

# ソフトウェア概論 A/B (2018/10/05)

Ver. 1.0

栗野 俊一

kurino@math.cst.nihon-u.ac.jp

<http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2018/soft/soft.html>

2018 年 10 月 5 日

## 概要

ソフトウェア概論 A/B<sup>1</sup> の 2018 年 10 月 5 日の資料<sup>2</sup>

## 目次

1	講義資料	1
1.1	当日の OHP 資料	1
1.2	Download	1
1.3	講義で利用するサンプルプログラム	1
1.4	講義中に作成したプログラム	22
1.5	本日の課題	22
1.5.1	課題 20181005-01 : キーボードから入力された二つの整数型の値の四則と余りの結果を表示する	22
1.5.2	課題 20181005-02 : キーボードから入力された二つの浮動小数点数型の四則の結果を表示する	24

---

<sup>1</sup><http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2018/soft/soft.html>

<sup>2</sup><http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2018/soft/20181005/20181005.html>

# 1 講義資料

## 1.1 当日の OHP 資料

- 当日の OHP 資料です。
  - 表示用 (HTML)<sup>3</sup>
  - 印刷用 (PDF)<sup>4</sup>

## 1.2 Download

- s\_random.h<sup>5</sup> : c:/usr/c/include にダウンロードする
- s\_csv.zip<sup>6</sup> : c:/usr/c/20181005 にダウンロードして、展開する。

## 1.3 講義で利用するサンプルプログラム

Download : sample-001.c<sup>7</sup>

リスト 1: sample-001.c

---

```
/*
 * 2018/10/05 sample-001.c
 */

/*
 * 型の違い
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-001.c
 *
 *          リンク
 *          cc -o sample-001.exe sample-001.c
 *
 *          実行
 *          ./sample-001.exe
 */

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
 *          main 関数
 */

int main ( void ) {
```

---

<sup>3</sup>ohp/html/index.html

<sup>4</sup>ohp/ohp.pdf

<sup>5</sup>files/s\_random.h

<sup>6</sup>files/s\_csv.zip

<sup>7</sup>program/sample-001.c

```

/*
 * 型が違ってても表示は同じ
 */

s_print_string ( "整数   : " );
s_print_int ( 9 );
s_print_newline();

s_print_string ( "文字   : " );
s_print_char ( '9' );
s_print_newline();

s_print_string ( "文字列 : " );
s_print_string ( "9" );
s_print_newline();

/*
 * 整数の場合の +1
 */

s_print_string ( "整数の計算 : 9 + 1 = " );
s_print_int ( 9 + 1 );
s_print_newline();

/*
 * 文字の場合の +1
 */

s_print_string ( "文字の計算 : '9' + 1 = " );
s_print_char ( '9' + 1 );
s_print_newline();

/*
 * 文字列の場合の +1
 */

s_print_string ( "文字の計算 : \"9\" + 1 = " );
s_print_string ( "9" + 1 );
s_print_newline();

/*
 * 型によって、計算結果が異なる
 */

return 0;
}

```

---

Download : [sample-002.c](#)<sup>8</sup>

リスト 2: [sample-002.c](#)

---

```

/*
 * 2018/10/05 sample-002.c
 */

```

---

<sup>8</sup>program/sample-002.c

```

$ ./sample-001.exe
整数   : 9
文字   : 9
文字列 : 9
整数の計算 : 9 + 1 = 10
文字の計算 : '9' + 1 = :
文字の計算 : "9" + 1 =
$

```

図 1: sample-001.c の実行結果

```

/*
 * 整数の四則計算式
 *
 * 使用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-002.c
 *          リンク
 *          cc -o sample-002.exe sample-002.c
 *          実行
 *          ./sample-002.exe
 */

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
 *      main 関数
 */

int main ( void ) {

    /*
     * 加法 (足し算)
     */

    s_print_string ( "12 + 5 = " );
    s_print_int ( 12 + 5 );
    s_print_newline();
    /* 整数の 12 と 5 を加える */

    /*
     * 減法 (引き算)
     */

    s_print_string ( "12 - 5 = " );
    s_print_int ( 12 - 5 );
    s_print_newline();
    /* 整数の 12 から 5 を引く */

    /*
     * 乗法 (かけ算)
     */

