# プログラミング通論'19 ＃1 – C言語の基本機能

久野 靖 (電気通信大学)

2019.4.20

今回は初回ですが、次のことが目標となります。

- C言語の基本的な機能について復習し、関数・変数・代入・制御構造を用いたプログラミングができることを確認する。

ただしその前にガイダンスから始めて、その後本題に入ります。

## 1　ガイダンス

### 1.1　本科目の主題・目標

本科目の主題ならびに達成目標は次のようになっています。

**主題:** プログラミングの初歩は学習したという前提で、再帰的手続き、データ構造の初歩、および、基本的アルゴリズムについて学習する。

**達成目標:** 再帰的手続き、基本的なデータ構造、基本的なアルゴリズムを理解し、それらを用いたC言語のプログラムを読むことと、書くことができる。

本学では授業1単位について45時間の学修を必要とすることとなっています。本科目は2単位ですから90時間となります。これを15週で割ると週あたり6時間となります。授業そのものは90分(1.5時間)であるので、時間外に4.5時間の学修が必要です。課題等もこのことを前提に用意されていますので、留意してください。

### 1.2　本科目の運用

本科目の運用ですが、各時間の内容は前もってテキストを公開しますので、予習してくるようにお願いします。授業時間にはその要点だけ説明し、あとは演習をしていただきます。

各回とも、出席相当の課題として「プログラム1つを作成し報告する当日一杯締切の課題(A課題)」と、プログラム2つ以上を作成し報告する翌週授業前日一杯締切の課題(B課題)」を課します。これは、演習してプログラムを書かない限りプログラミングは身につかないからです。[1]

各回の課題レポートはCEDシステム上の提出プログラムで提出していただきます(学外からでもsolを経由してCEDシステムに入って作業できますし、提出だけならsolでもできます)。レポートの内容は互いに見られるように学内限定で公開します。従って、レポートには公開されたくないもの(個人の情報など)は書かないでください。名前を書かないわけにはいかないので、行頭が「`@@@`」となっている行は除外して公開します。氏名等を記入した行は「`@@@`」で隠すことを勧めます。[2]

すべての課題レポートは×(未提出ないしそれと同等)、△(要件を満たしていないか遅刻)、○(通常点=要件を満たしたレポートである)、◎(特に見るべきところがある)の4段階で評価します。全レポートが○のときがレポート点の満点であり、◎は上積み(または減点の相殺)となります。50点

[1] このほかにも、プログラミングが身に付くようにするためのさまざまな工夫を本科目中に盛り込んでいます。
[2] 必要な箇所のみ隠すこと。すべて隠そうとして山のように「`@@@`」をつけた場合は「`@@@`」を全て削除します。

を超えた部分は係数 $\alpha$ ($0 < \alpha \leq 1$、期末時に調整) を掛けて、59 点を上限としてから試験点と加算します。

レポートは必ず個人単位で出して頂き、個人単位で採点します (試験も)。複数人から同一内容ないしコピーと思われるレポートが出て来た場合、両方とも減点します (後述するペアでプログラムが同一なのは許容します)。

試験は期末試験のみで、プログラムを書く試験 (並べ替え式を含む) を課す予定です。試験点も他のクラスと SABCD の比率が乖離しないよう調整してから加算します。テキスト内容のうち表題に option と記されている節は試験範囲となりません。あと、テキストに載っていなくてもプログラミングの基本部分はいずれにせよ含まれるものと考えてください。

### 1.3 ペアプログラミング

本科目では「ペアプログラミング」を推奨します。これは次のようなものです。

- 1 つの画面の前に 2 人で座り、一人がキーボードを持ちプログラムを打つ。もう 1 人はそれを一緒に眺めて意見やコメントや考えを述べる。キーボードの担当者は適宜交替してもよい。

このような方法がよい理由としては、次のものがあげられます。

- プログラムを作って動かすのには多くの緻密で細かい注意が必要ですが、1 人でやるより 2 人でやる方がこれらの点が行き届き、無用なトラブルによる時間の空費が避けられます。
- プログラミングではたまたま「簡単な知識」が足りなくて、それを調べて使うまでにすごく時間が掛かることがありますが、2 人いればそのような知識を「どちらかは知っている」可能性が高まり、時間の無駄が省けます。
- プログラミング的な考え方を身に付けるには、さまざまな方面から思考したり、それを身体的な活動と結びつけることが有効です。2 人で互いに議論することで、思考が活発になり、多方面にわたるアイデアが出やすくなるため、上記のことがらに貢献します。

