

計算機入門及び演習  
C 言語レポート課題\_\_くり返し文

6321120  
横溝尚也

提出日：11月 04日 (木)

## 1 レポート課題 1

次の漸化式で定義される数列について、 $x_5$  を出力する。

$$x_n = x_{n-1} \times 4 - 8$$
$$x_{10} = 87384$$

## 2 アルゴリズムの説明

漸化式は、初期値とその初期値から漸化式に代入することで芋ずる式にほかの値もわかる。 $x_{10}$  から 9, 8, 7, と一つ一つ求めていって、最後に  $x_5$  を出力すれば良い。

## 3 プログラムの説明

```
#include <stdio.h>

1int main (){
2
3    int i;                                /*整数型変数 i の宣言*/
4    int xn = 87384;                       /*整数型変数 xn の宣言と代入*/
5
6    for(i ==10;i > 4; i--){               /*i についてのくり返し文*/
7        if(i = 5){                        /*i=5 の時か否かで条件分岐*/
8            printf("x%d = %d\n" , i , xn); /*条件式を満たす時の処理*/
9        }
10       else{                             /*i=5 を満たさないときの場合*/
11           xn = xn + 8;                    /*xn に 8 を足す*/
12           xn = xn / 4;                   /*xn を 4 で割る*/
13
14       }
15}
```

まず初めに 3, 4 行目でくり返し文に使う整数型変数  $i$  と、問題文で与えられている  $x_{10}$  を変数として格納する。

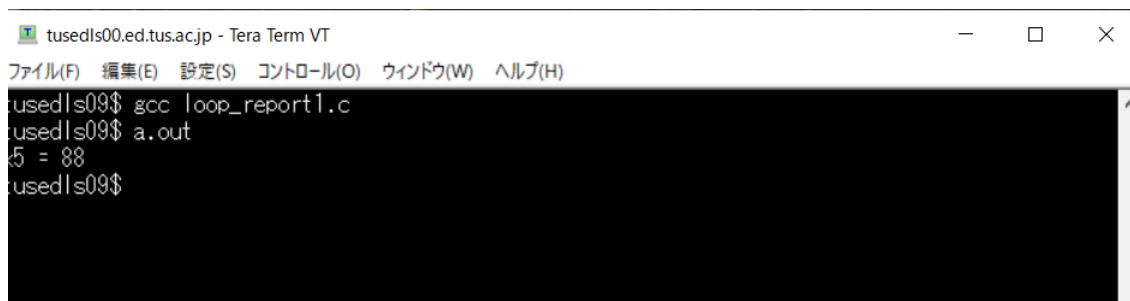
6 行目～12 行目ではくり返し文を使用している。 $n=5$  の時の値を出力したいので  $i=10$  から初めて、 $i$  を 1 ずつ減らしていき  $i=5$  までくり返し文の範囲を設定する。

7 行目から 13 行目がくり返し文の中で行うプログラムであり、7 行目で if 文を利用している。 $i=5$  であるならば、 $x_5$  の値を出力、それ以外ならば、 $x_n$  に 8 を足した後、4 で割る作業を行う。これは問題文の漸化式から  $n$  の値が一つ少ない項の計算を行っている。このようにして  $x_5$  まで計算していけばよい。

## 4 考察

この問題は  $n=10$  の時の値が問題文によって定められており、 $n=5$  を出力するという問題である。これは自分の手で一つ一つ計算していき、答えを求めることもできるし、プログラムの中で8を足して、4で割るという作業を5回すべて打ち込んでも求める値は出力することができる。しかし、 $n$  の値がかなり大きくなったり、すると一つ一つ打ち込むことはかなり大変である。そこで繰り返し文は変数の初期値と何回繰り返すのかを指定するだけですべて計算してくれるので繰り返し文を利用することがこの問題において一番適切であると考えた。

## 5 実行結果



```
tusedls00.ed.tus.ac.jp - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
tusedls09$ gcc loop_report1.c
tusedls09$ a.out
5 = 88
tusedls09$
```

図1 実行結果 (1)

