第3回
プログラミング及び実習 II

第2回課題の回答例

ex02-6

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    if (n>0)
        printf("%d は正です\n", n);
    else if (n<0)
        printf("%d は負です\n", n);
    else
        printf("%d です\n", n);

    return 0;
}
```


ex02-7

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    while (n >= 0) {
        printf("%d\n", n);
        n--;
    }
    return 0;
}
```

ex02-8

```
#include <stdio.h>

int main(){

    int a;
    scanf("%d", &a);

    if (a > 0) {
        while (a >= 0) {
            printf("%d\n", a);
            a--;
        }
    }
    else if (a < 0) {
        while (a <= 0) {
            printf("%d\n", a);
            a++;
        }
    }
    else
        printf("%d\n", a);

    return 0;
}
```


ex02-8'

```
#include <stdio.h>

int main() {
    int n, d;
    scanf("%d", &n);

    if (n>0) d = -1;
    else d = 1;

    while (n != 0) {
        printf("%d\n", n);
        n += d;
    }

    printf("0\n");
    return 0;
}
```


ソースコードをデバッグ(debug)する

cc のエラーを読む

```
y20999@mypc:~$ cc ex03.c -o ex03
ex03.c:1:9: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or
 ‘__attribute__’ before ‘<’ token
 include <stdio.h>
         ^
```

before '<'：< の前で

expected 〜 ：こんなのが期待されてるんですけど…

ソースコードのこのへんでワケが分からなくなりました

ex02-2.c というファイルの1行目の9文字目で error 発生

ex03.c の main() 関数の中の…

```
y20999@mypc:~$ cc ex03.c -o ex03
ex03.c: In function ‘main’:
ex03.c:5:4: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or
 ‘__attribute__’ before ‘int’
    int v, s;
    ^
ex03.c:7:17: error: ‘l’ undeclared (first use in this function)
    scanf("%d", &l)
                 ^
ex03.c:7:17: note: each undeclared identifier is reported only once for
each function it appears in
```

5行目の int の前???

〜 undecleared：そんなの知りません（宣言されてません）けど…

error と warning

error は失敗：コンパイルできませんでした

warning は警告：コンパイルはできたけど要注意

```
y20999@mypc:~$ cc ex03.c -o ex03
ex03.c:3:1: warning: type specifier missing, defaults to 'int' [-Wimplicit-
int]
main(){
^
ex03.c:5:5: error: implicit declaration of function 'tScanf' is invalid in
C99 [-Werror,-Wimplicit-function-declaration]
    tScanf("%d",&n);
    ^
1 warning and 1 error generated.
```


本日の目標

C言語の関数を理解する（main 以外の関数定義は後半）

main() 関数

return 文

整数型・実数型・void型の変数とキャストを理解する

関数

数学の関数

プログラミング言語の関数　func(a)　e.g. printf()

```
int a       func()       int v
```


プログラミング言語の関数

プログラミング言語の関数

入力がない場合もある

出力がない場合もある

別の入力や出力をする場合もある

```
int a       func()       int v

            func()       int v

int a       func()

            scanf()

int a       func()       int v

            printf()
```

main() 関数

C言語で最初に呼び出される関数

実行可能なCプログラムを書く ＝ main() 関数を定義する

return 文によって関数の値（返り値）を定義する

main という名前の関数

```
int main() {
                                main()      0
   return 0;
}
```

main は int 型の0を（返り値として）返す

main は int 型を（返り値として）返す

関数と返り値とreturn文

C言語の関数は「関数名() {...}」で定義される（いまはmain() だけ詳細は後日）

C言語の関数は「関数名()」で呼び出される(利用される)

C言語の関数は値をもつ（もたないこともある →）　int a　func()

return 文によって関数の値（返り値）が決まる

(実は) scanf() 関数と printf() 関数も int 型の返り値をもつ

```
int a, n;

n = scanf("%d", &a) + printf("a is %d\n", a);
// n?
```


main() 関数の return 文に意味あるの？

プログラムを実行した処理系（システム）やユーザーに、
そのプログラムがどういった状況で終了したかを伝える。

bash なら、実行直後に echo $? するとmain()関数の返り値が見られる。

```
int main() {
     :
   if (エラー A が発生した！)
       return -1;
     :
   if (ユーザーに終了が指示された！)     main()      -1, 0, 99?
       return 99;

     :
   return 0; // 正常終了
}
```


本日の目標

C言語の関数を理解する（main 以外の関数定義は後半）

main() 関数

return 文

整数型・実数型・void型の変数とキャストを理解する

整数型・実数型・void型の変数とキャストを理解する

いろいろな変数型(基本型、void型、符号なし変数)

異なる型での算術演算

代入による型変換

明示的な型の変換

変数の型(type)