課題提出に際してはもちろん、2 人で作ったものですから、その 2 人については同一のプログラムを出して頂いて構いません。ただし次の条件があります。

- 提出するレポートにおいて、互いに「誰がペアであるか (相手の学籍番号)」を明示する。個人的な好みや都合よりペアを組まずに作業することも認めますので、そのときは「個人作業」と記してください。
- 「当日課題 (A 課題)」と「翌週までの課題 (B 課題)」でペアを変更したり、1 人で作業との間で変更しても構わない。ただし課題の「途中で」は変更しないこと。たとえば B 課題をペアでやる場合は、時間外もプログラミングについてはすべて 2 人で時間を合わせて作業すること。

ペアで複数のプログラムを作った場合、どれを提出するかは各自で選んで構いません。

### 1.4 レビュー課題

A 課題 (授業当日の課題) レポートにおいて、自分 (達) が作成したプログラムの 1 つを「ペア以外の誰か」に見せて説明し、ひとこと (1 行、30 文字程度) コメントをもらってください。それを提出して頂きます (レビュー課題)。授業時にクラスメートに見せてコメントをもらうのが簡単だと思いますが、事後に先輩、後輩、家族など誰にもらうのでも構いません。ただし「すごいねえ」「がんばったね」などのプログラムの中身と関係ない (または抽象的な) コメントではなく、「プログラムのどこのところが、どうである (よい、悪い、このように工夫されているなど)」という形のコメントであること。

この課題を課す意図ですが、プログラムを他人に説明するすることは自分でプログラムを客観化して見る大変よい練習になるからです。面倒だと思うでしょうが、有用なので必ずお願いします。

### 1.5 担当教員との連絡

資料や皆様のレポートを掲載する授業サイトは次の場所となります。

`http://www.edu.cc.uec.ac.jp/~ka002689/prog19/`

ここに掲示も出しますので、定期的に (授業期間中は毎日 1 回以上) チェックしてください。掲示を確認しなかったことによる不利益は救済しません。久野のメールアドレスは `y-kuno@uec.ac.jp` ですので、個別の質問等はこちらにお願いします。では、よろしくお願いします。

## 2 C 言語の基本機能

### 2.1 プログラムの構造

初回はウォームアップとして、C 言語の基本機能をざっと整理し、それらに基づいた演習をやりましょう。具体的には、プログラムの構造、型、関数、変数、計算式、代入、制御構造、入出力などの内容について整理することから始めます。

まず確認ですが、皆様は初心者ではないはずなので、次のようなプログラムは読み書きできるものとして扱います。[3]

```
// triarea --- area of a triangle
#include <stdio.h>
double triarea(double w, double h) {
  double s = (w * h) / 2.0;
  return s;
}
int main(void) {
  double w, h;
  printf("w> "); scanf("%lf", &w);
  printf("h> "); scanf("%lf", &h);
  double x = triarea(w, h);
  printf("area of triangle = %g\n", x);
  return 0;
}
```

ここでまず、C 言語のプログラムは「関数の集まり」であるということを指摘します。**関数** (function) は実行の単位であり、「(必要ならパラメタを付して) 呼び出され、内部で計算を実行し、終わったら呼び出された箇所に戻る (値を返すこともできる)」のでした。そして、プログラムはまず `main` が呼び出されてそこから始まり、`main` の実行が戻ると終了します (図 1)。

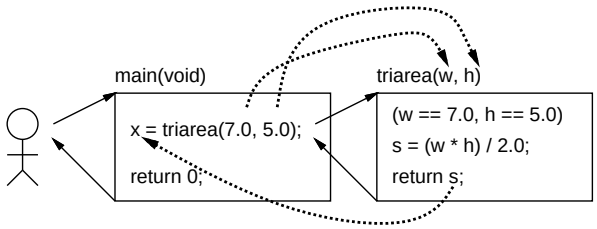



図 1: 関数の呼び出しと戻り

このほか、次のことも重要です。

[3] もしできない場合は、できる状態になるまでは、各自で自習してきてください。情報部会のサイト `https://joho.g-edu.uec.ac.jp/joho/` に基礎プログラミングおよび演習のテキストとビデオがあります。