## 1 レポート課題2

フィボナッチ数列の第1項から第10項までを出力する。

フィボナッチ数列は次の漸化式で定義される。

$$x_1 = 0$$

$$x_2 = 1$$

$$x_n = x_{n-1} + x_{n-2}$$

## 2 アルゴリズムの説明

フィボナッチ数列とは1項目を0, 2項目を1とし、3項目以降をその項の前2項の和と定めた数列のことである。つまり新たに次の項の値を知りたいければ前2項を足した値であり、順々にすべての項がわかるのである。

また、新たな項の値を調べるときに行う作業はすべて同じであるので、第10項まで知りたいければ第1項と第2項はわかっているので8回同じ作業を繰り返せばよいと解釈できる。

## 3 プログラムの説明

```
#include <stdio.h>
```

```
int main (){
```

```
2   int i;                                /*整数型変数 i の宣言*/
3   int xn = 0;                            /*整数型変数 xn の宣言と代入*/
4   int yn = 1;                            /*整数型変数 yn の宣言と代入*/
5   int zn = xn + yn;                      /*整数型変数 zn の宣言と代入*/
6
7   for(i = 1; i < 11; i++){               /*くり返し文を利用して10回繰り返す*/
8       if(i == 1){                        /*条件分岐 i=1 の場合*/
9           printf("x%d = %d\n" , i , xn);
10      }
11      else if (i == 2){                   /*i=2 の場合*/
12          printf("x%d = %d\n" , i , yn);
13      }
14      else{
15          printf("x%d = %d\n" , i , zn );
16          xn = yn;                        /*xn の値を yn に変更*/
```

```

17         yn = zn;                                /*yn の値を zn に変更*/
18         zn = xn + yn;                            /*zn の値を xn+yn に変更*/
19     }
20 }

}

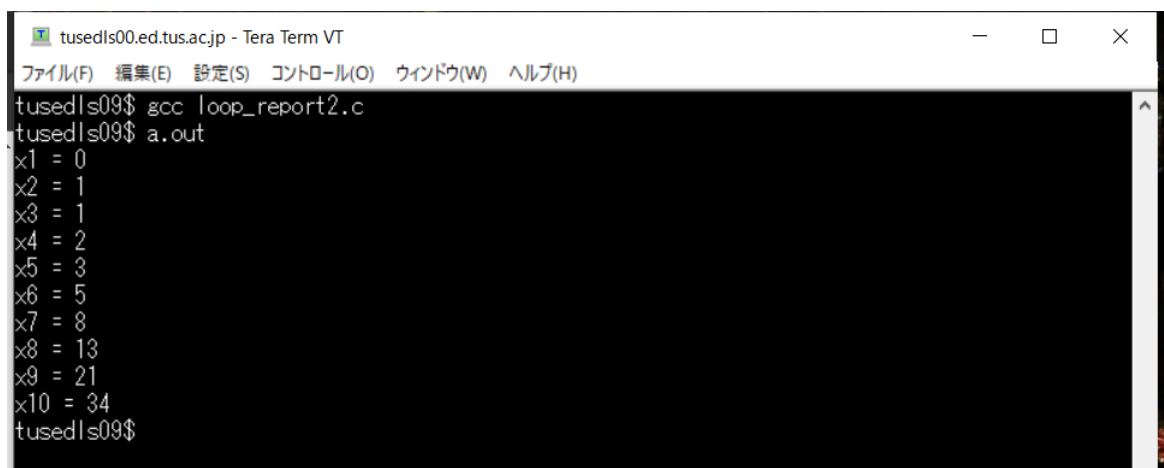
```

まず初めに2～5行目で4つの整数型変数の宣言を行う。アルゴリズムの説明でも述べた通り、第1, 2項から第3項目を調べることと本質的に同じことをすれば、第10項までも求まるのでくり返し文を利用する。くり返し文を利用するにあたってどの範囲で繰り返すか指定するための変数*i*が変数の一つ目である。2つ目、3つ目が既知の二つの項であり新たな項の値を調べるための材料を格納するための変数、4つ目は2つ目3つ目から調べる新たな変数である。

