プログラミング演習I第4回レポート

学籍番号：2364902

名前：キム　ギュソク

(1) 課題番号と課題内容

A. 基本課題 3

i. 任意の自然数（ n <= 2,147,483,647 ）を引数に呼び出されると、再帰処理を使って 3桁ごとに「 ,（ カンマ ）」で区切って出力する関数を作成

1. 入力値 n が 3 桁以下ならば、入力された値をそのまま表示する
2. そうでなければ、n/1000 を値を n として関数を再帰的に呼び出す(つまり、下 3 桁を除いた上記桁を n として呼び出す ）
3. 呼び出した関数から戻ってきた後に、「,」と再帰呼び出しの際に除いた下 3桁（ n%1000 ）を表示する。(例えば、12035007 なら再帰呼び出しで 12,035 を表示してから「,007」を表示する）

(2) フローチャートまたは疑似言語によるアルゴリズムの記述

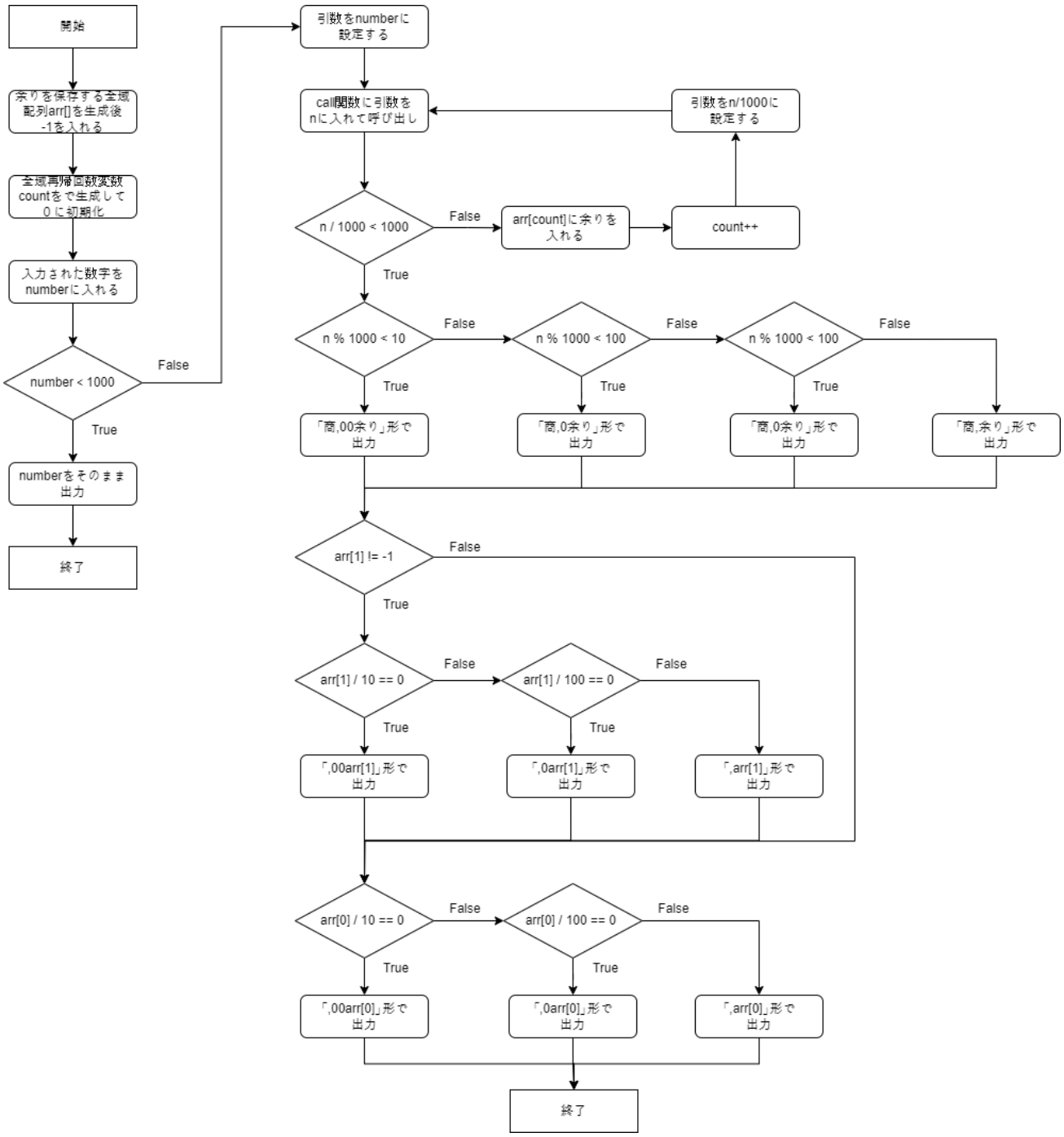


図 1。フローチャート

(3) アルゴリズムが正しいことの説明

i. 正当性

1. 3桁の場合

A. number＜1000の場合はそのまま出力するように設計されているのでコンマ無しで出力する。よって、正当な結果が得られる。

2. 3桁以上の場合

A. 1000に割った商が3桁になるまで余りは配列に保存して商は続けて1000で割る作業をする。そして、商が3桁になったときは余りが3桁か2桁か1桁かを調べてそれに応じて 0 を追加して出力する。そして、再帰するたびに保存しておいた消えた余りを桁に合わせて 0 を追加して出力する。 nが2,147,483,647の場合再帰するたびに消える可能性たがある余りは483と647の二つのあまりである。よって、483（arr[1])，647(arr[0])順に出力する必要があるのでまずarr[1]が存在するかを確認して存在するとまた同じく桁数に合わせてコンマを入れて 0 を追加して出力する。その後やarr[1]が存在しない場合はarr[0]が存在するか確認する。そして、arr[0]を桁数に合わせてコンマと 0 を入れて出力する。このような方法を使うと再帰で消えてしまう余りも出力することができるので正しくコンマが入った数字列が表現できる。よって、このアルゴリズムは正当だと言える。

