ソフトウェア概論 A/B (2018/10/19)

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 19 日

概 要

ソフトウェア概論 A/B[1] の 2018 年 10 月 19 日 の資料 [2]

目 次

[1] http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2018/soft/soft.html
[2] http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2018/soft/20181019/20181019.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)[3]
  - 印刷用 (PDF)[4]

1.2 講義で利用するサンプルプログラム

Download : sample-001.c[5]

リスト 1: sample-001.c

```
/*
 * 2018/10/19 sample-001.c
 */

/*
 *  Hello, World
 *
 *        いつでも、始まりはこれから..
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-001.exe sample-001.c
 *                実行
 *                        sample-001
 */

#include <stdio.h>

/*
 *        main
 */

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

        printf ( "Hello, World\n" );

        return 0;
}
```

Download : sample-002.c[6]

リスト 2: sample-002.c

[3] ohp/html/index.html
[4] ohp/ohp.pdf
[5] program/sample-001.c
[6] program/sample-002.c

```
$ ./sample-001.exe
Hello, World
$
```

図 1: sample-001.c の実行結果

```
/*
 * 2018/10/19 sample-002.c
 */

/*
 * n^2 の計算を足し算だけで実現する
 *
 * 利用方法
 *                 コンパイル
 *                         cc -Ic:\usr\c\include -o sample-002.exe sample-002.c
 *                 実行
 *                         sample-002
 */

#include <stdio.h>

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

/*
 * 四角数 http://ja.wikipedia.org/wiki/%E5%B9%B3%E6%96%B9%E6%95%B0
 */

/*
 * main
 */

int main ( int argc, char *argv[] ) {
        int n = 13;                /* n=13 として n^2 = n*n = 13*13 = 169 を計算する */
        int i = 0;                /* i は、0 から n-1 まで動き、足し算の回数を示す */
        int sum = 0;        /* sum には、個々の odd の値が足しこまれる。最初は 0 */
        int odd = 1;        /* odd は、最初は 1 で、以後 2 ずつふえる（奇数） */

        while ( i < n ) {                /* i が n より小さい間 */
                sum = sum + odd;        /* sum に現在の奇数を加える */
                odd = odd + 2;                /* 次の奇数は、今の奇数に 2 を加えればよい */
                i = i + 1;                        /* 何回加えたかを計数し、繰返し回数を確認
する */
        }

        s_print_int ( sum );        /* 加えた結果が表示される n^2 になっているはず.. */
        s_print_newline();

        return 0;
}
```

```
$ ./sample-002.exe
169
$
```

図 2: sample-002.c の実行結果

Download : sample-003.c[7]

リスト 3: sample-003.c

```
/*
 * 2018/10/19 sample-003.c
 */

/*
 *  printf の書式付き出力
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-003.exe sample-003.c
 *                実行
 *                        sample-003
 */

#include <stdio.h>

/*
 *        main
 */

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

        printf ( "5 + 4 = %d\n", 5 + 4 );
                /* 文字列の中の「%d」の所に 9 ( = 5 + 4 ) が入る */

        return 0;
}
```

```
$ ./sample-003.exe
5 + 4 = 9
$
```

図 3: sample-003.c の実行結果

Download : sample-004.c[8]

リスト 4: sample-004.c

[7] program/sample-003.c
[8] program/sample-004.c

```
/*
 * 2018/10/19 sample-004.c
 */

/*
 *  printf の書式付き出力 (その他色々)
 *
 * 利用方法
 *                 コンパイル
 *                         cc -Ic:\usr\c\include -o sample-004.exe sample-004.c
 *                 実行
 *                         sample-004
 */

#include <stdio.h>

/*
 *        main
 */

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

        printf ( "%d + %d = %d\n", 5, 4,  5 + 4 );
                /*
                                文字列の中の「%d」の所に
                                順番に数値を表す文字列が埋め込まれる */

        return 0;
}
```

```
$ ./sample-004.exe
5 + 4 = 9
$
```

