

## 情報リテラシー第10回：C言語によるプログラミング入門(2) 修正版 －最終課題をこなすための世界で最も短いC言語入門－

### 1. Cプログラム開発環境の整備 (続き)

前回の講義ノート5で折角、TeraPad@MinGWで追加したccという外部コマンドを使って編集・保存したCのソースコードをgccで編集時のCソースのファイル名(例えばfilename.c)から実行形式のファイル名を(filename.exe)としてコンパイルするツールを加えたのにも関わらず、立ち上げてみるとわかるが消えてしまっている(あらまっちゃんでもそのちゅうがえり!)。これは普通のPCの環境ではあり得ないことであるが、教育用計算機の場合、たぶんツールの設定が保存される先がデスクトップなどのように再起動時にリセットのかかってしまう領域に保存されるためであろう。

また、TeraPadのこのツールccでのコンパイルでは瞬間にDOS窓が開き実行されるが、ソースコードにエラーがあっても直ぐに閉じてしまうので、開発には向かないことがわかった。そこで次の記述するようなcc.batというバッチファイルをMinGWフォルダ内に作成し、コマンドプロンプト(MinGW)のMS-DOS画面からgccによる同様な簡素化したCソースコードのコンパイルを行うことにする。ここでは、そのための手順と関連したWindowsでの設定について述べる。

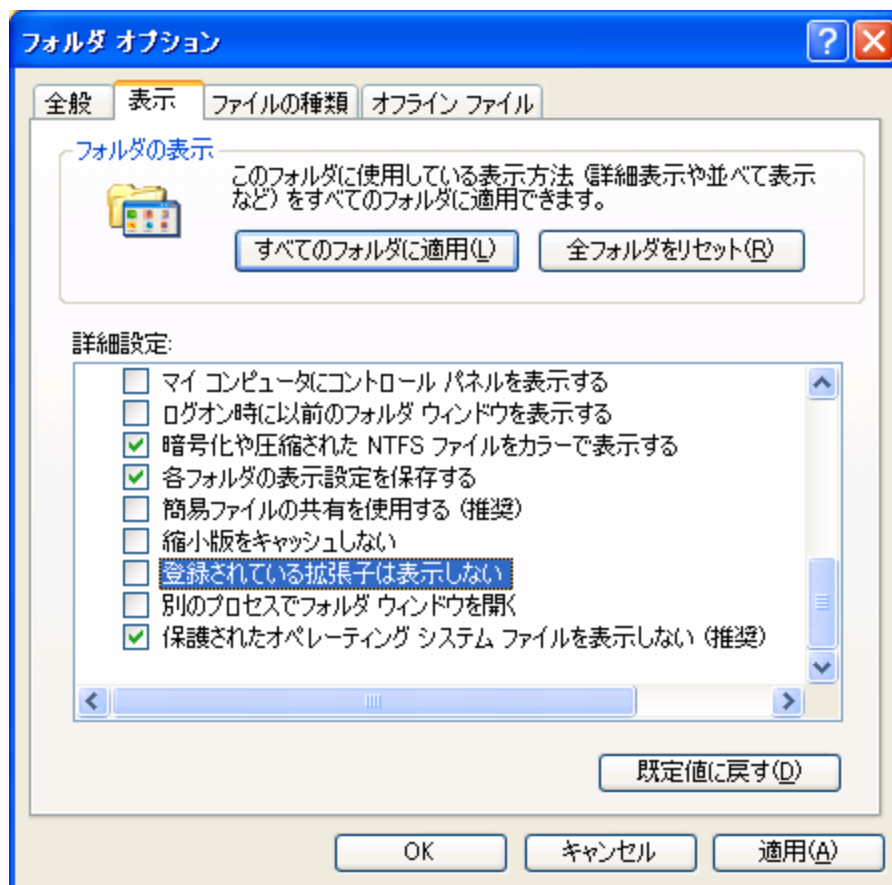


図 1

まず、これは毎回行わなければならない（やはり教育用計算機では設定が保存されない）ので、Windowsで開いた窓からメニューからツール→フォルダオプションを選択し、表示で図1に示すように「登録されていない拡張子は表示しない」のチェックをはずす。そうすると、図3、4のように各ファイルの拡張子（extensionという；通常3文字までで、フルでのファイル名をfilename.extとするとextに相当する部分を指す。これがdocならば、WordファイルであるとWindowsシステムでは関連付けている訳である）が表示されるようになる。この方が開発したCのソースコード（\*.c）と実行形式（\*.exe）の区別がついて見やすくなる（ここでは\*は何でもということを表すワイルドカードと呼ばれるものの一つである）では、ここでTeraPad@MinGWショートカットからcc.batをMinGWフォルダ内に図2のように入力して作成しよう。