- 呼び出し時に渡した値の並び (**実引数**、actual arguments) は、関数の定義冒頭に書かれた変数の並び (**仮引数**、formal argument、パラメタ) に順に対応させられる。図 1 の場合、`triarea` に入ったときにそのパラメタ `w` の値は渡された 1 番目の値 7.0 に、そして `h` の値は渡された 2 番目の値 5.0 になっている。
- **return** 文「`return 式;`」が実行されると、直ちにその関数の実行を終わって読んだ箇所に戻る。式の値は、関数の値となるので、それを計算に使ったり変数に代入できる。返値の型が `void` と指定された関数の場合は値を返さないので式も指定しないが、直ちに実行を終わることは同じである。なお、`main` は呼び出し元に整数を返すことになっており、正常終了のときは 0 を返しそうでない解きは 0 以外を返す約束になっている。[4]
- コンパイラは引数の個数と型、および返値の型が合っていることをチェックするが、C 言語ではその際「それまでに処理したソースコードの情報だけを使う」という制約がある。このため、上のように「関数定義が先にあり、その後で呼び出す箇所がある」場合は大丈夫だが、関数 `main` を先に書くなら、その前に次のような `triarea` の**プロトタイプ宣言** (prototype declaration) を置く必要がある。

```
double triarea(double w, double h);
```

  プロトタイプ宣言は、関数の本体部分「{ … }」を削除し、代わりに「`;`」を置いたものである。

## 2.2 基本概念の復習

例題に出て来るそのほかの概念をひととおり説明します。

- C 言語で扱う**基本型** (primitive type) として `char`(1 バイト==8 ビットの整数)、`int`(整数)、`long`(ビット数の多い整数)、`float`(実数)、`double`(倍精度実数) がある。整数は符号つき (2 の補数表現) だが、**unsigned** という修飾を前につけると符号なし整数にできる。
- 値を返さない関数は **void** 型として定義/宣言する。関数のパラメタが 0 個の場合はパラメタ部に `void` と書く (上の例の `main` がそう)。
- 変数はすべて**宣言** (declaration) してから使う必要がある。宣言は「型指定 変数,…;」の形をしている。それぞれの変数の後に「=式」をつけることで、宣言と同時に**初期値** (initial value) を入れてもよい。
- 「**式** (expression)」は値を計算することを表す。「`3.14`」などの**定数** (literal — その定数の表す値を意味する)、「`x`」などの**変数** (variable — 変数に格納されている値を表す)、および先に述べた「関数呼び出し」(関数が返す値を表す) をもとに、**演算子** (operator) でさまざまな演算をする。演算子には「`+, -, *, /, %`」(加減乗除と剰余)、「`++, --`」(整数変数を 1 増やす/減らす)、「`==, !=, >, >=, <, <=`」(比較)、「`&&, ||, !`」(かつ、または、〜でない)、「`&, |, ~`」(整数をビット列とみなしての and, or, not) などがある。そして「`=`」(**代入**、assignment) も演算子であり、結果として代入した値を返すので、「`x = y = 0`」などとも書ける。
- 関数本体には**文** (statement) の並びを書く。式を文として書くときは「式;」のように「`;`」をつける必要がある (**式文**、expression statement)。関数呼び出しだけ書く場合も式文になる。上の例では変数宣言と return 文以外はすべて式文である。
- 出力関数「`printf("書式文字列", 式,…)`」は、基本的に**書式文字列** (format string) をそのまま出力するが、その中に「`%`○」という形のもの (書式指定) があると、そこに後ろに指定した式の値を埋め込む。主要な書式指定に「`%d`」(整数を十進で出力)、「`%x`」(16 進で出力)、「`%f`」(実数を小数点つき形式で出力)、「`%e`」(指数形式で出力)、「`%g`」(おまかせで出力)、「`%c`」(文字を出力)、「`%s`」(文字列を出力) がある。「`%8d`」のように出力の幅を指定することもできる。

