

2021 計算機プログラミング演習

第6回

for 及び while による繰り返し

第6回の目標

- for による繰り返し処理
 - コンピュータは単純な繰り返しでも不平を言わない
 - それが何千回でも何億回でも
 - さらにそれを高速に実行する！！
 - 繰り返しを上手に使うことによって、
 - プログラム作成時の少ない手間で大きな成果が得られる
 - プログラムの役割がまとめられ(ブロック化)、理解しやすくなる
 - バグの発生が少なくなる
 - 反面、繰り返しの実現方法に慣れが必要
 - 繰り返しの規則性を発見する
 - 条件判断(`if`)との組み合わせによる処理の複雑化
 - 繰り返しの規則に合わない場合をどう処理するか

for文

- for文は指定した繰り返し回数まで繰り返し処理を行う
- 繰り返し回数が決まっている場合はfor文を使用する
- 繰り返し回数が分からない場合はwhile文を使用する

```
int i;
```

```
for( i = 1; i <= 繰り返し回数; i++ ){  
    実行文;  
}
```

for の使い方 (1) 基本

```
/*      for_test1.c      */

#include <stdio.h>

int main( void ){
    int    i ;

    for ( i = 1 ; i < 10 ; i++ ) {

        printf( " i = %d\\n", i ) ;

    }

    return 0 ;

}
```

ループ変数 `i` を最初に1に設定し(`i = 1`)、中括弧内の処理(`printf(...)`)を実行したあと、`i` を1増やす(`i++`)。
`i` が10未満(`i < 10`)の間は中括弧内の処理を実行する。

1 から 9 までの数を入力

for の使い方 (2) 基本

```
/*      for_test1.c      */
```

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

```
int main( void ) {  
    int i ;
```

```
    for ( i = 1 ; i <= 10 ; i++ ) {
```

```
        printf( " i = %d¥n", i ) ;
```

```
    }
```

```
    return 0 ;
```

```
}
```

ループ変数 i を最初に1に設定し($i = 1$)、
中括弧内の処理(`printf(...)`)を実行したあと、
 i を1増やす($i++$)。
 i が10以下($i \leq 10$)の間は
中括弧内の処理を実行する。

1 から 10 までの数を出力

for の使い方 (3) 基本

```
/*      for_test1.c      */
```

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

```
int main( void ){  
    int    i ;
```

```
    for ( i = 1 ; i <= 10 ; i = i + 2 ) {
```

```
        printf( " i = %d¥n", i ) ;
```

```
    }
```

```
    return 0 ;
```

```
}
```

ループ変数 `i` を最初に1に設定し(`i = 1`)、中括弧内の処理(`printf(...)`)を実行したあと、`i` を2 増やす(`i = i + 2`)。
`i` が10 以下の間は中括弧内の処理を実行する。

1 から 9 までの奇数を出力

for の使い方 (4) 基本

```
/*      for_test1.c      */

#include <stdio.h>

int main( void ){
    int    i ;

    for ( i = 10 ; i >= 1 ; i-- ) {

        printf( " i = %d¥n", i ) ;
        10 から 1 までをカウントダウン
    }

    return 0 ;

}
```

for の使い方 (5) 合計

```
/*      for_test2.c      */

#include <stdio.h>

int main( void ){
    int    i ;
    int    sum ;
    sum = 0 ;      合計の初期値は 0

    for ( i = 1 ; i <= 10 ; i++ ) {
        sum = sum + i ;
    }

    printf( "SUM = %d¥n", sum ) ;

    return 0 ;      1 から 10 までの合計を出力
}
```


for の使い方 (6) 計数(カウント)