次に7行目以降でくり返し文を利用している。繰り返す範囲は単純に第1項から第10項を調べるので*i*が1から10までの間である。

8から13行目では特殊な場合の処理を表している。*i*が1か2の時は既に問題文から自明であるのでただその項を出力すればよい。14行目以降では新たな項を出力した後に次に繰り返したときに出力する値を変更しなければならないので、16行目から18行目のように変数を変更する。

## 4 実行結果



```

tusedls00.ed.tus.ac.jp - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
tusedls09$ gcc loop_report2.c
tusedls09$ a.out
x1 = 0
x2 = 1
x3 = 1
x4 = 2
x5 = 3
x6 = 5
x7 = 8
x8 = 13
x9 = 21
x10 = 34
tusedls09$

```

図2 実行結果 (2)

## 5 考察

今回の問題で重要な点は新たな項を調べたい項がその項の前2項の和でどれも表せられるという事である。これによってくり返し文という同じ操作を何度もするときに便利な関数を使うことによってプログラムの簡潔

さがかなり増した。i の値を変更するだけでフィボナッチ数列の調べたい項だけを自由自在に出力できるプログラムとなっており、その点でも優れたプログラムだと思う。

また繰り返し文の中での処理には一つ工夫が必要だった。それが i が 1 と 2 の時である。ここで同じように変数の値を変更してしまうと第 3 項以降の値しか出てこない。よって繰り返し文の中で if 文を利用して条件分岐する必要があった。

## 1 レポート課題3

以下のように出力する。



```
$ ./a.out
*****
*       *
**    **
***  ***
**** *****
***** *****
***  ***
**    **
*       *
*****
$
```

図 3

## 2 アルゴリズムの説明

今回の課題ではある規則性に基づいて記号を出力していくという課題である。＊を出力するマス、空白として何も出力しないマスも含めて縦と横に座標を振る。そうすれば、例えば”縦3，横2の場所に＊を出力する”といった風にコンピュータでも解釈できる。

## 3 プログラムの説明

```
#include <stdio.h>

1int main(){
2
3    int i,j;                                /*整数型変数 i,j の宣言*/

4    for(i = 0;i < 10;i++){                  /*以下の処理を10回繰り返す*/
5
6        for(j = 0;j < 10; j++){             /*以下の処理を10回繰り返す*/
7            if(i == j){                     /*i と j が等しい場合*/
8                printf(" ");                /*空白の出力*/
9            }
10           else if(i + j == 9){/*i+j が9である場合*/
11               printf(" ");                /*空白の出力*/
12           }
13           else{                            /*二つの条件に一致しない場合*/
14               printf("*");                /*＊を出力*/
15           }
16       }
17   }
```

```

15         }

16     }
17     printf("\n");        /*改行する*/
18 }
19}

```

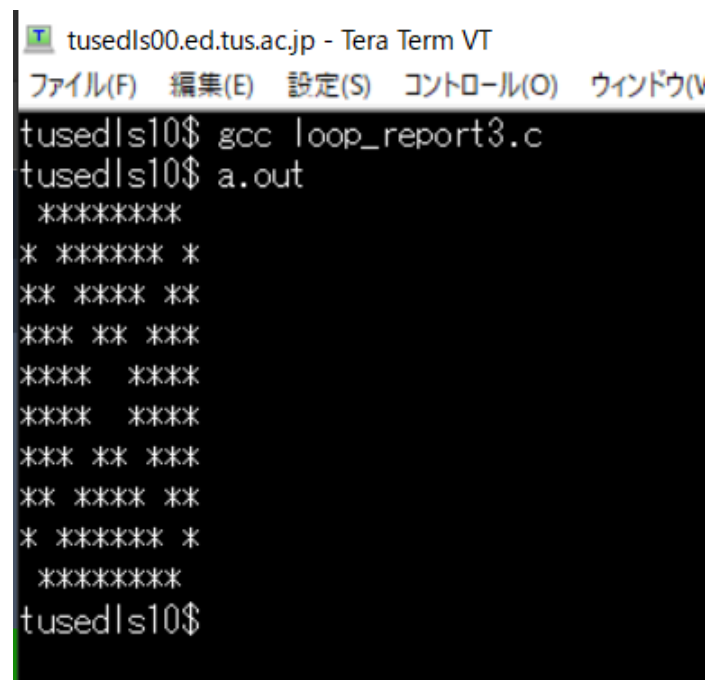
初めに3行目で変数の宣言を行っている。今回のプログラムではくり返し文を2回使用しているのでそれを利用する変数をひとつずつ宣言している。