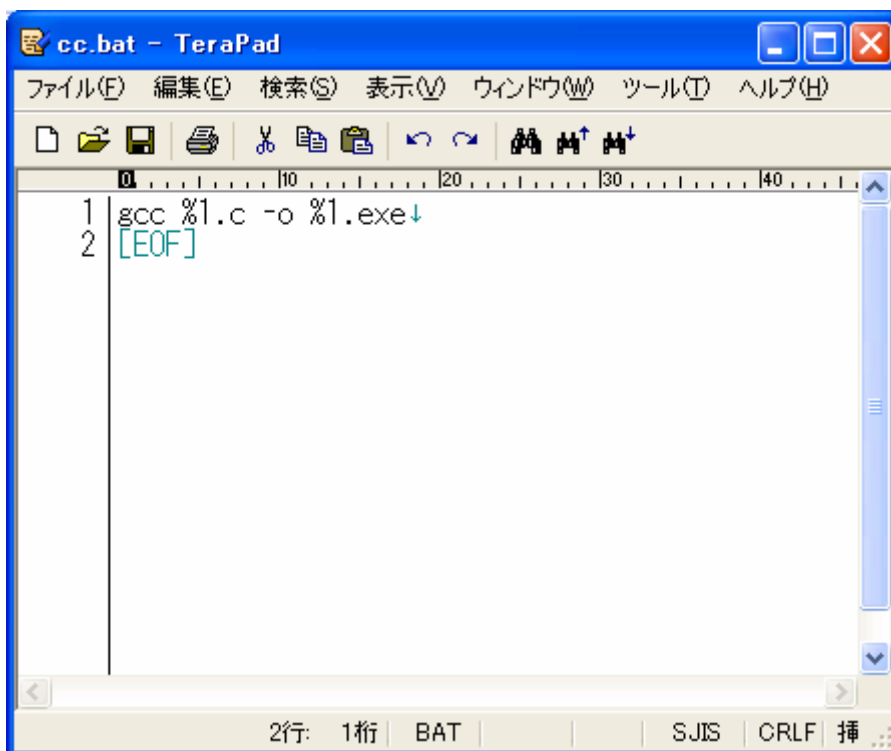


図2

ここで%1はDOS窓でのコマンドラインでの最初の引数を表すので、図6を見ると前回作ったhelloworld.cというプログラム（図5）のコンパイルが%1にhelloworldを入れた形で成されているのがわかるでしょう。

図3→図4へはWindowsでのMinGWのフォルダ表示を各ファイルが生成された時刻や種類も一覧として表示させるための表示をアイコンから詳細にする方法を示している。この表示の方が、DOS窓からでなくてもファイル生成時刻からコンパイルが成功しているかどうか判断の材料になるし、ファイルのソーティング（名前のアルファベット順、種類順、ファイル生成順に並べ直す）が可能となり便利である。