1～10の間の3の倍数の個数

```
/*      for_test2.c      */

#include <stdio.h>

int main( void ){
    int    i ;
    int    count ;
    count = 0 ;      初期値は 0

    for ( i = 1 ; i <= 10 ; i++ ) {
        if ( i % 3 == 0 ){

            count = count + 1 ;      条件に合ったときに1
                                     プラスする
        }
    }
    printf( "1から10の間の3の倍数の個数 %d\n", count ) ;
    return 0 ;      最終的に条件に合った個数が得られる
}
```

for の使い方 (7) break による脱出

```
/*          for_test2.c          */

#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>

int main( void ){
    int    i, a;
    int    sum = 0 ;

    for ( i = 1 ; i <= 10 ; i++ )
    {
        scanf("%d",&a );

        if ( a < 0 ) { break ; }

        sum = sum + a ;
    }
    printf( "%d 個の合計: %d\n", i - 1, sum ) ;
    return 0 ;
}
```

初期値は 0

入力値がマイナスならループを中断

【注意！】 break による中断は脱出条件がわかりにくくなるので、バグのもとになりやすい。できるかぎり避けること。

実行結果

- for の指定した回数を繰り返すことが基本
- ただし、条件により途中で繰り返しを中断する場合は break を使う。



```
cmd コマンドプロンプト - [X]  
> for_test2  
8  
9  
4  
-1  
3 個の合計 : 21  
> _
```

The screenshot shows a Windows Command Prompt window with a yellow title bar. The window title is 'cmd コマンドプロンプト'. The command prompt shows a series of inputs and outputs. The first input is '> for_test2', followed by the numbers 8, 9, 4, and -1 on separate lines. Then, the output '3 個の合計 : 21' is displayed. The prompt continues with '>' and a cursor on a new line.

for の使い方 (8) 実数のループ変数

- ループ変数には整数(int)だけでなく、実数(double, float)も使える。
 - 使えるけれども、**注意が必要**。

```
/*      for_test3.c      */

#include <stdio.h>
#include <math.h>

int main( void ){
    double th ;

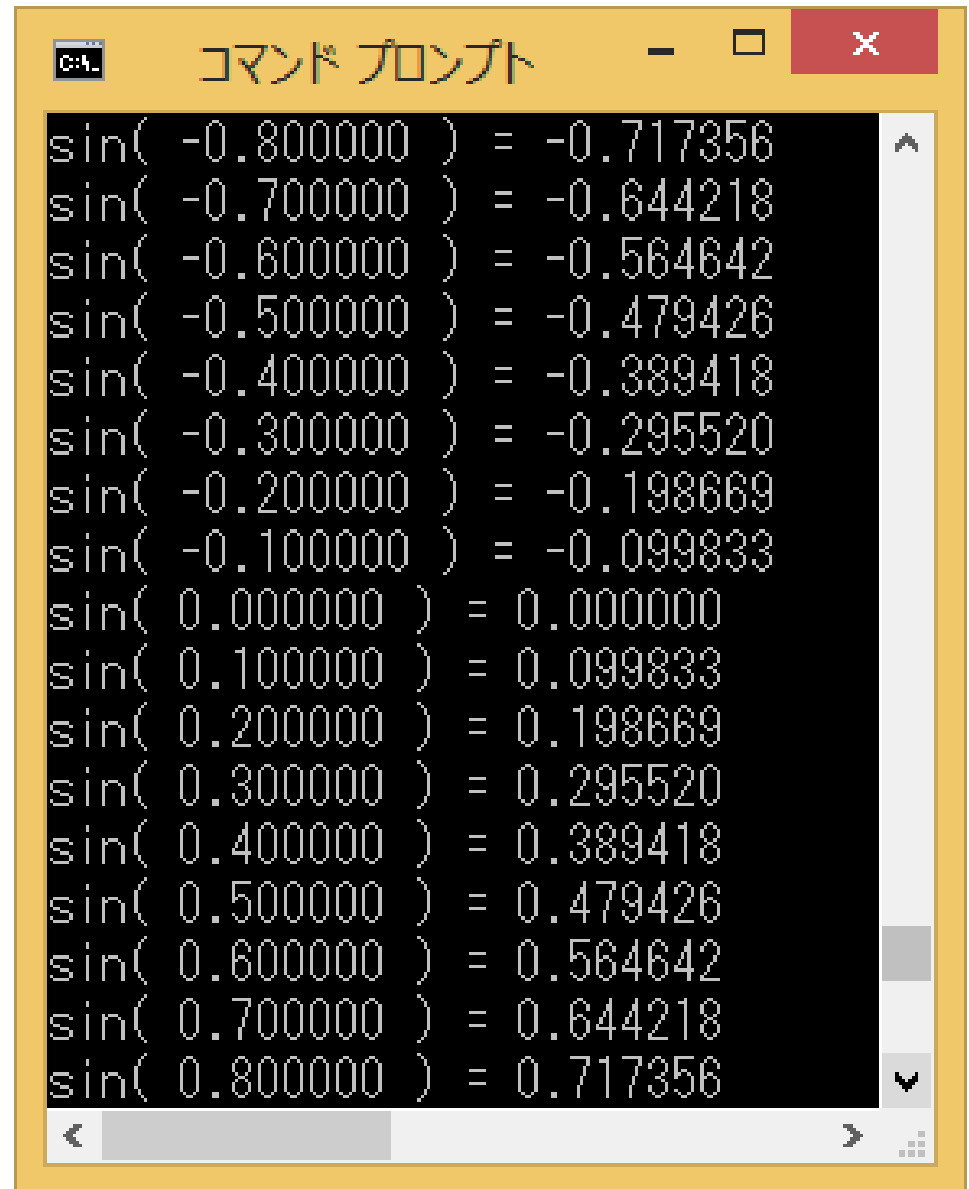
    for ( th = -3.1 ; th <= 3.1 ; th += 0.1  ) {

        printf("sin( %f ) = %f¥n", th, sin(th));
    }

    return 0 ;
}
```