```

```

s_print_string ( "12 * 5 = " );
s_print_int ( 12 * 5 );
s_print_newline();
/* 整数の 12 に 5 をかける */

/*
 * 商法 (割り算)
 */

s_print_string ( "12 / 5 = " );
s_print_int ( 12 / 5 );
s_print_newline();
/* 整数の 12 を 5 で割る */

/*
 * 式の優先順位やかっこも利用可能
 */

s_print_string ( "1 + 2 * 3 = " );
s_print_int ( 1 + 2 * 3 );
s_print_newline();
/* かけ算が優先される */

s_print_string ( "(1 + 2) * 3 = " );
s_print_int ( (1 + 2) * 3 );
s_print_newline();
/* かっこの中の計算が優先される */

return 0;
}

```

---

```

$ ./sample-002.exe
12 + 5 = 17
12 - 5 = 7
12 * 5 = 60
12 / 5 = 2
1 + 2 * 3 = 7
(1 + 2) * 3 = 9
$

```

図 2: sample-002.c の実行結果

Download : [sample-003.c](#)<sup>9</sup>

リスト 3: sample-003.c

---

```

/*
 * 2018/10/05 sample-003.c
 */

/*
 * 文字列の演算式

```

---

<sup>9</sup>program/sample-003.c

```

*
* 利用方法
*           コンパイル
*           cc -I ~/c/include -c sample-003.c
*           リンク
*           cc -o sample-003.exe sample-003.c
*           実行
*           ./sample-003.exe
*/

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
*   main 関数
*/

int main ( void ) {

    /*
    *   文字列への加法
    */

    s_print_string ( "\"abc\" + 1 = " );
    s_print_string ( "abc" + 1 );
    s_print_newline();
    /* 文字列が短くなる */

    /*
    *   先頭の文字の取出し
    */

    s_print_string ( "*\"abc\" = " );
    s_print_char ( *"abc" );
    /* 先頭の文字が出
て来る */
    s_print_newline();

    return 0;
}

```

---

```

$ ./sample-003.exe
"abc" + 1 = bc
*"abc" = a
$

```

図 3: sample-003.c の実行結果

Download : [sample-004.c](#)<sup>10</sup>

リスト 4: [sample-004.c](#)

---

<sup>10</sup>program/sample-004.c

```

/*
 * 2018/10/05 sample-004.c
 */

/*
 * 関数呼出しを含む形の「式」
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-004.c
 *
 *          リンク
 *          cc -o sample-004.exe sample-004.c
 *
 *          実行
 *          ./sample-004.exe
 */

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
 *      main 関数
 */

int main ( void ) {

    /*
     * 加法 (足し算)
     */

    s_print_string ( "12 + 5 = " );
    s_print_int ( 12 + 5 );
    s_print_newline();
    /* 整数の 12 と 5 を加える */

    /*
     * 減法 (引き算)
     */

    s_print_string ( "12 - 5 = " );
    s_print_int ( 12 - 5 );
    s_print_newline();
    /* 整数の 12 から 5 を引く */

    /*
     * 乗法 (かけ算)
     */

    s_print_string ( "12 * 5 = " );
    s_print_int ( 12 * 5 );
    s_print_newline();
    /* 整数の 12 に 5 をかける */

    /*
     * 商法 (割り算)
     */

    s_print_string ( "12 / 5 = " );
    s_print_int ( 12 / 5 );
    s_print_newline();
    /* 整数の 12 を 5 で割る */
}

```

```
        return 0;
    }
```

---

```
$ ./sample-004.exe
12 + 5 = 17
12 - 5 = 7
12 * 5 = 60
12 / 5 = 2
$
```

図 4: sample-004.c の実行結果

Download : [sample-005.c](#)<sup>11</sup>

リスト 5: sample-005.c

---

```
/*
 * 2018/10/05 sample-005.c
 */

/*
 * 複数の正の整数の総和を計算する
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-005.c
 *
 *          リンク
 *          cc -o sample-005.exe sample-005.c
 *
 *          実行
 *          ./sample-005.exe
 */

#include <stdio.h>

#include "s_print.h"
#include "s_input.h"

/*
  正の整数をいくつかキーボードから入力し、
  その総和を出力する
  ただし、最後に、0 以下の数を入力する

  入力は、関数を呼び出す前に行われ、引数に渡される
  目的 : 入力(したデータ)を、
         終了のためのチェック
         加えるデータ
         の二ヶ所で利用したい
         入力したデータ二ヶ所で利用したいは関数に引数として
         わたして、その引数変数を経由して利用するパターンを使う
*/
```