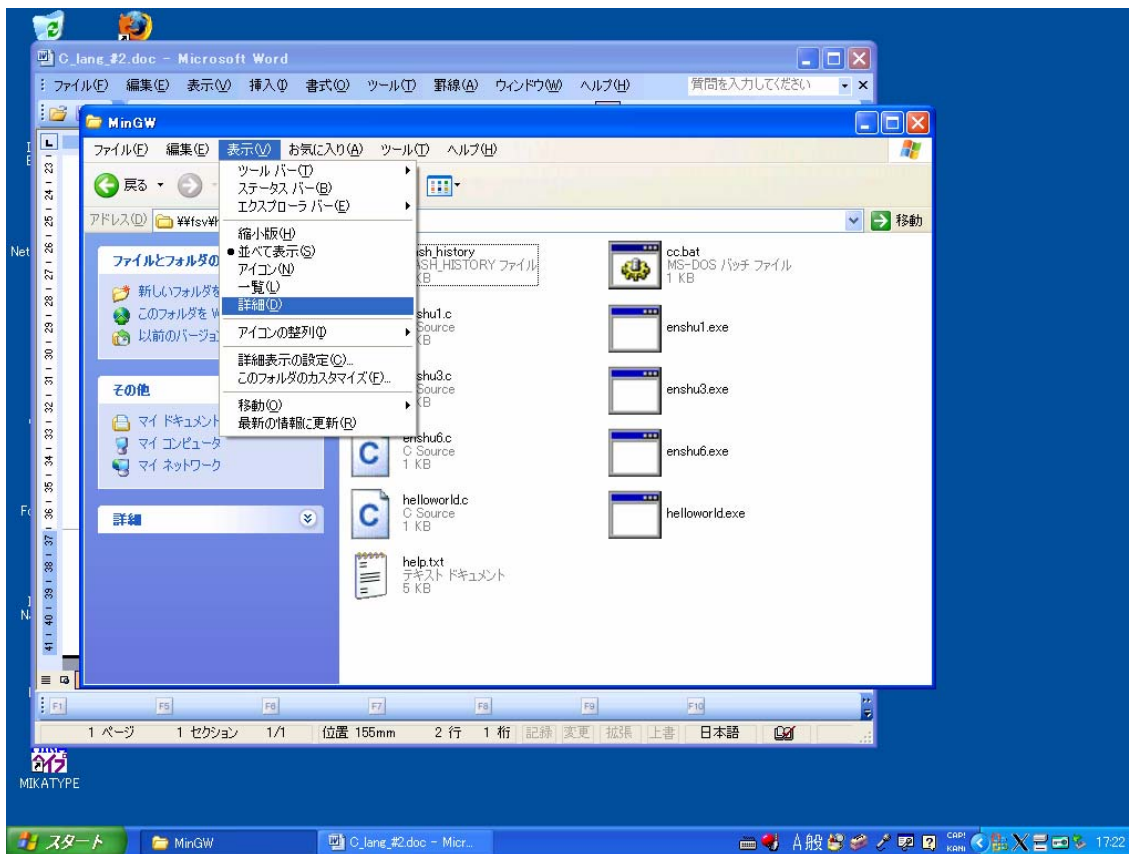


図 3

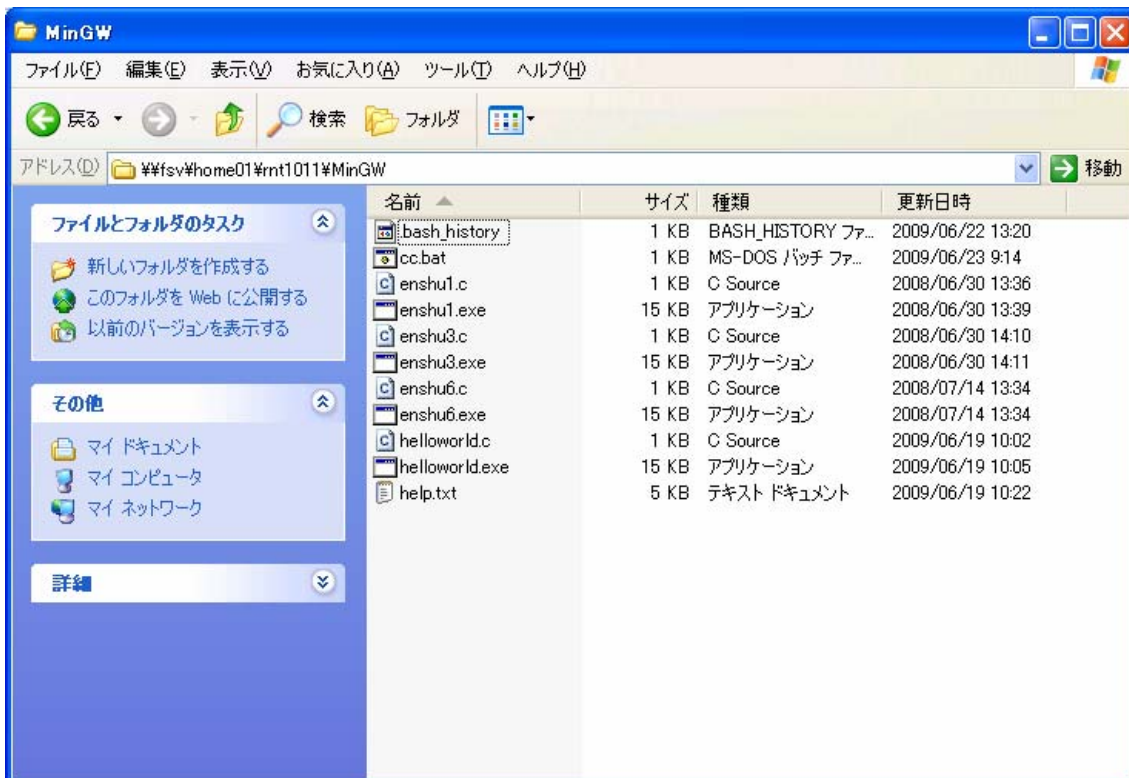


図 4

```

helloworld.c - TeraPad
ファイル(E) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(T) ヘルプ(H)
1 #include <stdio.h>↓
2 ↓
3 main()↓
4 {↓
5     printf("Hello world.¥n");↓
6 }↓
7 [EOF]
7行: 1桁 C/C++ SJIS CRLF 挿入