[4]実際はこの `main` が返した値を使うことはあまり多くありませんが。

- 入力関数「scanf("書式文字列", 式,…)」は、書式指定に従って入力をおこなう。主要な書式指定に「%d」(整数入力)、「%c」(文字入力)、「%f」(float 入力)、「%lf」(double 入力) がある。対応する式は「変数のアドレス」でなければならない。変数のアドレスを取るのには、**アドレス演算子** (addressing operator)「&」を用いて「&変数名」でできる (そのほかポインタ値やポインタ演算などでもできる)。

### 2.3 論理値の扱い

C 言語には**論理型** (boolean type、はい/いいえを表す型) が無く、int で代用し、「0」が「いいえ」、「1」が「はい」を表します。しかし今日の多くの言語では論理型として「bool」、値として「false」「true」を使うようになっているので、次のような定義をする C プログラマも多数います。

```
#define bool int
#define true 1
#define false 0
```

毎回そうするのも面倒なので、#include <stdbool.h>を書いておくことで上記と同じに使えるようになります。以下でもそのようにします (論理値と整数を分けて考えておく方が間違えにくいので)。

### 2.4 制御構造の復習と文法

関数定義の { … } の中には文の並びが入ります。具体的にどのようなものが書けるかについては、プログラミング言語の**文法** (syntax specification) によって定められています。ここでは文法を読む練習として、「文」の定義を見ましょう (分かりやすさのため簡略化しています)。なお、「$A|B$」は「A または B」、「$[A]$」は「A があってもなくてもよい」を意味します。

文 ::= 変数定義 | ; | 式 ; | if 文 | while 文 | for 文
　　| return [ 式 ] ; | break ; | continue ; | { 文… }
if 文 ::= if( 式 ) 文 [ else 文 ]
while 文 ::= while( 式 ) 文
for 文 ::= for( [ 変数定義 | 式 ] ; [ 式 ] ; [ 式 ] ) 文

「;」だけの文がありますが、これは**空文** (empty statement) といい、何も動作をしない文です。なんでこれが必要かというと、たとえば **if** 文で次のような場合のためです。

```
if(x <= 10)
  ;   // x は 10 以下なので何もしない
else
  printf("x too large: %d\n", x);
```

文法を見ると「if と else の間には 1 つ文がある」となっているので、何も書かないわけにはいかないので、それで空文を入れています。しかしこれまでそんなものは使わないで済んでいたが…そうでしょう。つまり、常に「{ … }」を使えば、それ自体がちょうど 1 つの文ですし、その中には文が 0 個でもよいので、上のようなことを気にしないでよいのです。

```
if(x <= 10) {
  // x は 10 以下なので何もしない
} else {
  printf("x too large: %d\n", x);
}
```

なので、こちらのスタイルを使うことを勧めます。あと、if-else if の連鎖が文法に無いですね? それは「else の直後にすぐ次の if 文を書いた」だけで、それで上の文法に合致しています。

```
if(x > 0) {
  printf("positive.\n");
} else if(x < 0) {
  printf("negative.\n");
} else {
  printf("zero.\n");
}
```

次にループですが、C 言語では **for** 文が **while** 文と同様、任意の条件を指定する文であることが特徴です。たとえば次の 2 つは同じ動作です。

```
i = 0;
while(i < length) { b[i] = a[i]; ++i; }

for(int i = 0; i < length; ++i) { b[i] = a[i]; }
```

1 つだけ小さい違いがあります。それは、ループの中で **continue** 文 (次の周回に進めという命令) を使ったとき、for 文の場合は「`++i`」のところに飛びますが、while 文であれば次の条件テストにすぐ進む、というところです。なお、ループを脱出する **break** 文の場合は両者の差はありません。

あと、for 文の初期化部分には (C99 以降では)「変数定義」が書けます。上の断片でもそうなっています。ただし、ここで定義した変数のスコープはこの for 文の中だけなので注意 (ループを出た後で `i` の値を参照することはできなくなるという意味です)。