---

<sup>11</sup>program/sample-005.c



(注意) 後で、「代入」を学ぶと、もっと簡単になる

引数 :

```
int pre_input
    関数を呼び出すまえに入力されたデータ
int total
    これまで、入力されたデータの総和が入っている
```

入力の例

```
1 2 3 4 -1
-> 結果の総和として 10 が表示される
```

\*/

```
void input_sum ( int pre_input, int total ) {

    if ( pre_input <= 0 ) {          /* 加えるデータはもうない */
        /* お仕舞いなので.. */
        s_print_string ( "総和は " );
        s_print_int ( total );
        s_print_string ( " です。 \n" );
    } else {                          /* 入力が正なので、まだやる可能性がある */
        input_sum ( s_input_int(), total + pre_input );
        /*
            total にはこれまでの和
            pre_input には、今回の入力
            なので
            ここまでの総和は
                total + pre_input
            になる
        */
    }
}

int main(void) {

    input_sum( s_input_int(), 0 );

    return 0;
}
```

---

12  
83  
40  
58  
42  
0

図 5: 入力例

```

$ ./sample-005.exe < sample-005.in
12
83
40
58
42
0
総和は 235 です。
$

```

図 6: sample-005.c の実行結果

Download : [sample-006.c](#)<sup>12</sup>

リスト 6: sample-006.c

---

```

/*
 * 2018/10/05 sample-006.c
 */

/*
 * rand 関数の利用例
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-006.c
 *
 *          リンク
 *          cc -o sample-006.exe sample-006.c
 *
 *          実行
 *          ./sample-006.exe
 */

#include <stdio.h>

#include "s_print.h"

void print_n_random ( int n ) {

    if ( n <= 0 ) {
    } else {

        /* ここから */
        s_print_int ( rand() );
        s_print_char ( '\n' );
        /* ここまでが、 n 回繰り返される */

        print_n_random ( n - 1 );
    }
}

int main(void) {
    print_n_random ( 10 );          /* 10 個の乱数を出力 */
}

```

<sup>12</sup>program/sample-006.c

```
    return 0;
}
```

---

```
$ ./sample-006.exe
1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
596516649
1189641421
$
```

図 7: sample-006.c の実行結果

Download : [sample-007.c](#)<sup>13</sup>

リスト 7: sample-007.c

---

```
/*
 * 2018/10/05 sample-007.c
 */

/*
 * srand による乱数系列の変更
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-007.c
 *
 *          リンク
 *          cc -o sample-007.exe sample-007.c
 *
 *          実行
 *          ./sample-007.exe
 */

#include <stdio.h>

#include "s_print.h"
#include "s_input.h"

int main(void) {

    s_print_string ( "乱数の seed(種) を入力してください : " );
```

---

<sup>13</sup>program/sample-007.c

```

    srand(s_input_int());      /* キーボードから seed を入力 */

    s_print_string ( "1回目 : " );
    s_print_int ( rand() );
    s_print_char ( '\n' );

    s_print_string ( "2回目 : " );
    s_print_int ( rand() );
    s_print_char ( '\n' );

    s_print_string ( "3回目 : " );
    s_print_int ( rand() );
    s_print_char ( '\n' );

    return 0;
}

```

---

1234

図 8: 入力例

```

$ ./sample-007.exe < sample-007.in
乱数の seed(種) を入力してください : 1234
1回目 : 479142414
2回目 : 465566339
3回目 : 961126155
$

```

図 9: sample-007.c の実行結果

Download : [sample-008.c](#)<sup>14</sup>

リスト 8: sample-008.c

---

```

/*
 * 2018/10/05 sample-008.c
 */

/*
 * 時刻による乱数系列の選択
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-008.c
 *
 *          リンク
 *          cc -o sample-008.exe sample-008.c