```

図 5

```

コマンド プロンプト (MinGW)
Z:¥MinGW>dir
ドライブ Z のボリューム ラベルは home01 です
ボリューム シリアル番号は F002-37A0 です

Z:¥MinGW のディレクトリ

2009/06/25 17:17 <DIR>      .
2009/06/19 10:30 <DIR>      ..
2009/06/22 13:20          332 .bash_history
2008/06/30 13:36           64 enshu1.c
2008/06/30 13:39       14,627 enshu1.exe
2008/06/30 14:10           324 enshu3.c
2008/06/30 14:11       14,627 enshu3.exe
2009/06/19 10:22         4,395 help.txt
2008/07/14 13:34           186 enshu6.c
2008/07/14 13:34       14,627 enshu6.exe
2009/06/19 10:02           64 helloworld.c
2009/06/19 10:05       14,627 helloworld.exe
2009/06/23 09:14           20 cc.bat

          11 個のファイル          63,893 バイト
           2 個のディレクトリ     302,333,952 バイトの空き領域

Z:¥MinGW>cc helloworld

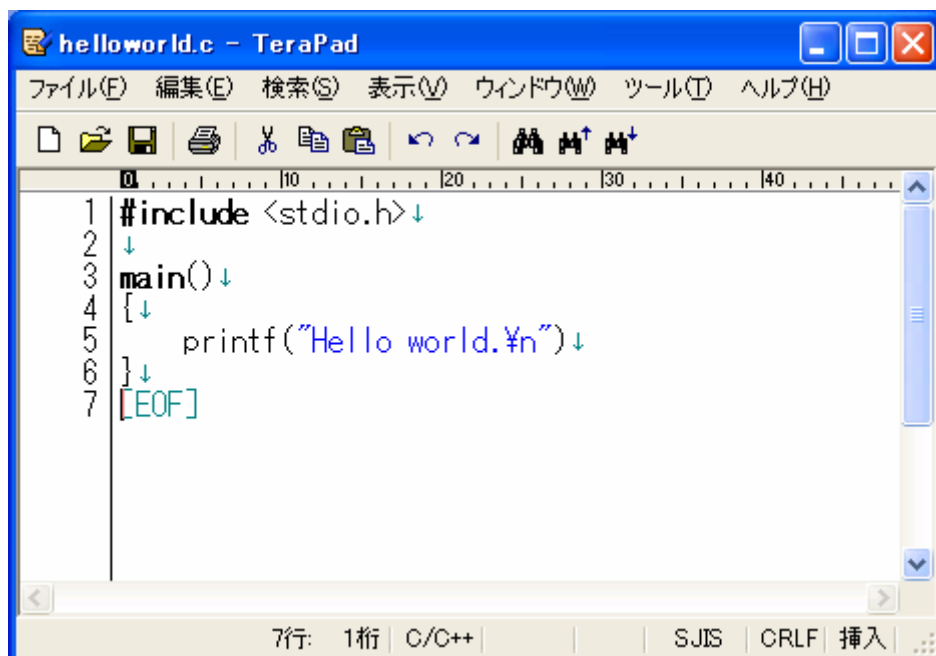
Z:¥MinGW>gcc helloworld.c -o helloworld.exe

Z:¥MinGW>