図 4: sample-004.c の実行結果

Download : sample-005.c[9]

リスト 5: sample-005.c

```
/*
 * 2018/10/19 sample-005.c
 */

/*
 *  printf の書式付き出力 (その他色々)
 *
 * 利用方法
 *                 コンパイル
 *                         cc -Ic:\usr\c\include -o sample-005.exe sample-005.c
```

[9] program/sample-005.c

```
 *                実行
 *                        sample-005
 */

#include <stdio.h>

/*
 *        main
 */

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

        printf ( "整数型の出力は %%d を使って「%d」と出せる。\n",
                                123 );

        printf ( "浮動小数点数型の出力は %%f を使って「%f」と出せる。\n",
                                123.456 );

        printf ( "文字型の出力は %%c を使って「%c」と出せる。\n",
                                'Z' );

        printf ( "文字列の出力は %%s を使って「%s」と出せる。\n",
                                "pqr" );

        return 0;
}
```

```
$ ./sample-005.exe
整数型の出力は %d を使って「123」と出せる。
浮動小数点数型の出力は %f を使って「123.456000」と出せる。
文字型の出力は %c を使って「Z」と出せる。
文字列の出力は %s を使って「pqr」と出せる。
$
```

図 5: sample-005.c の実行結果

Download : sample-006.c[10]

リスト 6: sample-006.c

```
/*
 * 2018/10/19 sample-006.c
 */

/*
 *  printf の書式付き出力 (その他色々)
 *
 * 利用方法
 *                コンパイル
```

[10] program/sample-006.c

```
 *                      cc -Ic:\usr\c\include -o sample-006.exe sample-006.c
 *              実行
 *                      sample-006
 */

#include <stdio.h>

/*
 *      main
 */

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

        printf ( "書式は、色々な形の物のを混在させても良い。\n" );
        printf ( "%d, %f, %c, %s\n", 10, 2.3, 'x', "abc" );

        return 0;
}
```

```
$ ./sample-006.exe
書式は、色々な形の物のを混在させても良い。
10, 2.300000, x, abc
$
```

図 6: sample-006.c の実行結果

Download : sample-007.c[11]

リスト 7: sample-007.c

```
/*
 * 2018/10/19 sample-007.c
 */

/*
 *  scanf の書式付き入力
 *
 * 利用方法
 *              コンパイル
 *                      cc -Ic:\usr\c\include -o sample-007.exe sample-007.c
 *              実行
 *                      sample-007
 */

#include <stdio.h>

/*
 *      main
 */
```

[11] program/sample-007.c

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

        printf ( "整数値を入力してください : " );

        scanf ( "%d", &i );
                        /* 整数型なので「%d」の書式を付ける */
                        /* 変数 i の前に「&」を付ける (今回は「お呪い」考えてください)*/

        printf ( "整数値 %d が入力されました。\n", i );

        return 0;
}
```

```
123
```

図 7: 入力例

```
$ ./sample-007.exe < sample-007.in
整数値を入力してください : 123
整数値 123 が入力されました。
$
```

図 8: sample-007.c の実行結果

Download : sample-008.c[12]

リスト 8: sample-008.c

```
/*
 * 2018/10/19 sample-008.c
 */

/*
 *  scanf の書式付き入力 (色々)
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-008.exe sample-008.c
 *                実行
 *                        sample-008
 */

#include <stdio.h>
```

[12] program/sample-008.c

```
/*
 *        main
 */

int main( int argc, char *argv[] )
{
        int i;
        double x;

                /* 整数 */
        printf ( "整数値を入力してください : " );

        scanf ( "%d", &i );
                        /* 整数型なので「%d」の書式を付ける */
                        /* 変数 i の前に「&」を付ける (今回は「お呪い」と考える)*/

        printf ( "整数値 %d が入力されました。\n", i );