実行結果

- なんら問題はないように見える。



```
sin( -0.800000 ) = -0.717356
sin( -0.700000 ) = -0.644218
sin( -0.600000 ) = -0.564642
sin( -0.500000 ) = -0.479426
sin( -0.400000 ) = -0.389418
sin( -0.300000 ) = -0.295520
sin( -0.200000 ) = -0.198669
sin( -0.100000 ) = -0.099833
sin( 0.000000 ) = 0.000000
sin( 0.100000 ) = 0.099833
sin( 0.200000 ) = 0.198669
sin( 0.300000 ) = 0.295520
sin( 0.400000 ) = 0.389418
sin( 0.500000 ) = 0.479426
sin( 0.600000 ) = 0.564642
sin( 0.700000 ) = 0.644218
sin( 0.800000 ) = 0.717356
```

for の使い方 (8) 実数のループ変数

- `th = 0` の時だけ表示させたい。

```
/*      for_test3.c      */

#include <stdio.h>
#include <math.h>

int main( void ){
    double th ;

    for ( th = -3.1 ; th <= 3.1 ; th += 0.1 ) {
        if ( th == 0 ) {

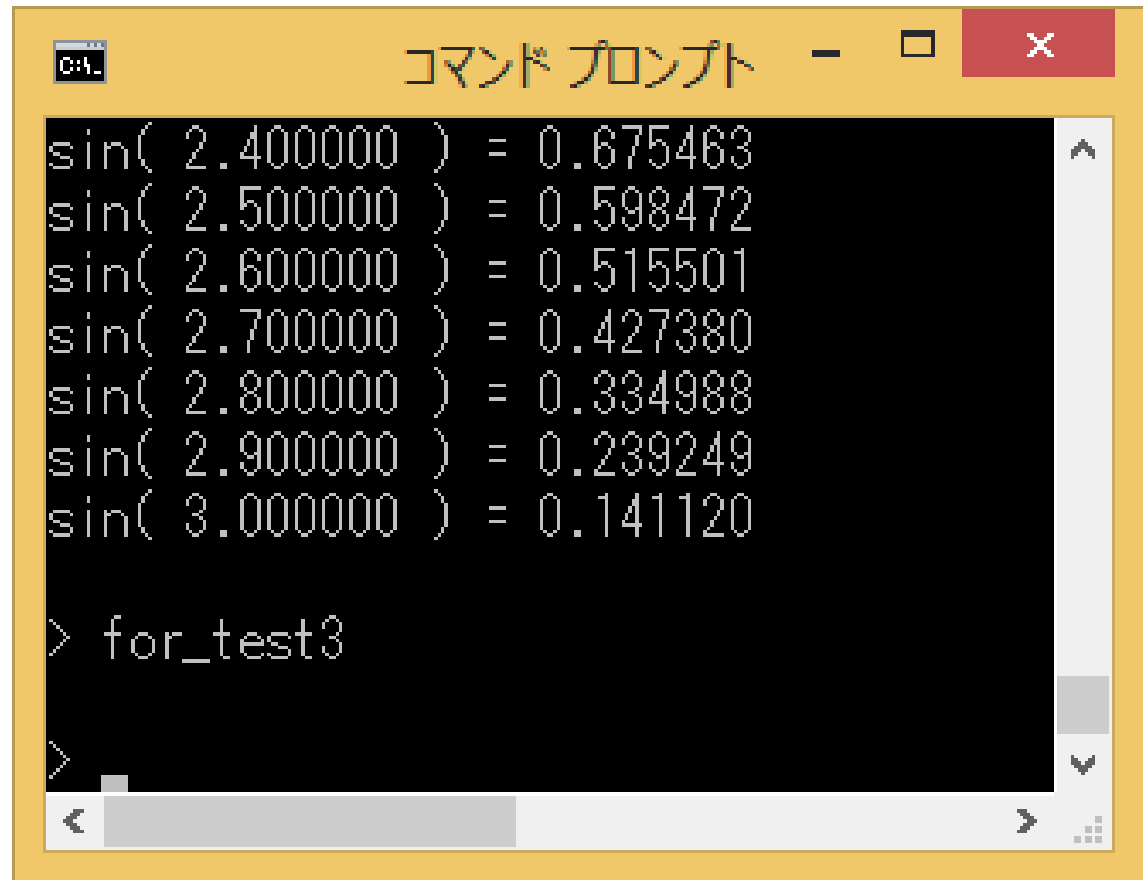
            printf("sin( %f ) = %f¥n", th, sin(th));

        }
    }

    return 0 ;
}
```

実行結果

- 何も表示されない！！
- 「th == 0 の場合はない」という結果になっている。
- 0に限らず、th == -1.0 などであっても同様



The screenshot shows a Windows Command Prompt window with a yellow title bar. The window contains the following text:

```
sin( 2.400000 ) = 0.675463
sin( 2.500000 ) = 0.598472
sin( 2.600000 ) = 0.515501
sin( 2.700000 ) = 0.427380
sin( 2.800000 ) = 0.334988
sin( 2.900000 ) = 0.239249
sin( 3.000000 ) = 0.141120

> for_test3
>
```

The window has a standard Windows interface with a maximize button, a close button (red X), and a scroll bar on the right side.