```

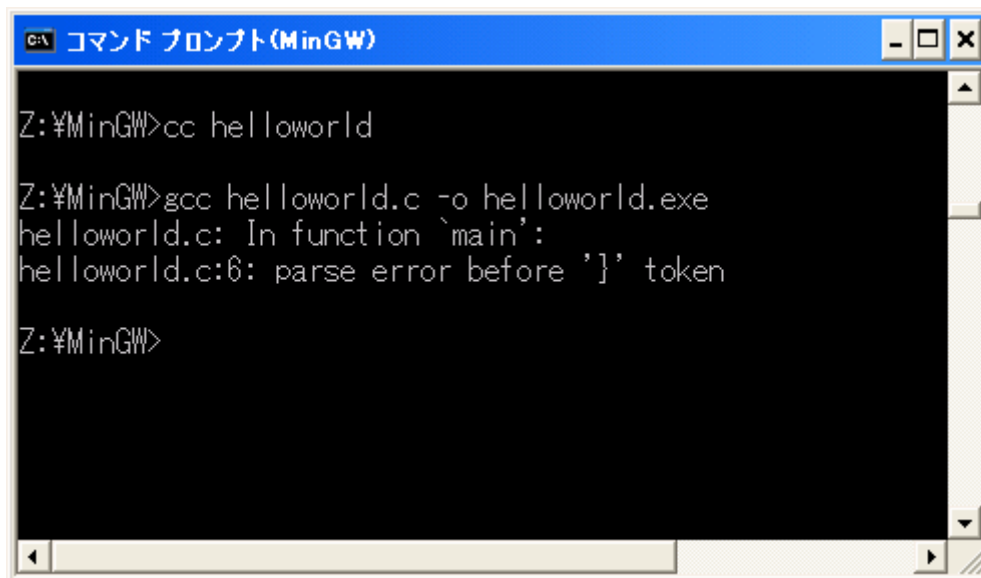
図 6

ここで DOS 窓で cc.bat を使った gcc によるソースコードのエラー出力の例をみる。図5のプログラムで5行目で、Cでは実行したい式の右端には必ずセミコロン (;) を書き区切る必要があるのだが、わざとここではそれを取って保存し (図7)、cc.bat を使ってコンパイルしてみる (図8)。



```
helloworld.c - TeraPad
ファイル(E) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(T) ヘルプ(H)
[Icons]
1 #include <stdio.h>
2
3 main()
4 {
5     printf("Hello world.%n")
6 }
7 [EOF]
7行: 1行 C/C++ SJIS CRLF 挿入
```

図7



```
コマンド プロンプト (MinGW)
Z:\MinGW>cc helloworld
Z:\MinGW>gcc helloworld.c -o helloworld.exe
helloworld.c: In function 'main':
helloworld.c:6: parse error before '}' token
Z:\MinGW>
```

図8

すると図8のようにその影響は6行目の前に **parse error** (文法の間違い) があるよと英語 (残念ながら常に) で出てくるのでそこで、その前に何かエラーがないか考える訳である。こういうプログラムの修正作業をデバック (誤りを取る) 作業という。

## 2. Cの問題演習 (今日はここからが本番)

以下は主に参照 URL から引用した次週に出題する課題をこなすために必要な基本的な C の文法を示す。各人、枠で囲まれた例題のプログラム (reidai?.c) を TeraPad@MinGW で入力・保存して、コマンドプロンプト (MinGW) で、MinGW フォルダ内に各 \*.c と実行可能な \*.exe を作成して欲しい。ソースコードを作成する上での注意は、コンパイラは式としては半角英数字文字しか判別しないので、その中に全角の日本語文字を入れないこと (これが多くの場合、見えないエラーの原因 ; 例えばスペース (空白 1 文字) を全角文字で入れてしまっていたりする)。コンパイルに関係しないコメント (説明文) は /\* comments \*/ のように記号 /\* と \*/ でくくって加えることが出来るが、例題のように全て書く必要はない (全角でプログラムソースを記してしまう間違いをし易くなる)。また、C では見た目に字下げを Tab (TeraPad のデフォルトでは 8 文字となっているが、ちょっと深過ぎで 4 文字くらいがちょうどよい) で行って、{ ではなく { で始まり } で終わる入れ子構造を確認しながら作成することが出来る。だから、スペースで字下げを行うのではなく Tab キーを使うこと。

その他、C 言語に特徴的なことは、小文字を使うのが基本であること、プログラム中で使用する変数の型宣言を main() { で始まる main 関数の最初に全て記す点である。

### 2.a 変数を使ったプログラム

reidai1.c

```
#include <stdio.h> /* ここが 1 行目。include 文で関数を定義した header file *.h を読み込む。
ここで、stdio.h は printf() など標準入出力関数が定義されている */
main()
{
    int a, b=2; /* 整数型の a と b を宣言して、b を 2 で初期化 */
    char c; /* 文字型の c を宣言 */
    a = 3; /* 代入。変数 b のように宣言時に初期化してもよい */
    c = 'A'; /* c に、文字 A を代入 ( c は整数型でなく文字型である) */
    printf("a = %d\n", a);
        /* a = 3 と表示して改行。%d は、整数 a と置き換わる */
    printf("b = %d\n", b);
        /* b = 2 と表示して改行。%d は、整数 b と置き換わる */
    printf("c = %c\n", c);
        /* c = A と表示して改行。%c は、文字 c と置き換わる */
        /* 命令の前の空白は、TAB キーで入力するのが普通 */
}
```