```

<sup>14</sup>program/sample-008.c

```

*           実行
*           ./sample-008.exe
*/

#include <stdio.h>

#include "s_print.h"
#include "s_input.h"

int main(void) {

    srand(time(NULL));          /* プログラムの実行開始時間を種にする */
    //srand(0);                /* プログラムをチェックする時は乱数列を固定 */

    s_print_string ( "1回目 : " );
    s_print_int ( rand() );
    s_print_char ( '\n' );

    s_print_string ( "2回目 : " );
    s_print_int ( rand() );
    s_print_char ( '\n' );

    s_print_string ( "3回目 : " );
    s_print_int ( rand() );
    s_print_char ( '\n' );

    return 0;
}

```

---

```

$ ./sample-008.exe
1回目 : 1460904650
2回目 : 1838669848
3回目 : 1045757348
$

```

図 10: sample-008.c の実行結果

Download : [sample-009.c](#)<sup>15</sup>

リスト 9: sample-009.c

---

```

/*
 * 2018/10/05 sample-009.c
 */

/*
 * 数当てゲーム
 *
 * 利用方法
 *           コンパイル

```

---

<sup>15</sup>program/sample-009.c

```

*           cc -I ~/c/include -c sample-009.c
*           リンク
*           cc -o sample-009.exe sample-009.c
*           実行
*           ./sample-009.exe
*/

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"
#include "s_random.h"           /* s_random.h を利用してみる */

/*
s_random.h によって、次の関数が利用可能になる
s_random()                rand() と同じ、整数の乱数を返す
s_init_random()           時刻を利用して、乱数系列を変更
s_n_random(N)             0 ~ (N-1) の間の整数の乱数を返す
s_random_a_b(A,B)        A ~ B の間の整数の乱数を返す
s_coin()                  0(裏) or 1(表) のどちらかを返す
s_dice()                  1 ~ 6 の間の整数の乱数を返す
*/

/*
プログラムが考えた数 ( 1 ~ 100 の乱数 ) を
人間が、予想して、当てるというゲームプログラム

入力する個数は、当るまでなので、何度も入力する必要がある
-> input loop を使う
問題を作るのに乱数を利用する
*/

void game( int input, int question ) {

    if ( input == question ) {           /* 予想が当たっている */
        s_print_string ( "おめでとう、ございます。当たりです\n" );
    } else {
        if ( input > question ) {
            s_print_string ( "大きすぎます。 \n" );
        } else {
            s_print_string ( "ちいさすぎます。 \n" );
        }

        s_print_string ( "もう一度予想してください : " );
        game( s_input_int(), question );
    }
}

/*
* main
*/

int main(int argc, char *argv[]) {

    s_init_random();                    /* 現在の時刻を利用して乱数を初期化*/
                                        /* 毎回異なる問題になる */

    s_print_string ( "コンピュータが考えた 1 ~ 100 の数を予想して入力してください : " );

```

```
game ( s_input_int(), s_random_a_b(1,100) );  
}
```

---

コンピュータが考えた 1 ~ 100 の数を予想して入力してください : 40  
40  
ちいさすぎます。  
もう一度予想してください : 60  
60  
ちいさすぎます。  
もう一度予想してください : 80  
80  
大きすぎます。  
もう一度予想してください : 70  
70  
大きすぎます。  
もう一度予想してください : 65  
65  
ちいさすぎます。  
もう一度予想してください : 68  
68  
大きすぎます。  
もう一度予想してください : 67  
67  
大きすぎます。  
もう一度予想してください : 66  
66  
おめでとう、ございます。当たりです