4行目から18行目までが縦の行に関するくり返し文である。6行目から17行目までのプログラムをくり返し行うことになっている。出力したい形は10行であるので10回繰り返している。

次に6行目から17行目で再びくり返し文を利用している。これは横の列に関するくり返し文である。出力したい形から横はすべて10列であるので今回も10回繰り返せばよい。

7行目以降でif文を利用している。これはある規則性によって何を出力するのか判別するために利用している。

## 4 実行結果



```

tusedls00.ed.tus.ac.jp - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W)
tusedls10$ gcc loop_report3.c
tusedls10$ a.out
*****
* ***** *
** ***** **
*** ***** ***
**** ***** ****
***** *****
***** *****
*** ***** ***
** ***** **
* ***** *
*****
tusedls10$

```

図4 実行結果 (3)



## 5 考察

$10 \times 10$  マスすべて\*を出力するのであれば、くり返し文を2回利用してかつ、常に\*を出力して入れれば良い。しかし今回の課題はある規則性によって出力しないのか、\*を出力するのか判断しなければならないここでは縦成分を  $i$ 、横成分を  $j$  として座標を  $(i,j)$  と表すとする。ただし、左上を  $(0,0)$ 、右下を  $(9,9)$  とする。規則性を考えると、 $(0,0),(1,1),(2,2),\dots(9,9)$  と、 $(0,9),(1,8),(2,7)\dots(9,0)$  の座標は何も出力していないことがわかる。よってくり返し文の中で  $i=j$  の時と  $i+j=9$  の時だけ空白を出力するように条件分岐すればよいことがわかる。

今回のプログラムではくり返し文を2回利用している。くり返し文を1回のみ利用して各行ごとにどのように\*を出力すればいいかすべてプログラムすることもできるが2回利用することによってかなり簡潔になり、見やすくなったと考える。

## 1 レポート課題 4

256 と 368 の最大公約数を出力する。

## 2 アルゴリズムの説明

2 整数の最大公約数を求めるときに、プログラミングではなく数学的に求めるときも主にユークリッドの互除法を利用する。これは 2 数のうち大きな数  $a$ , 小さな数  $b$ ,  $a$  の  $b$  による剰余を  $r$  としたとき、 $a$  と  $b$  の最大公約数は、 $b$  と  $r$  の最大公約数に等しいという定理である。これを今回でも利用することによってプログラミングを行う。

## 3 プログラムの説明

```
#include <stdio.h>

1 int main(){

2     int num1 = 256;          /*整数型変数 num1 の宣言と代入*/
3     int num2 = 368;          /*整数型変数 num2 の宣言と代入*/

4     if( num1 >= num2)        /*num1 と num2 のどちらが大きいかで条件分岐*/
5         int i = num1 % num2;  /*整数型変数 i の宣言と代入*/

6         while(i != 0){        /*i が 0 出ないときに行う繰り返し文の条件式を設定*/
7             num1 = num2;      /*num1 を num2 に変更*/
8             num2 = i;         /*num2 を num1 に変更*/
9             i = num1 % num2;   /*再び新たな num1 と num2 から剰余を設定*/
10        }
11        printf("%d\n" , num2 ); /*i が 0 になったときに num2 を出力*/
12    }

13    else{                     /*num2 の方が num1 より大きい場合の処理*/
14        int i = num2 % num1;   /*以下同じ処理を行い、最大公約数を出力*/

15        while(i != 0){
16            num2 = num1;
17            num1 = i;
18            i = num2 % num1;
```