変数とは数を入れておける箱のようなものです。変数を使用するには、(中カッコの中の) 一番はじめに宣言を書かねばなりません。宣言は、同じ型 (「よく使うデータ型一覧」を参照) である

ならカンマ (,) で区切っていくつでも書くことができます。変数の名前には、アルファベット、数字そしてアンダースコア ( \_ ) を使うことができます。ただし、先頭の 1 文字に数字は使えません。変数の値を `printf()` で表示するために、`%d` や `%c` といった文字列を使用しています。`%` で始まる文字列は、カンマで区切った後に書いてある変数の値と置き換わります。また、`printf()` は文字列や変数の値を指定した書式に画面に出力させる標準関数で、” “で囲まれた中に `%d` や `%c` などの変換指定子の後にここでは `¥n` (これは画面に出力改行するの意) が付いています。これはエスケープシーケンスと呼ばれるもので、よく使われるものを以下に記しておきます。

## よく使うデータ型一覧

型	扱える値の範囲	printf 中の表記	サイズ(バイト)
char	-127~128(文字)	<code>%c</code> (文字を表示)、 <code>%d</code> (値を表示)	1
int	-32768~32767	<code>%d</code>	2
long	-2147483648~2147483647	<code>%ld</code>	4
float	$10^{38} \sim 10^{-37}$ ; 有効桁数 6~7 桁の実数	<code>%f</code> ( <code>%e</code> )	4
double	$10^{308} \sim 10^{-307}$ ; 有効桁数 15~16 桁の実数	<code>%f</code> ( <code>%e</code> )	8

## よく使うエスケープシーケンスの例

エスケープ文字列	機能
<code>¥n</code>	改行
<code>¥t</code>	水平タブ
<code>¥r</code>	キャリッジリターン(複改)
<code>¥f</code>	フォームフィード(改頁)
<code>¥"</code>	文字"を表示する
<code>¥¥</code>	文字¥を表示する

## 2.b Cでの算術演算子

C 言語での記号 数学での記号 機能

`++` 加算(足し算)

`--` 減算(引き算)

`*` `×` 乗算(掛け算)

`/` `÷` 除算(割り算)

`%` … 剰余算(割り算の余り)

この表を見ると、「数学とは異なる記号」を使うことがあることに気づくと思います。一般的なコンピュータのキーボードで表現出来る記号には `×` や `÷` はないため、C 言語に限らず、多くのプログラミング言語では、別の記号が使われています。演算子の使い方は数学と全く同じです。次のプログラムは、ここで紹介した演算子を一通り使ってみる例です。

`reidai2.c`

```
#include <stdio.h>
main()
{
    printf("%d¥n",10 + 3);
    printf("%d¥n",10 - 3);
    printf("%d¥n",10 * 3);
    printf("%d¥n",10 / 3);
    printf("%d¥n",10 % 3);
}
```

このプログラムの実行結果は、次の通りになります。

[実行結果]

```
13
7
30
3
1
```

このプログラムで注目してほしい部分は、「10/3(10÷3)の計算結果が 3」と表示されていることです。電卓なら、3.3333333 と表示されるのが普通ですが、ここではどの数値も整数として計算しているので、結果も整数になります。

今までは、数値を表示する時には、%d 指定子を使用してきましたが、これは、整数値を数字に変換するための指定子です。実数値を数字に変換する場合は、「%f」指定子を使用しなければ行えません。ここまでわかれば、後は簡単なことです。

次のプログラムは、前章のプログラムを、実数による計算に直した例です（**剰余の計算は除いています。何故なら%演算子は整数型の演算だけに使用されるからです。**）。

`reidai3.c`

```
#include <stdio.h>
main()
{
    printf("%f¥n",10.0 + 3.0);
    printf("%f¥n",10.0 - 3.0);
    printf("%f¥n",10.0 * 3.0);
    printf("%f¥n",10.0 / 3.0);
}
```