for の使い方 (6) 実数のループ変数

- `th = 0` の時だけ表示させたい。(改良1) ループカウンタを整数にする

```
/*      for_test3.c      */

#include <stdio.h>
#include <math.h>

int main( void ){

    double th ;
    int     i ;

    for ( i = -31 ; i <= 31 ; i++ ){

        th = i * 0.1 ;
        if ( th == 0 ) {

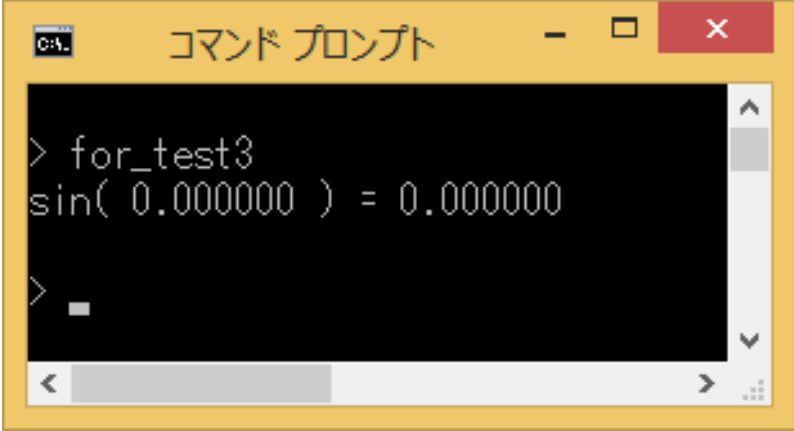
            printf("sin( %f ) = %f¥n", th, sin(th));

        }

    }

    return 0 ;

}
```



```
> for_test3
sin( 0.000000 ) = 0.000000
>
```


for の使い方 (8) 実数のループ変数

- $th = 0$ の時だけ表示させたい。(改良2) 判断を整数変数で行う

```
/*      for_test3.c      */

#include <stdio.h>
#include <math.h>

int main( void ){
    double th ;
    int     i ;

    for ( i = -31 ; i <= 31 ; i++ ) {