                /* 浮動小数点数 */
        printf ( "浮動小数点数値を入力してください : " );

        scanf ( "%lf", &x );
                        /* 浮動小数点数型なので「%lf」の書式を付ける */
                        /* 変数 x の前に「&」を付ける (今回は「お呪い」と考える)*/

        printf ( "浮動小数点数値 %f が入力されました。\n", x );

        return 0;
}
```

```
123
456.789
```

図 9: 入力例

```
$ ./sample-008.exe < sample-008.in
整数値を入力してください : 123
整数値 123 が入力されました。
浮動小数点数値を入力してください : 456.789000
浮動小数点数値 456.789000 が入力されました。
$
```

図 10: sample-008.c の実行結果

Download : sample-009.c[13]

[13] program/sample-009.c

リスト 9: sample-009.c

```
/*
 * 2018/10/19 sample-009.c
 */

/*
 *  一文字鸚鵡返しをするプログラム
 *                  キーボードから文字を入力するには、「Enter」を押す必要がある
 *                  # つまり、「'A' の一文字」を入力する場合も「'A'」と [ENTER] のニ文字の入力が必
要って事
 *                  # 厄介な事に、もちろん [ENTER] も一文字になる。
 *
 * 利用方法
 *                  コンパイル
 *                          cc -Ic:\usr\c\include -o sample-009.exe sample-009.c
 *                  実行
 *                          sample-009
 */

#include <stdio.h>

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        int ch;                 /* 文字を扱うのだが、都合により整数型 (int 型) を使う */

        ch = getchar();         /* 一文字入力する */
        putchar ( ch );         /* その文字を出力する */

        return 0;
}
```

```
A
```

図 11: 入力例

```
$ ./sample-009.exe < sample-009.in
A
A$
```

図 12: sample-009.c の実行結果

Download : sample-010.c[14]

リスト 10: sample-010.c

[14] program/sample-010.c

```
/*
 * 2018/10/19 sample-010.c
 */

/*
 *  一文字と改行を鸚鵡返しをするプログラム
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-010.exe sample-010.c
 *                実行
 *                        sample-010
 */

#include <stdio.h>

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        int ch;                 /* 文字を扱うのだが、都合により整数型 (int 型) を使う */

        ch = getchar();         /* 一文字入力する */
        putchar ( ch );         /* その文字を出力する */

        ch = getchar(); /* ニ文字目 (恐らく [Enter]) を入力する */
        putchar ( ch );         /* その文字を出力する */

        return 0;
}
```

```
A
```

図 13: 入力例

```
$ ./sample-010.exe < sample-010.in
A
A
$
```

図 14: sample-010.c の実行結果

Download : sample-011.c[15]

リスト 11: sample-011.c

[15] program/sample-011.c

```
/*
 * 2018/10/19 sample-011.c
 */

/*
 *  「終わり」のマークが来るまで鸚鵡返し
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-011.exe sample-011.c
 *                実行
 *                        sample-011
 */

#include <stdio.h>

/*
 *        main
 */

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

        ch = getchar();                 /* 最初の一文字入力する */

        while ( ch != EOF )         {         /* 「入力」が EOF でなければ.. 以下を繰り返す */
                        /* EOF は「End Of File(ファイルの終わり)」を示す */
                        /* EOF は文字 (コード) ではなく、整数型 */
                        /* ch を整数型 (int 型) にしたのは EOF が入る可能性があるので.. */
                        /* 間違えて ch を char で宣言すると EOF の判定ができない */

                putchar ( ch );         /* 入力された文字を出力する */

                ch = getchar();         /* 次の文字を入力する (EOF の可能性もある..) */
        }

        /* 「Windows」で、「キーボード」から「EOF」を入力する場合は
                Ctrol-Z ( ^Z : [Ctrl] キーを押しながら 「Z」をポンと押す)
                を入力する ( OS や入力先により、操作が異る事が多い .. )
        */

        return 0;
}
```

```
This is first line.
And, it is second line.
Last, it is third line.
```

図 15: 入力例