このプログラムの実行結果は、次の通りになります。

[実行結果]

```
13.000000
7.000000
30.000000
3.333333
```

#### reidai4.c

reidai3.c の場合と似ていますが、今度は%d を%10.5f に置き換えて下さい。これは結果を少数点を含めて 10 文字で小数点以下を 5 桁まで表す指定となっています。そのような結果になったでしょうか？数値の無い大きい桁にはスペース（半角 1 文字分の空白）が入ります。このような書式指定は演算結果の書式を揃える上で便利です。

#### 2.c 計算をしてみよう

次の例題では実際に使う変数を型宣言した上で、簡単な算術計算を行っています。

#### reidai5.c

```
#include <stdio.h>
#include <math.h> /* 数学関数のヘッダファイルをインクルードする */

main()
{
    double radius = 10;    /* 半径 */
    double pi = 3.1415926; /* π */
    double circumference; /* 円周（こんなに長い名前も使える） */
    double area;          /* 面積（変数名には誰が見ても分かる名前を） */
    double volume;        /* 体積 */

    circumference = radius * 2 * pi;
    area = radius * radius * pi;
    volume=4.0*pi*pow(radius, 3.0)/3.0; /* pow(x,y)は x の y 乗の値を返す数学関数 */

    printf("半径 = %f\n", radius); /* double を表示するには %f */
    printf("半径 %f の円周 =%f\n", radius, circumference);
    printf("半径 %f の円の面積 = %f\n", radius, area);
    printf("半径 %f の球の体積 = %f\n", radius, volume);
```

```
printf("¥n¥n");          /* 改行を 2 つ */
}
```

今度は整数でなく、実数（小数点以下がある数）の計算をするために、**double** 型（**float** でも可）を使用しています。また、計算の優先度もあるので、加算を乗算より先に行いたいときはカッコでくくるようにしましょう。

より汎用にするためにキーボードから半径を入力できるように **scanf()**関数を用いて次のように少し書き変えて実行してみましょう。

`reidai6.c`

```
#include <stdio.h>
#include <math.h> /* 数学関数のヘッダーファイルをインクルードする */

main()
{
    double radius;    /* 半径 */
    double pi=3.1415926; /* π */
    double circumference; /* 円周（こんなに長い名前も使える） */
    double area;      /* 面積（変数名には誰が見ても分かる名前を） */
    double volume;    /* 体積 */

    printf("任意の半径の値を入れて下さい: ¥n");
    scanf("%lf", &radius);

    /* double の値を入力するには%lf (エルエフ) を用いる。&はポインタを表す記号だが、やっかいなのでここでは説明は省略 */
    circumference=radius*2.0*pi;
    area=radius*radius*pi;
    volume=4.0*pi*pow(radius, 3.0)/3.0; /* pow(x,y)は x の y 乗の値を返す数学関数 */
    printf("半径 = %f¥n", radius); /* double を表示するには %f */
    printf("半径 %f の円周 = %f¥n", radius, circumference);
    printf("半径 %f の円の面積 = %f¥n", radius, area);
    printf("半径 %f の球の体積 = %f¥n", radius, volume);
    printf("¥n¥n");          /* 改行を 2 つ */
}
```

## よく使う演算子一覧

算術演算		論理／ビット演算	
*	乗算	&&	論理積(AND)
/	除算		論理和(OR)
+	加算	==	等しい
-	減算	!=	等しくない
%	剰余(余り)	<	より小さい
++	1 加算(単項)	<=	以下
--	1 減算(単項)	>	より大きい
*=	積の代入	>=	以上
/=	商の代入	!	論理否定(NOT)(単項)
+=	和の代入	&	ビットごとの AND
-=	差の代入		ビットごとの OR
%=	余の代入	^	ビットごとの XOR
		~	ビットごとの NOT(単項)

## 2.d 繰り返し処理

プログラミングのアルゴリズム処理及びプログラムらしさを実感するには「分岐処理」と「繰り返し処理」を学ぶのがよいが、ここでは時間と目的の関係上、繰り返し処理のみ紹介する。

```
reidai7.c
```

```
#include <stdio.h>
main()
{
    int i;

    for(i=0; i<20; i++){ /* 20 回繰り返す。(i は 0 から 19 まで) */
        printf("i = %d\n", i);
        /* ここには、繰り返したい文をいくつも書くことができる */
    }
}
```

まず、一番はじめに `i=0` が実行されます。次に `i<20` を評価し、条件が成立していれば `printf()`

を実行します。それが終わると `i++` を実行し、また `i<20` の評価に移ります。それが成立していれば、後は `printf() => i++ => i<20` と、繰り返すこととなります。もちろん、`i<20` が成立しなくなったら、`for` 文は終了します。