        th = i * 0.1 ;
        if ( i == 0 ) { 整数カウンタiで判断させる

                                printf("sin( %f ) = %f¥n", th, sin(th));

        }
    }

    return 0 ;
}
```

ループ変数は整数がbetter

- ループ変数を実数にした場合
 - ループ変数に誤差が累積する
 - 実数には丸め誤差が存在する。
 - コンピュータ内部では数値は2進数に変換されるので、丸め誤差が生じる。
 - 十進数でキリの良い数は2進数でキリがよいわけではない。
 - `if (th == 0)` のような実数ループ変数を使った条件判断は要注意。
 - 特に等号 (`==`)、不等号 (`!=`) を使う場合

【例題1】 キーボードから入力した整数 n が素数であるかを判定するプログラムを作成せよ。(for_test4.c)

```
/*      for_test4.c   nが素数であるかを判定するプログラム  */

#define   _CRT_SECURE_NO_WARNINGS   1
#include   <stdio.h>

int   main( void ){

    int     i, n ;
    int     count ;

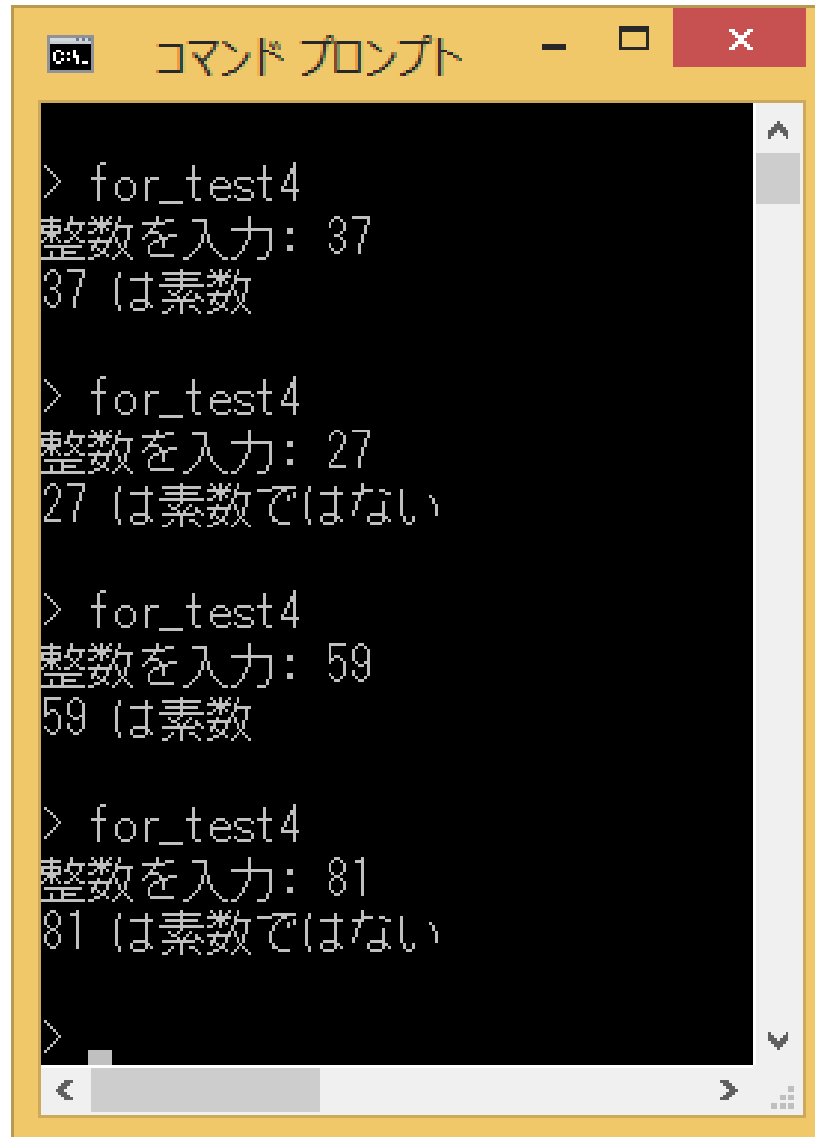
    ( この部分を作成する )

    return 0 ;
}
```