図 11: sample-009.c の実行結果

Download : [sample-010.c](#)<sup>16</sup>

リスト 10: sample-010.c

---

```
/*  
 * 2018/10/05 sample-010.c  
 */  
  
/*  
 * 数当て (逆)  
 */
```

---

<sup>16</sup>program/sample-010.c

```

* 利用方法
*
*           コンパイル
*           cc -I ~/c/include -c sample-010.c
*
*           リンク
*           cc -o sample-010.exe sample-010.c
*
*           実行
*           ./sample-010.exe
*/

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
sample-009.c とは逆に、
人間が数を考え、
コンピュータが当てる
*/

/*
考えかた
答えが、区間 [a,b] の中にあるとして、
答の案として  $x = (a+b)/2$  を言う
もし、 $x$  が大きい場合は、答えは
区間 [a,x-1] の中にある
逆に、そうでない場合は
区間 [x+1,b] の中にある
にある事がわかる
これをヒントに考える

ルール
コンピュータの予想が正しい時は 0
コンピュータの予想が大きいときは 1
コンピュータの予想が小さいときは 2
を答える
*/

void game ( int ans, int a, int b, int x ) {

if ( ans == 0 ) {           /* 当たり */
s_print_string ( "やっぱり " );
s_print_int ( x );
s_print_string ( " でしたか.. 当って嬉しいです\n" );
} else {
if ( ans == 1 ) {         /* 予想が大きい */
/* 答えは、区間 [a, x-1] にある.. */
s_print_string ( "私の予想は " );
s_print_int ( (a+(x-1))/2 );
s_print_string ( " です。いかがでしょうか : " );
game(s_input_int(), a, x-1, (a+(x-1))/2 );
} else { /* ans == 2 のはず.. 予想が小さい */
/* 答えは、区間 [x+1,b] にある.. */
s_print_string ( "私の予想は " );
s_print_int ( ((x+1)+b)/2 );
s_print_string ( " です。いかがでしょうか : " );
game(s_input_int(), x+1, b, ((x+1)+b)/2 );
}
}
}

```



```

    }
}

int main(void) {

    s_print_string ( "1 ~ 100 数を考えてください\n" );
    s_print_string ( "私が、その数を予想します\n" );

    s_print_string ( "私の予想は 50 です。いかがでしょうか : " );

    game ( s_input_int(), 1, 100, 50 );

}

```

---

```

1
2
1
0

```

図 12: 入力例

```

$ ./sample-010.exe < sample-010.in
1 ~ 100 数を考えてください
私が、その数を予想します
私の予想は 50 です。いかがでしょうか : 1
私の予想は 25 です。いかがでしょうか : 2
私の予想は 37 です。いかがでしょうか : 1
私の予想は 31 です。いかがでしょうか : 0
やっぱり 31 でしたか.. 当たって嬉しいです
$

```

図 13: sample-010.c の実行結果

Download : [sample-011.c](#)<sup>17</sup>

リスト 11: sample-011.c

---

```

/*
 * 2018/10/05 sample-011.c
 */

```

---

<sup>17</sup>program/sample-011.c

```

/*
 * 2 のべき乗を順に n 個出力する
 *
 * 利用方法
 *          コンパイル
 *          cc -I ~/c/include -c sample-011.c
 *          リンク
 *          cc -o sample-011.exe sample-011.c
 *          実行
 *          ./sample-011.exe
 */

#include <stdio.h>
#include "s_print.h"
#include "s_input.h"

/*
 * 2 のべき乗を順に n 個出力する
 */

void power ( int n, int b ) {

    if ( n <= 0 ) {

    } else {
        s_print_int ( b );
        s_print_char ( '\n' );
        power ( n - 1, b * 2 );
    }
}

/*
 * main
 */

int main( int argc, char *argv[] ) {

    power ( s_input_int(), 2 );

    return 0;
}

```

---

10

図 14: 入力例

Download : [sample-012.c](#)<sup>18</sup>

リスト 12: [sample-012.c](#)

---

<sup>18</sup>program/sample-012.c

```

$ ./sample-011.exe < sample-011.in
10
2
4
8
16
32
64
128
256
512
1024
$

```

図 15: sample-011.c の実行結果

```

/*
 * 2018/10/05 sample-012.c
 */

/*
 * 浮動小数点数の出力
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-012.c
 *
 *          リンク
 *          cc -o sample-012.exe sample-012.c
 *
 *          実行
 *          ./sample-012.exe
 */

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

int main(void) {

    s_print_double ( 1.234 );          /* 小数点を含む数が扱える */
    s_print_char ( '\n' );

    return 0;
}

```

---

Download : [sample-013.c](#)<sup>19</sup>

リスト 13: [sample-013.c](#)