プログラムを書く上で、`for` 文の中は 1 タブ深く右に字下げされていることに注意して下さい。しかし、閉じる括弧記号 `}` は `for` 文の頭と同じ位置に戻ってきています。

あまりいい例ではありませんが、`for` を `while` で書き換えると次のようになります。

`reidai8.c`

```
#include <stdio.h>
main()
{
    int i;

    i=0;
    while(i < 20){      /* i < 20 が成立している間繰り返し */
        printf("i = %d\n", i);
        i++;
    }
}
```

最後に、`do~while` 文というのを見てみましょう。

`reidai9.c`

```
#include <stdio.h>
main()
{
    int i=0;
    do{                /* 条件式はループの最後を書く */
        printf("i = %d\n", i);
        i++;
    }while(i < 20);
}
```

これらの構文をまとめると・・・

- `for` 文の書式

`for(式 1; 条件式; 式 2)`

文

※ 式 1 → 条件式 → 文 → 式 2 → 条件式 → . . . の順番

- while 文の書式

while(条件式)

文

※ 条件式 → 文 → 条件式 → . . . の順番

- do-while 文の書式

do

文

while(条件式);

※ 文 → 条件式 → 文 → 条件式 → . . . の順番

上の 3 つのプログラムは全く同じ動作をしています。

## 2.e 配列を使う

ここでは最終課題を行うにあたって配列を使った方が便利であること、どんな高級言語でも必ず配列変数はあるのでここで最も簡単な 1 次元配列を使った例について述べる。最初の例では `scanf()`関数を用いて配列の要素を 5 個まで順番にキーボードから読み込んでいます。また、ここでは整数型の 1 次元配列を `MAX` で 5 要素まで定義しているので、

```
int point[5];
```

としても同じですが、これは配列の要素が 5 個で、`point[0]`, `point[1]`, `point[2]`, `point[3]`, `point[4]`までの 5 つしか確保していないことに注意しましょう。

```
reidai10.c
```

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

```
#define MAX 5      /* これ以後の MAX を 5 に読み替える命令 */
```

```
main()
```

```
{
```

```
    int i;
```

```
    int point[MAX];    /* MAX 個の要素を持つ配列 point を宣言 */
```

```
    int total=0, avr;
```

```
    for(i=0; i<MAX; i++){    /* MAX 回の繰り返し */
```

```

printf("%d 人目の点数は? >", i+1);
scanf("%d", &point[i]);
    /* 配列の場合でも & を忘れずに付ける */
}
for(i=0; i<MAX; i++){
    printf("%d 人目 = %d¥n", i+1, point[i]);
    total += point[i];
}
avr = total / MAX;
printf("平均点 = %d¥n", avr);
}

```

次に配列を初期化した例を示しましょう。

`reidai11.c`

```

#include <stdio.h>
#define MONTHS 12
main()
{
    int day[MONTHS+1] =    /* 配列の初期化は中カッコでくくる */
        { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    int input;

    printf("今日は何月ですか? >");
    scanf("%d", &input);
    printf("すると今月は %d 日までだから、", day[input]);
    /* if else 文は本稿では省略した分岐処理の一つです。感じをわかって下さい。*/
    if (day[input] == 31)    /* 単文だから中カッコを省く */
        printf("月末には中野ブロードウェイへ行きましょう¥n");
    else
        printf("サーティワンの日はないのですね¥n");
}

```

参考URL及び参考書:

- C言語入門 <http://www.nmn.jp/~hidai/c/>
- 苦しんで覚えるC言語 [http://homepage3.nifty.com/mmgames/c\\_guide/index.html](http://homepage3.nifty.com/mmgames/c_guide/index.html)

その他、参考書となる本は沢山あるので図書館を利用して下さい。自分が持っている中では例えば、

- 杉江 日出澄・鈴木 淳子 共著「C言語と数値計算法」培風館，2001.

**Tips:**

プログラムの間違いがどうしてもわからないときは、このような統合開発環境でない場合、丁寧なエラーの内容が `gcc` のエラーメッセージで出てこないことが多いので自分だったら次の方策を取ります。

- 1) プログラムソースをプリンターに出力させよく眺めて考える。
- 2) `getchar()`関数はキーボードから入力された1文字を取り込む関数であるが、エラーの主生じているところで、  
`getchar();`  
を挿入し、一時的にプログラムを中断させると同時に、その前に問題になっている変数の値他を `printf()`文で画面に出力させ、途中経過をチェックする。