```
$ ./sample-011.exe < sample-011.in
This is first line.
This is first line.
And, it is second line.
And, it is second line.
Last, it is third line.
Last, it is third line.
$
```

図 16: sample-011.c の実行結果

Download : sample-012.c[16]

リスト 12: sample-012.c

```
/*
 * 2018/10/19 sample-012.c
 */

/*
 *  鸚鵡返しの fgtec, fputc 版
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-012.exe sample-012.c
 *                実行
 *                        sample-012
 */

#include <stdio.h>

/*
 *        main
 */

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

        ch = fgetc ( stdin );                /* 入力元を stdin (標準入力) にする */

        while ( ch != EOF )        {                /* 「入力データ」が EOF になるまで */

                fputc ( ch, stdout );        /* 出力先を stdout (標準出力) にする */

                ch = fgetc ( stdin );
        }

        return 0;
}
```

[16]program/sample-012.c

```
This is first line.
And, it is second line.
Last, it is third line.
```

図 17: 入力例

```
$ ./sample-012.exe < sample-012.in
This is first line.
And, it is second line.
Last, it is third line.
$
```

図 18: sample-012.c の実行結果

Download : sample-013.c[17]

リスト 13: sample-013.c

```
/*
 * 2018/10/19 sample-013.c
 */

/*
 *  stdout と stderr の違い
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-013.exe sample-013.c
 *                実行
 *                        sample-013
 */

#include <stdio.h>

/*
 *        main
 */

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

        printf ( "O: これは標準出力\n" );
                /* printf は標準出力に出す */

        fprintf ( stderr, "E: これは標準エラー出力\n" );
                /* 標準エラー出力に出す場合は、fprintf を用い、stderr を指定する */

        fprintf ( stdout, "O: これも標準出力\n" );
                /* fprintf で stdout を指定すれば、標準出力に出せる */

        fprintf ( stderr, "E: 再び標準エラー出力\n" );
```

[17] program/sample-013.c

```
        /*
                だまっていると、混在して画面に表示される
                標準出力をリダイレクトすると、この二つが分離できる
        */

        return 0;
}
```

```
O: これは標準出力
E: これは標準エラー出力
O: これも標準出力
E: 再び標準エラー出力
```

図 19: sample-013.c の実行結果

Download : sample-014.c[18]

リスト 14: sample-014.c

```
/*
 * 2018/10/19 sample-014.c
 */

/*
 *  fopen の利用
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-014.exe sample-014.c
 *                実行
 *                        sample-014
 */

#include <stdio.h>

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        FILE        *ifp;                /* ファイルポインタ変数の宣言 */

        ifp = fopen ( "sample-014.c", "r" );
                                                /* sample-014.c を 読み出しモードで開く */

        if ( ifp == NULL ) {        /* Open に失敗した場合は.. */
                fprintf ( stderr, "ファイルのオープンに失敗しました。\n" );
                                                /* エラーメッセージを出力 */
```

[18]program/sample-014.c

```
                /* 後は何もしない (クローズもしない) */
        } else {                                /* Open に成功した場合のみ利用してよい */
                int ch;

                ch = fgetc ( ifp );                /* fgetc で、ファイルから読込み */

                while ( ch != EOF )        {        /* ファイルからでも EOF は「入力終わり」
*/
                        putchar ( ch );                /* 結果は標準出力へ.. */
                        ch = fgetc ( ifp );
                }

                fclose ( ifp );        /* 最後に、オープンできた場合はクローズする */
        }

        return 0;
}
```

Download : sample-015.c[19]

リスト 15: sample-015.c

```
/*
 * 2018/10/19 sample-015.c
 */

/*
 *  fopen の利用
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-015.exe sample-015.c
 *                実行
 *                        sample-015
 */