基本型

整数型
(char, short, int, long)
char 型はまた後で

浮動小数点型
(float, double, long double)

void 型

整数型(integer types)

整数を保持する変数や定数の型：
基本的に保持できるサイズが違うだけ

char 1バイト = 8ビット (-128〜127)

short 2バイト(以上) (-32,768〜32,767)

int 2または4バイト

long 4バイト(以上) (-2,147,483,648〜2,147,483,647)

(**long long** 8バイト(以上) ANSI C99 より)

大きさは処理系によって異なるが順序はある
char <= short <= int <= long <= long long

変数の型(type)

基本型

整数型
(char, short, int, long)

浮動小数点型
(float, double, long double)

void 型

浮動小数点型(floating point types)

実数を保持する変数や定数の型：
保持できるサイズと精度が異なる

float 4バイト = 32ビット

double 8バイト = 64ビット

long double 10バイト= 80ビット

大きさはC言語規格で決められていないが、ほとんどの処理系で
IEEE754–1985 という規格に沿っており上の通りだと思ってよい

変数の型(type)

基本型

整数型
（char, short, int, long）

浮動小数点型
（float, double, long double）

void 型

void型

値がないことを明示的に示すための型

```
#include <stdio.h>

void main(void) {
   int a;
   void v;

   return;
   // return 0;
}
```

main() 関数の返り値（出力）は「無い」

// int型の変数 a は定義できるが、
// void型の変数定義はエラーになる

main() 関数の引数（入力）は「無い」

void な main() 関数は何も値を返さない

void　main()　void

unsigned (符号なし変数)

int など整数型の変数は、変数定義の際に unsigned を型名の前に付けることで、符号なし変数として定義することができます。

```
#include <stdio.h>

int main() {
   int a;               // int型 (-2,147,483,648〜2,147,483,647)
   unsigned int ua;     // unsigned int型 (0〜4,294,967,296)

   unsigned double ux;  // unsigned double は無いのでエラー！

   ua = -1;             // ただし負数を代入できてしまうので注意！
   ua = 4294967296

   printf("ua = %u\n", ua);

   return 0;
}
```

%u で unsigned int として表示

整数型・実数型・void型の変数とキャストを理解する

いろいろな変数型(基本型、void型、符号なし変数)

異なる型での算術演算

代入による型変換

明示的な型の変換

異なる型での算術演算

異なる型同士の演算はより**上位**の型に自動的に（暗黙的に）変換されて計算される

int < long < long long < float < double < long double

（char, short の演算は全て自動的に int に変換される）

```
int i = 2;       long l = 3;
float f = 2.1;   double d = 3.0;

i+l;            // i は自動的に long に変換され i+l は long (5)

i*f;            // i は自動的に float (2.0) に変換され i*f は float (4.2)

i*0.1;          // i は自動的に double (2.0) に変換され i*0.1 は double (0.2)

(i/l)*f;        // i は自動的に long に変換され i/l は long (0)
                // long (0) は自動的に float に変換され (0.0)*f は float (0.0)

i*(l/0.1);      // l は自動的に double に変換され l/0.1 は double (30.0)
                // i は自動的に double に変換され (30.0)*f は double (60.0)
```


代入による型変換

異なる変数型への代入は自動的に（暗黙的に）その変数型に変換代入される

a = b;　// b の値は常に a の変数型に変換され代入される

```
int i = 2;       long l = 3;
float f = 2.1;   double d = 3.0;

i = 0.1;        // 0.1 は自動的に int に変換され i は int (0)

i = f;          // f は自動的に int に変換され i は int (2)

l = f*i;        // i は自動的に float (2.0) に変換され f*i は float (4.2)
                // float (4.2) は自動的に long に変換され l は long (4)

d = i%l/f;      // i は自動的に long に変換され i%l は long (2)
                // long (2) は自動的に float に変換され
                //                          (2.0)/f は float (0.952381)
                // float (0.952381) は自動的に double に変換され
                //                              d は double (0.952381)
```


明示的な型の変換

キャスト(cast) によって明示的に型変換

(type)x;　// x の型は明示的（強制的）に type型に変換される

```
int i = 2;       long l = 3;
float f = 2.1;   double d = 3.0;

(int)0.1;       // 0.1 は int に変換され int (0)

l/(double)i;    // i は double に変換され double (2.0)
                // l も double に変換され double (1.5)

d*(int)f;       // f は int に変換され int (2)
                // int (2) は double に変換され d*(2.0) は double (6.0)

(int)f*d;       // f は int に変換され int (2)
                // int (2) は自動的に double に変換され double (6.0)

(int)(f*d)      // f は自動的に double に変換され f*d は double (6.3)
                // double (6.3) は int に変換され int (6)
```