では演習に進む前に、2 番目の例題として「`n` を入力し、`n` が素数か否かを出力する」というものを示しましょう。

数学ライブラリを使うので `math.h` を取り込むこととコンパイル時に「`gcc8 prime.c -lm`」のように追加指定が必要です。[5]

```
// prime.c --- see if an integer is a prime.
#include <stdio.h>
#include <math.h>      // sqrt を使う場合必要
#include <stdbool.h>   // true, false, bool を使う場合必要
bool isprime(int n) {
  int limit = (int)sqrt(n);
  for(int i = 2; i <= limit; ++i) {
    if(n % i == 0) { return false; }
  }
  return true;
}
int main(void) {
  int n;
  printf("n> "); scanf("%d", &n);
  if(isprime(n)) { printf("%d is a prime.\n", n); }
  else           { printf("%d is not a prime.\n", n); }
  return 0;
}
```

ある数が素数かどうか調べるには、$2 \sim \sqrt{n}$ までの数で割ってみればよいわけです。**平方根** (square root)「`sqrt(n)`」は当然実数を返すので、実数値を切り捨てて整数に変換するためにキャスト演算「`(int)`」を使用して、結果を変数 `limit` に入れています。

[5] C99 対応の GCC バージョン 8 を入れてもらいましたので、それを使います。コマンドは「`gcc8`」です。

ループでは 2 から初めてその `limit` まで割り切れるかを調べています。`%`は剰余演算なので、剰余が 0 なら割り切れたことになります。return 文は実行すると直ちに値を返してその関数を終わるので、ループの途中でやめる (その先を調べないで済ませる) ことができるわけです。最後まで割り切れなければ素数なので「はい」を返します。

**演習 1** まず「三角形の面積」「素数判定」を打ち込み動かせ。動いたら次のようなプログラムを作れ。

a. 直角三角形の直角をはさむ 2 辺の長さを入力し、斜辺の長さを出力する。
b. 整数 $n$ を入力し、$2^n$ を出力する。$n \geq 0$ のときは整数、そうでないときは実数形式で出力すること。
c. 整数 $n$ ($n > 1$) を入力し、$n$ の素因数分解を表示する。たとえば 60 を入力したら「2 2 3 5」を出力する (出力の順番や形式は任意)。
d. 整数 $n$ を入力し、$n$ 以下の素数を出力する (出力の順番や形式は任意)。
e. 整数 $n$ を入力し、何らかの数列 (選択は任意) の最初の $n$ 項を出力する。

# 3 PostScript を用いた描画 option

## 3.1 PostScript を用いた描画ライブラリ

数値ばかり扱うのも飽きるので、後半は **PostScript** 言語 (以下 PS) を用いた描画のためのライブラリと API を題材に取り上げます。PS については「コンピュータリテラシ」において取り上げていますので、学んでいない人は詳しくはそちらの資料を見てください。必要最低限のことは以下で説明します。

以下で使い方を説明するライブラリは、BoundingBox 指定を持った PS 形式 (**EPS**、Encapsulated PostScript) を出力するためのものです。PS そのものが図形を記述する機能を用意していますが、そのままだと使い慣れないとやりにくいので、その上に C 言語むけ API をかぶせて使いやすくしています (細かいことは直接 PostScript の命令を使って指定します)。

API を使うためのヘッダファイル `eps.h` を示します (実装部分 `eps.c` は付録に掲載しました。またその中身は読めば読めると思うので細かくは説明していません)。

```
// eps.h --- eps library API
void eps_open(char *fname, int w, int h);
void eps_close(void);
void eps_cmd(char *cmd);
void eps_num(double val);
void eps_drawline(double x0, double y0, double x1, double y1);
void eps_drawrect(double x, double y, double w, double h);
void eps_fillrect(double x, double y, double w, double h);
void eps_drawcircle(double x, double y, double r);
void eps_fillcircle(double x, double y, double r);
int eps_newfont(char *font, double size);
void eps_puts(int id, double x, double y, char *s);
```