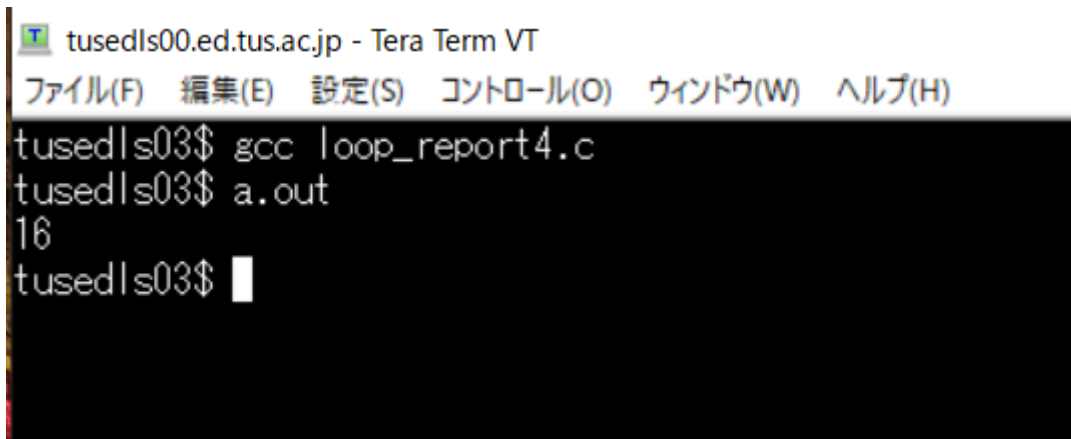
```

19     }
20     printf("%d\n" , num1 );
21 }
22}

```

まず2, 3行目で最大公約数を調べたい2数を変数に格納している。次に4から12行目と、13行目から21行目では格納した2数のうちどちらの方が大きな数なのかで場合分けしている。5行目から12行目では新たに変数を宣言し、それを2数の剰余としている。そのあとに剰余が0でない場合にくり返し以下の処理を行う while 文を使用している。その処理の内容が7, 8, 9行目である。2数の剰余が0になるまでくり返し剰余をとっていくので num2 を num1 に、i を num2 に、i を新たに変更した num1 と num2 の剰余にそれぞれ変更している。11行目は while 文の処理が終了した、つまり剰余が0になった後のプログラムである。最終的に求めたい最大公約数は剰余が0になった後の num2 であるので num2 を出力している。15行目以降は num1 と num2 の大小関係を逆にした時の処理より、前半と本質的には同じ操作をしている。

## 4 実行結果



```

tusedls00.ed.tus.ac.jp - Tera Term VT
ファイル(F)  編集(E)  設定(S)  コントロール(O)  ウィンドウ(W)  ヘルプ(H)
tusedls03$ gcc loop_report4.c
tusedls03$ a.out
16
tusedls03$ 

```

図5 実行結果 (4)

## 5 考察

こんかいのプログラムでは剰余が0になるまでくり返し同じ処理を行いたい課題であった。そこで、ある条件式を満たす場合くり返し処理を行う while 文が適切であると考えた。初期値といつまで繰り返すのか指定しなければいけない for 文と比べてより簡潔にプログラムできたと思う。

このプログラムでは調べたい2数の最大公約数をはじめに変数として格納することで整数であればどんな数でも出力することができる。今回のプログラムには入れ込まなかったが、num1 と num2 が整数ではないとき、エラーを表示するように if 文を利用すればかけると考える。また、考察を書いているときに気づいた改善点がある。このプログラムでは大きく分けて num1 が大きい場合と num2 が大きい場合で同じことを2回も書いている。しかし、初めに if 文を利用して num2 の方が num1 よりも大きい場合は変数の格納を逆にして、そ

ののちに6行目から12行目を書けばよいとおもった。今回の課題ではそこまでプログラムが長くないが、プログラムが長くなってくると同じことを2回書くことがかなりの手間になり、簡潔さに欠けると思った。

## 1 レポート課題5

限りなく円周率に近い値を出力する。

## 2 アルゴリズムの説明

円周率に限りなく近い値を出力するにはまず円周率がどのように定義されているかを考える。そこで今回自分が使用した円周率を近似していくために必要な公式は、グレゴリーライプニッツ級数である。この級数は以下のように定められている。

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \quad (1)$$

この式からまず  $\pi/4$  を求めて最後にその値を4倍すればよい。

ここで  $\pi/4$  を求める過程について考える。式(1)の右辺の分母に着目すると奇数が順に並んでいる。またそれぞれの項が正負を繰り返す交項級数である。この2点の規則性をうまくプログラミングすれば円周率の値を近似できると考える。

