作業棒に残っていないため、if(n>1)という条件にしている。

【結果】

```
ws\SchoolJava_575429d3\bin 第02回再帰.Sum
1
3
6
10
15
21
28
36
45
55
```

図1　Sum.javaの実行結果

```
ws\SchoolJava_575429d3\bin 第02回再帰.Fact
1
2
6
24
120
720
5040
40320
362880
3628800
```

図2　Fact.javaの実行結果

```
ws\SchoolJava_575429d3\bin 第02回再帰.hanoi
1:A - > B
2:A - > C
1:B - > C
3:A - > B
1:C - > A
2:C - > B
1:A - > B
```

図3　hanoi.javaの実行結果

【考察】

Pythonでコードを作成してからJavaに書き換えるという形で課題を解決した。静的型付け言語の書き方に慣れるのはまだまだ時間がかかりそう。アルゴリズムは、ifとfor文で書かれているイメージなので、書き方に慣れてきたら、フローチャートなどを書いて、Pythonで書くということを経由せずにそのままJavaで書いていきたいと思う。

初めて、再帰関数を用いて、プログラムを記述した。課題2,3は、for文を使わずに各やり方を新たに知ることが出来た。

問が再帰的な性質を持つということを気づくのが一番難しい点だと考えた。そのため、問題文中で起こっていることを一度数式にすると考えると、再帰的な性質を持つかを容易に判断することが出来ると考える。
数式にしてから、コードを記述するというやり方で書くと、コードを簡単に記述することが出来る。そのため、問題を解くために、数学力を鍛えると容易にコードを書くことが出来ると考える。

【参考文献】

「プログラムができない人用の再帰関数」〈https://qiita.com/nomunomu0504/items/125483f57b0ecc69ab82〉
「Pythonで再帰関数を作る方法【初心者向け】」〈https://techacademy.jp/magazine/19308〉
増井敏克(2020)「Pythonで始めるアルゴリズム入門」翔泳社
「衰えか･･･」〈http://sapphire.hatenadiary.com/entry/20100604/1275661981〉
「 Javaのreturn文の使い方を現役エンジニアが解説【初心者向け】」〈https://techacademy.jp/magazine/22242〉

「参考文献の書き方」〈http://www7a.biglobe.ne.jp/nifongo/ron/ron_04.html〉