API に含まれる関数の機能は次の通りです。

- `eps_open` — 出力するファイル名と描画面の大きさを指定する。単位は pt (1pt = $\frac{1}{72}$inch)。A4 版の紙に収まるには「600 × 850」程度までにするのがよい。
- `eps_close` — 描画を終わる (ファイルを完成させる)。
- `eps_cmd` — 任意の PS コマンドをそのまま出力。PS の細かい機能を使いたいときは基本的にこれを使う。

- eps_num — PS コマンドの中に C 言語で計算した数値を混ぜたいとき、その数値を出力するのに使う。
- eps_drawline、eps_drawrect、eps_fillrect、eps_drawcirlce、eps_fillcircle — 指定した座標間の線分を描く、または指定した位置と大きさの長方形/ 円を描く/塗りつぶす。長方形の指定は「左下隅の XY 座標、幅、高さ」、円の指定は「中心の XY 座標、半径」でおこなう。
- eps_newfont — 文字を描画するのに使うフォントを準備する。フォント名とポイント数を指定する。PS で標準で (必ず) 使えるフォントは Times-Roman, Times-Italic, Times-Bold, Times-BoldItalic, Helvetica, Helbetica-Oblique, Helvetica-Bold, Helvetica-BoldOblique, Courier, Courier-Oblique, Courier-Bold, Courier-BoldOblique がある (Italic、Oblique は斜体を表す)。結果として「フォント番号」が返され、その番号を eps_puts で使う。
- eps_puts — フォント番号、XY 座標、文字列を指定することで、指定位置に指定した文字を指定したフォントで描画する。

### 3.2 PostScript の主要なコマンド

参照に便利なように、直接使いそうな PS のコマンドを示しておきます。

- gsave、grestore — グラフィクスの状態を保存/ 復元する。以下で説明する座標変換、色や線幅の変更を行なったあと、元に戻すには、gsave しておいてから変更を行ない、戻したいところで grestore する。
- $R$ $G$ $B$ setrgbcolor、$H$ $S$ $S$ sethsbcolor — 描画に使う色を RGB または HSB で指定。これらの値は 0.0-1.0 までの実数で、RGB の場合は赤/緑/青の強さ、HSB の場合は色相/ 彩度 (鮮やかさ)/明度 (明るさ) を意味する。
- $B$ setgray — 描画に使う色 (白黒) を明るさで指定。B は 0.0-1.0 までの実数で、真っ黒から真っ白までの間の灰色を指定する。
- $T_x$ $T_y$ translate、$S_x$ $S_y$ scale、$D$ sotate — 描画の座標の変換。原点を $(T_x, T_y)$ に移動、X および Y 方向に $S_x$ および $S_y$ 倍、原点を中心に反時計回りに $D$ 度回転がおこなえる
- $W$ setlinewidth — 線幅を変更。単位は pt。

### 3.3 eps ライブラリの使用例

それでは実際にライブラリを使った例を示しましょう。このプログラムで描画した結果は図 2 なので、照合しながら見てください。

```
// testeps.c --- demonstration of eps library.
#include <stdio.h>
#include "eps.h"

int main(void) {
  eps_open("out.ps", 480, 480);
  eps_cmd("240 240 translate");
  eps_drawline(-200, 0, 200, 0);
  eps_drawline(0, 200, 0, -200);
  for(int i = 1; i <= 8; ++i) {
    eps_num(i*0.1); eps_cmd("setgray");
    eps_cmd("4 setlinewidth");
    eps_drawrect(i*20, i*20, 30, 30);
    eps_num(i*0.1); eps_cmd("1.0 1.0 sethsbcolor");
```

```
    eps_fillcircle(-i*20, -i*20, 15);
  }
  int f1 = eps_newfont("Courier", 20);
  eps_puts(f1, -180, 50, "This is a pen.");
  int f2 = eps_newfont("Helvetica", 30);
  eps_puts(f2, 20, -50, "How are you?");
  eps_close();
  return 0;
}
```