---

<sup>19</sup>program/sample-013.c

```
$ ./sample-012.exe
1.234000
$
```

図 16: sample-012.c の実行結果

```
/*
 * 2018/10/05 sample-013.c
 */

/*
 * 浮動小数点数の入力
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-013.c
 *
 *          リンク
 *          cc -o sample-013.exe sample-013.c
 *
 *          実行
 *          ./sample-013.exe
 */

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

int main(void) {

    s_print_double ( s_input_double() );
    /* 小数点を含む数を入力して、そのまま出力 */
    /* 整数の時には s_print_int ( s_input_int() ); だった */
    s_print_char ( '\n' );

    return 0;
}
```

---

---

12.34

図 17: 入力例

Download : [sample-014.c](#)<sup>20</sup>

リスト 14: [sample-014.c](#)

---

---

<sup>20</sup>[program/sample-014.c](#)

```
$ ./sample-013.exe < sample-013.in
12.340000
12.340000
$
```

図 18: sample-013.c の実行結果

```
/*
 * 2018/10/05 sample-014.c
 */

/*
 * 浮動小数点数の計算
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-014.c
 *          リンク
 *          cc -o sample-014.exe sample-014.c
 *          実行
 *          ./sample-014.exe
 */

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
 * main
 */

int main( int argc, char *argv[] ) {

    s_print_double ( s_input_double() * 1.08 );
    /* 小数点を含む数を入力して、消費税込みの値段に.. */
    s_print_char ( '\n' );

    return 0;
}



---



---


```

123

図 19: 入力例

Download : [sample-015.c](#)<sup>21</sup>

リスト 15: [sample-015.c](#)

---

---

<sup>21</sup>[program/sample-015.c](#)

```

$ ./sample-014.exe < sample-014.in
123.000000
132.840000
$

```

図 20: sample-014.c の実行結果

```

/*
 * 2018/10/05 sample-015.c
 */

/*
 * s_random.h の利用
 *
 * 利用方法
 *
 *          コンパイル
 *          cc -I ~/c/include -c sample-015.c
 *
 *          リンク
 *          cc -o sample-015.exe sample-015.c
 *
 *          実行
 *          ./sample-015.exe
 */

#include <stdio.h>

#include "s_print.h"
#include "s_input.h"
#include "s_random.h"

/*
 * s_random.h によって、次の関数が利用可能になる
 *
 *          s_random()          rand() と同じ、整数の乱数を返す
 *          s_init_random()     時刻を利用して、乱数系列を変更
 *          s_n_random(N)       0 ~ (N-1) の間の整数の乱数を返す
 *          s_random_a_b(A,B)   A ~ B の間の整数の乱数を返す
 *          s_coin()            0(裏) or 1(表) のどちらかを返す
 *          s_dice()            1 ~ 6 の間の整数の乱数を返す
 */

/*
 * main
 */

int main(int argc, char *argv[]) {

    s_init_random();          /* 時計を使って、乱数列を初期化 */

    s_print_string ( "1回目 : " );
    s_print_int ( s_dice() );          /* サイコロの結果 */
    s_print_char ( '\n' );

    s_print_string ( "2回目 : " );
    s_print_int ( s_dice() );
    s_print_char ( '\n' );
}

```

```

s_print_string ( "3回目 : " );
s_print_int ( s_dice() );
s_print_char ( '\n' );

return 0;
}

```

---

```

$ ./sample-015.exe
1回目 : 3
2回目 : 5
3回目 : 3
$

```

図 21: sample-015.c の実行結果

## 1.4 講義中に作成したプログラム

- 講義中に作成したプログラム<sup>22</sup>

## 1.5 本日の課題

課題プログラム内の「/\*名前:ここ\*/」の部分を書き換え「/\*この部分を完成させなさい\*/」の部分にプログラムを追加して、プログラムを完成させます。

### 1.5.1 課題 20181005-01 : キーボードから入力された二つの整数型の値の四則と余りの結果を表示する

Download : 20181005-01.c<sup>23</sup>

リスト 16: 20181005-01.c

---

```

/*
 * 20181005-01-QQQQ.c
 *
 *     二つの整数値をキーボードから入力し、その四則並びに余りを出力する
 *
 *     コンパイル :
 *             cc -I ~/c/include -c 20181005-01-QQQQ.c
 *             cc -o 20181005-01-QQQQ.exe 20181005-01-QQQQ.o
 *     実行 :
 *             ./20181005-01-QQQQ.exe
 *
 */