#include <stdio.h>

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        FILE        *ofp;                /* ファイルポインタ変数の宣言 */

        ofp = fopen ( "sample-015.txt", "w" );
                        /* sample-015.txt を 読み出しモードで開く */
                        /* 元々あった sample-015.txt の内容は上書きされる事に注意 */

        if ( ofp == NULL ) {        /* Open に失敗した場合は.. */
                fprintf ( stderr, "ファイルのオープンに失敗しました。\n" );
                                                /* エラーメッセージを出力 */
```

[19] program/sample-015.c

```
$ ./sample-014.exe
/*
 * 2018/10/19 sample-014.c
 */

/*
 *  fopen の利用
 *
 * 利用方法
 *                コンパイル
 *                        cc -Ic:\usr\c\include -o sample-014.exe sample-014.c
 *                実行
 *                        sample-014
 */

#include <stdio.h>

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        FILE        *ifp;               /* ファイルポインタ変数の宣言 */

        ifp = fopen ( "sample-014.c", "r" );
                                                /* sample-014.c を 読み出しモード
で開く */

        if ( ifp == NULL ) {        /* Open に失敗した場合は.. */
                fprintf ( stderr, "ファイルのオープンに失敗しました。\n" );
                                                /* エラーメッセージを出力 */

                /* 後は何もしない (クローズもしない) */
        } else {                                /* Open に成功した場合のみ利用して
よい */
                int ch;

                ch = fgetc ( ifp );               /* fgetc で、ファイルから読込
み */

                while ( ch != EOF )        {        /* ファイルからでも EOF は「入
力終わり」*/
                        putchar ( ch );                /* 結果は標準出力へ.. */
                        ch = fgetc ( ifp );
                }

                fclose ( ifp );         /* 最後に、オープンできた場合はクローズする */
        }
```

```
                /* 後は何もしない (クローズもしない) */
        } else {                        /* Open に成功した場合のみ利用してよい */
                int ch;

                ch = getchar ();                /* キーボード入力 */

                while ( ch != EOF )         {
                        fputc ( ch, ofp );              /* 結果はファイルへ.. */
                        ch = getchar ();
                }

                fclose ( ofp );         /* 最後に、オープンできた場合はクローズする */

                /*
                        結果は、画面でなく、sample-015.txt に保存される
                */

        }


        return 0;
}
```

```
Fist Line
Sencod Line
Last Line
```

図 21: 入力例

```
$ ./sample-015.exe < sample-015.in
Fist Line
Sencod Line
Last Line
$
```

図 22: sample-015.c の実行結果

1.3 講議中に作成したプログラム

- 講議中に作成したプログラム [20]

[20] p/

1.4 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。

1.4.1 課題 20181019-01 : printf

Download : 20181019-01.c[21]

リスト 16: 20181019-01.c

```
/*
 * 課題 CNAME-01
 *
 * 20181019 20181019-01-QQQQ.c
 *
 *        printf の書式指定
 *
 */

#include <stdio.h>

/*
 *
 */

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        int ia = 7;
        int ib = 15;

        double da = 3.14;
        double db = 1.59;

        printf ( "/* o:ここ */ + /* o:ここ */ = /* o:ここ */\n", /* p:ここ */, /* q:ここ */, /* p:
ここ */ + /* q:ここ */ );        /* 整数型の和 */
        printf ( "/* r:ここ */ / /* r:ここ */ = /* r:ここ */\n", /* s:ここ */, /* u:ここ */, /* s:
ここ */ / /* u:ここ */ );        /* 浮動小数点数の商 */

        return 0;
}
```

```
abc123
```

図 23: 入力例

[21] ex/01/q/01.c

```
$ ./20181019-01-QQQQ.exe
7 + 15 = 22
3.140000 / 1.590000 = 1.974843
$
```

図 24: 20181019-01.c の実行結果

1.4.2 課題 20181019-02 : scanf

Download : 20181019-02.c[22]

リスト 17: 20181019-02.c