## 3 プログラムの説明

```
#include <stdio.h>

1 int main() {
2     int i;                                /*整数型変数の宣言と*/
3     float pi = 0;                         /*少数型変数 pi の宣言*/
4     float x , y , z;                     /*少数型変数 x,y,z, の宣言*/
5     for (i = 1; i < 1000000; i++){ /*i の初期値と繰返し範囲を指定し、以下の処理を繰り返す。*/
6         x = 2*i - 1;                     /*x を 2i-1 とする*/
7         if(i % 2 == 1){                  /*i を 2 で割ったときの剰余が 1 の時の場合*/
8             y = 1 / x;                   /*y の値を 1/x とする*/
9             pi = pi + y;                  /*pi の値を pi+y とする*/
10        }else{                            /*i を 2i-1 で割ったときの剰余が 1 でない時の場合*/
11            y = -1 / x;                   /*y の値を -1/x とする*/
12            pi = pi + y;                  /*pi を pi+y とする*/
13        }
14    }
15    z = 4*pi;                             /*z を 4pi とする*/
```

```
16    printf("%f\n" , z);          /*z の値を出力する*/
```

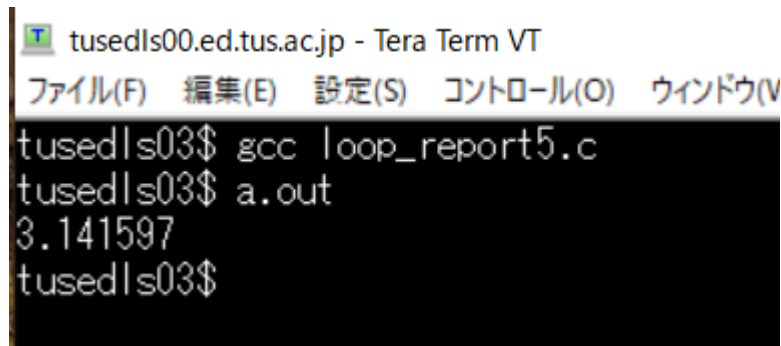
```
17}
```

2行目で宣言している変数  $i$  は for 文を使うときに初期値を設定するために使用する。3, 4行目で少数型変数を4つ宣言する。5行目での for 文で1000000より  $i$  が小さいときに繰り返すよう設定しており、これはライプニッツ級数無限に展開される項を1000000項でやめたのという事である。

7行目以降では if 文を利用して  $\pi$  に足すようなのか引くようなのかを判別している。なお、6行目で  $x$  の値を常に奇数となるように設定している。

最後に15, 16行目で求めた  $\pi$  を4倍してそれを出力している。

## 4 実行結果



```
tusedls00.ed.tus.ac.jp - Tera Term VT
ファイル(F)  編集(E)  設定(S)  コントロール(O)  ウィンドウ(W)
tusedls03$ gcc loop_report5.c
tusedls03$ ./a.out
3.141597
tusedls03$
```

図6 実行結果 (5)

## 5 考察

円周率を近似しているライプニッツ級数は規則性があり、シグマ記号であらわすこともできる。このことから for 文を使うことでうまく円周率を表すことで繰り返す回数をさらに多く試行すればより円周率に近い値を出力できる。

今回自分がプログラムした for 文での繰り返し回数は  $10^6$  回である。この繰り返し回数を  $10^7, 10^8$  回に変更し、より細かく近似しようとしても同じ 3.1415997 と同じ結果が出力される。また、 $10^9$  にすると teraterm が重くなり実行結果を出力するのに1分以上待っても出力されない。実行時間と近似の具合から考えて、 $10^6$  回が最適であると考えられる。またこの出力結果である、3.141597 は問題文に掲載されていた 3.141522 よりも正確に近似できている。

## 6 感想

今回の課題を行うことでただ繰り返し文について身につけた知識を5通りの課題を通して繰り返し文の汎用性の高さを実感した。また、課題5の円周率の近似をプログラムするのがとても時間がかかった。