```

---

<sup>22</sup>p/

<sup>23</sup>ex/01/q/01.c

```

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
 * print_int_result
 */

void print_int_result ( char *name, int a, int b, char op, int value ) {

    s_print_string ( name );
    s_print_string ( " : " );
    s_print_int ( a );

    /*
     **      この部分を完成させなさい
     */

    s_print_char ( '=' );
    s_print_int ( value );
    s_print_newline();
}

/*
 *      print_int_calc
 */

void print_int_calc ( int a, int b ) {

    print_int_result ( "和", a, b, '+', a + b );
    print_int_result ( "差", a, b, '-', a - b );

    /*
     **      この部分を完成させなさい
     */

    print_int_result ( "余り", a, b, '%', a % b );

}

/*
 *      print_int_calc_1
 */

void print_int_calc_1 ( int a ) {

    /* 二つ目の数の入力と、関数呼出し */
    print_int_calc ( a, s_input_int() );
}

/*
 *      main
 */

int main( int argc, char *argv[] )
{

```



```

/*
 *      複数の入力を伴うので、入力の順序を保証するために、
 *      入力してから関数を呼ぶ
 */

      /* 一つ目の数の入力と、関数呼出し */
print_int_calc_1 ( s_input_int() );

return 0;
}

```

---

```

175
51

```

図 22: 入力例

```

$ ./20181005-01-QQQQ.exe
175
51
和 : 175+51=226
差 : 175-51=124
積 : 175*51=8925
商 : 175/51=3
余り : 175%51=22
$

```

図 23: 20181005-01.c の実行結果

### 1.5.2 課題 20181005-02 : キーボードから入力された二つの浮動小数点数型の四則の結果を表示する

Download : [20181005-02.c](#)<sup>24</sup>

リスト 17: [20181005-02.c](#)

---

```

/*
 * 20181005-02-QQQQ.c
 *
 *      二つの浮動小数点数値をキーボードから入力し、その四則を出力する
 *
 *      コンパイル :
 *      cc -I ~/c/include -c 20181005-02-QQQQ.c

```

<sup>24</sup>[ex/02/q/02.c](#)

```

*          cc -o 20181005-02-QQQQ.exe 20181005-02-QQQQ.o
*      実行 :
*          ./20181005-02-QQQQ.exe
*
*/

#include <stdio.h>

#include "s_input.h"
#include "s_print.h"

/*
* print_double_result
*/

void print_double_result ( char *name, double a, double b, char op, double value ) {

    s_print_string ( name );
    s_print_string ( " : " );
    s_print_double ( a );

    /*
    **      この部分を完成させなさい
    */

    s_print_char ( '=' );
    s_print_double ( value );
    s_print_newline();
}

/*
*      print_double_calc
*/

void print_double_calc ( double a, double b ) {

    print_double_result ( "和", a, b, '+', a + b );
    print_double_result ( "差", a, b, '-', a - b );

    /*
    **      この部分を完成させなさい
    */

}

/*
*      print_double_calc_1
*/

void print_double_calc_1 ( double a ) {

    /* 二つ目の数の入力と、関数呼出し */
    print_double_calc ( a, s_input_double() );
}

/*
*      main

```

```

*/
int main( int argc, char *argv[] )
{
    /*
    *     複数の入力を伴うので、入力の順序を保証するために、
    *   入力してから関数を呼ぶ
    */

    /* 一つ目の数の入力と、関数呼出し */
    print_double_calc_1 ( s_input_double() );

    return 0;
}

```

---

```

123.456
789.012

```

図 24: 入力例

```

$ ./20181005-02-QQQQ.exe
123.456000
789.012000
和 : 123.456000+789.012000=912.468000
差 : 123.456000-789.012000=-665.556000
積 : 123.456000*789.012000=97408.265472
商 : 123.456000/789.012000=0.156469
$

```

図 25: 20181005-02.c の実行結果