```
/*
 * 課題 CNAME-02
 *
 * 20181019 20181019-02-QQQQ.c
 *
 *        scanf の書式指定
 *
 */

#include <stdio.h>

/*
 *
 */

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        int i;                  /* キーボードから入力された整数値を保持する整数型変数 */
        double d;        /* キーボードから入力された浮動小数点数値を保持する浮動小数点数型変
数 */

        printf ( "整数値をキーボードから入力します : " );
        scanf ( "/* p:ここ */", /* q:ここ */ );
        printf ( "入力された整数値は %d でした。\n", i );

        printf ( "浮動小数点数値をキーボードから入力します : " );
        scanf ( "/* r:ここ */", /* s:ここ */ );
        printf ( "入力された浮動小数点数値は %f でした。\n", d );

        return 0;
}
```

[22] ex/02/q/02.c

```
123
456.789
```

図 25: 入力例

```
$ ./20181019-02-QQQQ.exe
整数値をキーボードから入力します : 123
入力された整数値は 123 でした。
浮動小数点数値をキーボードから入力します : 456.789000
入力された浮動小数点数値は 456.789000 でした。
$
```

図 26: 20181019-02.c の実行結果

1.4.3 課題 20181019-03 : 変数宣言と代入文

Download : 20181019-03.c[23]

リスト 18: 20181019-03.c

```
/*
 * 課題 CNAME-03
 *
 * CDATE FILENAME
 *
 *        代入文
 *                整数型変数 ( a, b, c ) を三つ宣言する
 *                整数型変数 a, b にそれぞれ、123, 4 を代入する
 *                整数型変数 c に a と b の和を代入する
 *                整数型変数 c の値を画面に出力する
 */

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

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        /* 整数型変数 a の宣言 */
        int        a;
        /* 整数型変数 b の宣言 */

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

        /* 整数型変数 c の宣言 */
        int        c;
```

[23] ex/03/q/03.c

```
        /* 整数型変数 a に 123 を代入 */
        a = 123;

        /* 整数型変数 b に 4 を代入 */

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


        /* 整数型変数 c に、変数 a と 変数 b の和を代入 */

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


        /* 整数型変数 c の値を画面に出力 */
        s_print_string ( "変数 c の値は " );
        s_print_int ( c );
        s_print_string ( " です。\n" );

        return 0;
}
```

```
12.34
```

図 27: 入力例

```
$ ./20181019-03-QQQQ.exe
変数 c の値は 127 です。
$
```

図 28: 20181019-03.c の実行結果

1.4.4 課題 20181019-04 : while

Download : 20181019-04.c[24]

リスト 19: 20181019-04.c

```
/*
 * 課題 CNAME-04
 *
 * CDATE FILENAME
```

[24] ex/04/q/04.c

```
 *
 *        while 文
 */

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

/*
 *        main
 */

int main( int argc, char *argv[] )
{
        /* 整数型変数 odd の宣言とその初期化 ( 初期値は 1 を指定 ) */
        int odd = 1;
        /* 整数型変数 sum の宣言とその初期化 ( 初期値は 0 を指定 ) */
        int sum = 0;

        /*
                sum = 1 + 3 + .. + 9 = 25 = 5^2
        */

        /* odd の値が 9 以下の間は繰り返す */

        while ( /* q:ここ */ )        {

                /* sum に odd の値を「追加」する */

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


                /* odd の値を次の奇数にする */
                odd = odd + 2;

        }

        /* 整数型変数 sum の値を画面に出力 */
        s_print_string ( "1 から 9 までの奇数の和は " );
        s_print_int ( sum );
        s_print_string ( " です。\n" );

        return 0;
}
```

12.34

図 29: 入力例

```
$ ./20181019-04-QQQQ.exe
1 から 9 までの奇数の和は 25 です。
$
```

図 30: 20181019-04.c の実行結果