ii. 停止性

1. 今回の再帰関数では入力された数字を1000で割ったときにその商が3桁になるまで再帰する関数である。そして、3桁以上の場合は商を入れて関数を呼んでいる。そのため関数に入れる数字は再帰するたびに小さくなるのでいつか商が3桁になり正しく停止すると言える。

$$1回目：n$$

$$2回目：\frac{n}{1000}$$

$$3回目：\frac{n}{1000}\times\frac{1}{1000}$$

$$\cdots$$

$$n回目：\frac{n}{1000^n}$$

$$\lim_{n\to\infty}\frac{n}{1000^n}=0$$

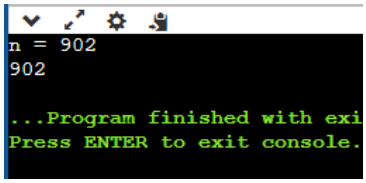
(4) ソース・プログラムの説明

ソースコード

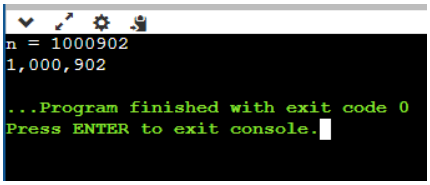
main.c

```c
#include <stdio.h>

static int arr[] = {-1, -1};//余りを保存する変数要素がない時の識別数字-1
static int count = 0;//何回再帰関数を読んだか確認する変数

int call(int n);

int main(void){
    int number = 0;

    printf("n = ");
    scanf("%d", &number);

    if(number < 1000){
        printf("%d", number);
        return 0;
    }
    else{
        call(number);
    }


}

int call(int n){
    if(n/1000 < 1000){
        if(n%1000 < 10){//余りが1桁の場合
            printf("%d,00%d", n/1000, n%1000);
        }
        else if(n%1000 < 100){//余りが2桁の場合
            printf("%d,0%d", n/1000, n%1000);
        }
        else{//余りが3桁の場合
            printf("%d,%d", n/1000, n%1000);
        }
        //もし、2,147,483,647の場合は
        if(arr[1] != -1){//483の部分
            if(arr[1]/10 == 0){//消えた余りが1桁の場合
                printf(",00%d",arr[1]);
            }
            else if(arr[1]/100 == 0){//消えた余りが2桁の場合
                printf(",0%d", arr[1]);
            }
            else{//消えた余りが3桁の場合
                printf(",%d", arr[1]);
            }
        }
        //647の部分
        if(arr[0]/10 == 0){
            printf(",00%d",arr[0]);
        }
        else if(arr[0]/100 == 0){
            printf(",0%d", arr[0]);
        }
        else{
            printf(",%d", arr[0]);
        }
    }
    else{//消える余りが存在するとき
        arr[count] = n%1000;//保存
        count++;
        return call(n/1000);
    }
```

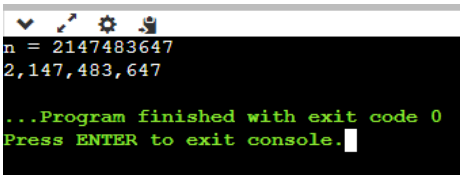
実行結果1



実行結果 2



実行結果3



A. ソースコードの説明

i. 再帰で消える余りを保存する配列arr[]を要素-1で宣言する。この時全域変数として宣言することでどこの関数でも参照できるようにする。また要素-1は配列が空いているかを確認する数字である。そして、再帰するたびに増えるcount全域変数を宣言する。Userから数字を入力してもらってすぐに入力された数字が1000より小さいのかを確認する。3桁の場合はコンマが必要ないからである。もし、1000以上の場合はcall関数に数字を入れて呼び出す。call関数では3桁ごとに調べる必要があるので1000に割った商が3桁以下なのか以上なのかをif文を利用して確認する。もし、3桁以上の場合はもう一回1000で割る必要があるのでもう一回1000に割った商を入れて再帰する。しかし、この時余りが消えるので余りを配列に入れてcount++をする。このようにして商が3桁になるまで繰り返して3桁になると一番最上位の桁なのでその桁の前にはコンマや2桁になって 0 が必要になったりしないのでそのまま出力してコンマを入れて余りを桁数をif文を利用して確認して 0 を加えて出力する。その後、消えていった余りを出力するために配列の要素が存在するか確認する。printfの場合左から順に出力するので後で配列に入った要素を先に調べる必要があるのでarr[1]から存在を確認して桁数に合わせて出力する。その後、arr[1]が存在するとarr[0]は必ず存在するのでarr[0]は存在を調べることなく桁数に合わせて出力する。

(5) 考察

A. 再帰関数を利用して得られる利点について考えたいと思う。課題 2 でも言っているように再帰関数の場合次の計算との連動性が長けているのでこのような次の再帰関数の引数として前回の割った商を渡すのでより直観的なコードが作れる。よって、この問題の場合は次に計算結果を渡す必要があるので繰り返し処理より再帰関数を利用した計算がより効率的だと考える。また、今回は桁数に合わせて 0 を追加して出力するコードを作成したが、%03dを利用することでより簡単なコードを作れるので。桁数に合わせて 0 を追加するときは%[追加したい文字][何桁]dを利用することがより効率的である。そして、そのようなコードを作成すると無駄に長いコードが短くなるので読みやすくなるので効率的だと考える。

(6) 感想

A. 桁数に合わせて 0 を表現するときには%03dという方法があることを知った。また、再帰関数を使うことで得られる利点について考える機会になった。