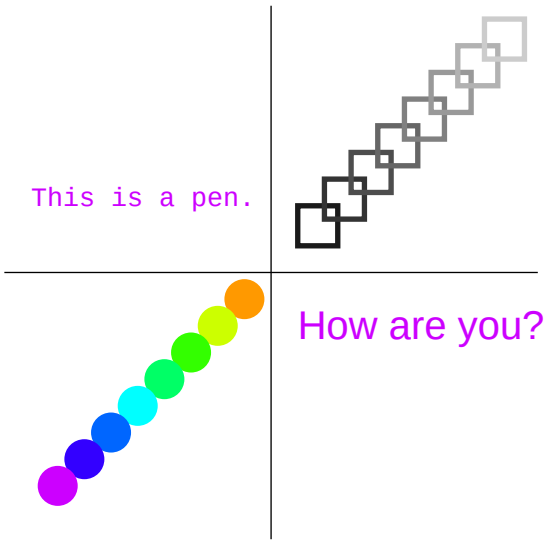



図 2: eps 例題プログラムの出力

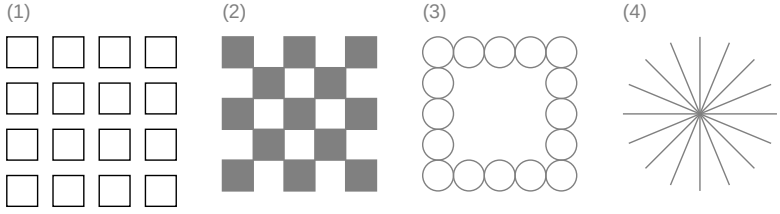
コメントで書いた通り、さまざまな機能をひととおり呼んでいます。色の設定と線幅の設定は PS のコマンドを直接出力しますが、そのときのパラメタを C の変数で設定する箇所は eps_num を用いて指定しています。

これを動かすには、このプログラムのファイル (たとえば testeps.c) に加えて、eps.h と eps.c が必要です。動かすコマンドは次のようになります。

```
% gcc8 testeps.c eps.c      ←必要に応じて -lm なども指定
% ./a.out                   ←このプログラムは入力なし
% gv out.ps                 ←画面で直接見る場合。または
% convert out.ps out.pdf    ← PDF やその他の形式に変換し利用
```

**演習 2** 上の例題をそのまま動かせ。動いたら色をや図形の位置、個数などを変更して結果を確認すること。それができたら、以下のプログラムを作れ。

a. 次の図のような絵を作成する。色や線幅は自由にしてよい。



b. 数学関数 ($sin$、$cos$、$tan$ など)、多項式による関数 ($y = x^2 - 4$、$y = \frac{1}{3}(x^3 - x)$ など)、その他好きな関数を 1 つ選びグラフを描画する (細かい折れ線で近似する)。

c. お天気サイト等から自分の居住地 (または出身地) に関する何らかのデータ (月毎平均気温、月毎降水量、月毎の晴れ日数など) を取得し、分かりやすいグラフにする。

d. その他 eps ライブラリを利用して面白い描画をおこなう。

### 本日の課題 1a

「演習 1」「演習 2」で動かしたプログラム 1 つを含むレポートを本日中 (授業日の 23:59 まで) に提出してください。

1. sol または CED 環境で「`/home3/staff/ka002689/prog19upload 1a` ファイル名」で以下の内容を提出。文字コード UTF8。「ファイル名」の代わりに「`-show`」と指定すると提出内容が確認できる。以後毎回同様。
2. 学籍番号、氏名、ペアの学籍番号 (または「個人作業」)、提出日時。名前の行は先頭に「@@@」を付けることを勧める。
3. プログラムどれか 1 つのソースと「簡単な」説明。
4. レビュー課題。提出プログラムに対する他人 (ペア以外) からの簡単な (ただしプログラムの内容に関する) コメント。
5. 以下のアンケートの回答。

   Q1. C 言語のプログラミングは好き/嫌いどちら? 理由は?