例題1 素数の考え方

- 入力した数 n (候補)
- 素数ではない
2 から $n - 1$ までの間に割り切れる数がある
- 素数
1 とそれ自身でしか割り切れない数
2 から $n - 1$ までの間で割り切れる数がない
よい (count == 0)

実行結果 素数の判定



```
コマンド プロンプト

> for_test4
整数を入力: 37
37 は素数

> for_test4
整数を入力: 27
27 は素数ではない

> for_test4
整数を入力: 59
59 は素数

> for_test4
整数を入力: 81
81 は素数ではない

>
```

While文

- while文は条件式を満たすまで繰り返し処理を行う
- 繰り返し回数が不明の場合でも使える

```
while( 条件式 ){  
    実行文;  
}
```

【例題2】 $n!$ (n の階乗)

- $n!$ の結果が1000を超えない最大の n を求めよ。
 - まず n を定数にして、 $n!$ を計算するプログラムを作成せよ。(繰り返し回数が n)
 - ループの終了条件が回数(n)でなく結果($n!$) であるプログラムを作成せよ。
 - `for` を使ったプログラム
 - `while` を使ったプログラム

while で作ってみる

```
/*      while1.c      */
#include <stdio.h>

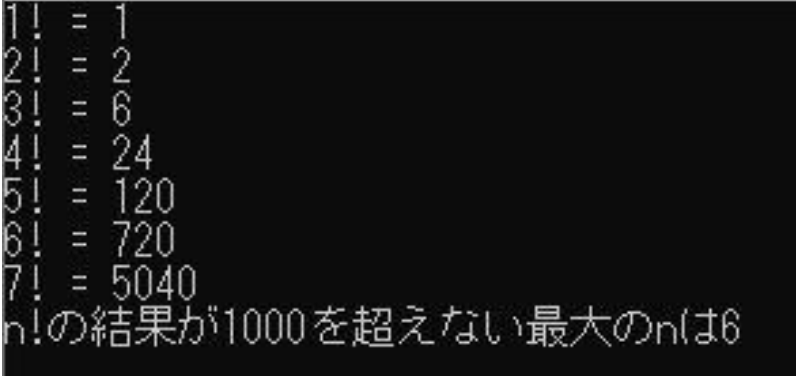
int main(void) {

    int    i, fac; // factorial: 階乗

    fac = 1;
    i = 1;

    (ここに while を使って作成)

    return 0;
}
```



```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
n!の結果が1000を超えない最大のnは6
```