   Q2. eps ライブラリについてどのように思いましたか。

   Q3. リフレクション (今回の課題で分かったこと) ・感想・要望をどうぞ。

### 次回までの課題 1b

「演習 1」「演習 2」(ただし 1a で提出したものは除外、以後も同様) の小課題全体から選択して 2 つ以上プログラムを作り、レポートを提出しなさい (図形やグラフなどを 2 つ作って 2 個ということでもよい)。できるだけ演習 2 からも選ぶこと。レポートは次回授業前日 23:59 を期限とします。

1. sol または CED 環境で「`/home3/staff/ka002689/prog19upload 1b` ファイル名」で以下の内容を提出。
2. 学籍番号、氏名、ペアの学籍番号 (または「個人作業」)、提出日時。名前の行は先頭に「@@@」を付けることを勧める。
3. 1 つ目の課題の再掲 (どの課題か分かればよい)、プログラムのソースと「丁寧な」説明、および考察 (課題をやってみて分かったこと、分析、疑問点など)。
4. 2 つ目の課題についても同様。
5. 以下のアンケートの回答。

   Q1. プログラムを作るという課題はどれくらい大変でしたか?

   Q2. eps ライブラリを使ってみてどのように感じましたか。

   Q3. リフレクション (今回の課題で分かったこと) ・感想・要望をどうぞ。

## 4　付録: eps ライブラリの実装

eps ライブラリの実装はそれほど大したものではないので、掲載しておきます。PostScript の細かい機能を使っていますが、それはここでは説明しません。適宜調べてください。C 言語関係でまだ学んでいない機能としては、任意のファイルへの書き出しを使います。その概略は次の通り。

```
FILE *fd = fopen("ファイル名", "wb"); ←ファイルを書き出しオープン
...
fprintf(fd, "書式文字列", 引数…);  ← printf と類似。何回も使う
...
fclose(fd);                    ←後しまつ。最後に必ず呼ぶ必要あり
```

この API は実は stdio.h で宣言ずみです。fopen はファイルを「開く」(読み書きの準備をすることをそう呼ぶ) 操作で、ここでは書き出しモードとします (wb は write binary)。返値はファイル型のポインタ値で、その中身は知らなくてもよいです (fprintf や fclose にそのまま渡すだけなので)。

fprintf は printf そっくりですが、ただし最初の引数としてファイル型のポインタ値を受け取ることで、その対応するファイルへの書き出しを行なうところが違います。fclose はファイルを「閉じる」(処理を終了する) もので、これによりファイルが完成します。

```
// eps.c --- eps library implementation
#include <stdio.h>
#include "eps.h"
static FILE *fd = NULL;
static int fontid = 0;

void eps_open(char *fname, int w, int h){
  fd = fopen(fname, "wb");
  fprintf(fd, "%%!PS-Adobe-2.0\n%%%%BoundingBox: 0 0 %d %d\n", w, h);
}
void eps_close(void) {
  fprintf(fd, "showpage\n"); fclose(fd); fd = NULL;
}
void eps_cmd(char *cmd) {
  fprintf(fd, "%s\n", cmd);
}
void eps_num(double val) {
  fprintf(fd, "%.2f ", val);
}
void eps_drawline(double x0, double y0, double x1, double y1) {
  fprintf(fd, "newpath %.2f %.2f moveto %.2f %.2f lineto stroke\n",
          x0, y0, x1, y1);
}
static void rect(double x, double y, double w, double h) {
  fprintf(fd, "newpath %.2f %.2f moveto %.2f 0 rlineto\n", x, y, w);
  fprintf(fd, " 0 %.2f rlineto %.2f 0 rlineto closepath\n", h, -w, -h);
}
void eps_drawrect(double x, double y, double w, double h) {
  rect(x, y, w, h); fprintf(fd, "stroke\n");
```

```
}
void eps_fillrect(double x, double y, double w, double h) {
  rect(x, y, w, h); fprintf(fd, "fill\n");
}
static void circle(double x, double y, double r) {
  fprintf(fd, "%.2f %.2f %.2f 0 360 arc closepath\n", x, y, r);
}
void eps_drawcircle(double x, double y, double r) {
  circle(x, y, r); fprintf(fd, "stroke\n");
}
void eps_fillcircle(double x, double y, double r) {
  circle(x, y, r); fprintf(fd, "fill\n");
}
int eps_newfont(char *font, double size) {
  fprintf(fd, "/%s findfont %.2f scalefont /font%d exch def\n",
          font, size, ++fontid);
  return fontid;
}
void eps_puts(int id, double x, double y, char *s) {
  fprintf(fd, "font%d setfont %.2f %.2f moveto (%s) show\n",
          id, x, y, s);
}
```