基礎プログラミング演習 II 教材 (#3)

■ データ型（実数と整数）

（教科書 p.49「型」を参照）

□ 型変換、キャスト

まず定数の表記ですが、1, 2, 100 などは整数、1.5 など小数点が付けば実数として扱われます。つまり 2 は整数ですが 2.0 は実数となります。整数と整数の演算（ex. 1+100）結果は整数、実数どうし（10.0+20.5）は実数となります。問題はその「型」が混在するときです。

int 型変数は整数しか扱えません。そこで int 型変数に実数を代入すると、自動的に型変換が行われ、小数部が切り落とされて整数部だけが代入されます。
つまり `int i; i=123.456;` なら、i には少数以下が切り落とされた 123 が代入されます。

実数変数に整数を代入した場合は、単純に実数化された値が入ります。
`double a; a=123;` なら、123 が実数化されて 123.0 となって a に代入されます。

計算式に整数、実数の値や変数が混在していた場合は、精度の高い方に合わせて型変換が行われてから計算が行われます。`double a; a=1.5 * 3;` なら、1.5 * 3 は 1.5 * 3.0 として計算され、4.5 が a に代入されます。

注意が必要なのは /（割り算）で、整数と整数の割り算は整数となって余りは捨てられますが、実数と実数、または実数と整数の割り算では可能な限りの精度で小数点以下まで求められます。

つまり 10 / 4 は 2 ですが、10.0 / 4.0 は 2.5 です。変数を用いた計算でも同じことで、

```
int i, k; double a, b;
i=10; k=4; a=10.0; b=4.0;
```

の場合、i / k は 2 ですが、a / b は 2.5 です。i / b や a / k のように実数と整数を混在させた場合は、整数側が実数化されて計算され、結果はともに 2.5 になります。

これらは暗黙の型変換と呼ばれますが、明示的に指示して行うこともできます。
もし、i / k の計算を実数化して行い、2.5 という結果を実数変数に代入したい場合は、

```
double x; x = (double)i / (double)k;
```

と書きます。
この、値の直前に () で囲んで型名を明示指定する方法を「キャスト」と呼びます。

> キャストの有効範囲がどこまでか不安になるような場合があり得ます。例えば、
> `x = (int) a / k * 100;` のようなケースは、a だけキャストされるのか、全体の計算結果に最後に一度だけキャストされるのか、不安に思うかも知れません。そうしたときは、
> `x = (int)(a / k * 100);` のように、カッコをうまく使って記述すると良いでしょう。

□ 定数における型の明記

123.456 は実数でも double 型とみなされます。float 型と明記したい場合は 123.456f と最後に f をつけて表記します。（サンプルプログラムの f = 300.0f; を参照。）
実数はまた 1.2e-5; というように表記できます。（ 1.2×10^{-5} つまり 0.000012 ）
（興味のある受講生は教科書 p.38 も参照。整数定数の 8, 16 進表記法がある。）

□ データの型に合わせた変換文字

printf() を利用する際はデータの型に合わせて変換文字を指定しなければなりません。整数は %d, 実数は %f です。

実験：右のプログラムを入力して実行し、その結果を表示させて結果を確認してください。また、プログラムを修正して、i/30, i/30.0, i*d, i/30*d, i/d*30 がそれぞれどのような結果になるか試してください。納得できますか？
（他にも例えば整数を %f で表示させると？その逆では？なども試してください。）

```
short s;
int i;
float f;
double d;

s=100;       i=200;
f=300.0f;   d=400.0;

printf("short  %d\n",s);
printf("int    %d\n",i);
printf("float  %f\n",f);
printf("double %f\n",d);
```

□ 変数の型と精度

変数の型によって格納できる値の範囲・精度が異なります。

実験：精度（有効桁数）が型によって異なることを確認しましょう。上のプログラムに精度を超えると想像できる値を指定して結果を見てください。（教科書 p.49 表 3-1 参照）

□ データの型に合わせた変換文字列と桁数指定

整数は %d, 実数は %f あるいは %e が利用できます。また、%10d のように、% と変換文字の間に桁数の指定などができます。以下に代表的な変換文字列を示します。

	意味	使用例	その結果
%d	整数を表示	printf("[%d]\n",10);	[10] 桁数不定
		printf("[%5d]\n",10);	[10] 5 桁で表示。不足分は空白。
		printf("[%05d]\n",10);	[00010] 5 桁。不足はゼロで埋める。
%f	実数を表示	printf("[%f]\n",12.345);	[12.345000] 桁数不定
		printf("[%9.5f]\n",12.345);	[12.34500] 小数点含めて全体が 9 桁、小数以下が 5 桁。
%e	実数を表示	printf("[%e]\n",12.345);	[1.234500e+01] 浮動小数点で表示

また、scanf( ) 関数で実数型に値を入力する場合は、float 型変数であれば %f、double 型であれば %lf を変換文字として指定します。（つまり明確に型を合わせなければなりません）

```
float f; double d;
scanf(“%f %lf”, &f, &d);
```

□ 誤差：if 文による判定

右のプログラムは 100 回ループすると停止するように見えますが、実際には if 文の条件は成立しません。
しかし 0.1 ではなく 1.0 ずつ加算すれば停止します。
これは 0.1 が 2 進での浮動小数点表現では無限小数（割り切れない数）になるために生じる誤差が原因です。
制御文字を %50.45f などとして printf() すれば確認できます。

```
double d;
d=0.0;
while(1) {
  if(d == 10.0) break;
  printf("%f\n", d);
  d+=0.1;
}
```

このような場合に対処するため、実数では同一性判定は一定の範囲を定めて行うのが安全です。
`double e; e=0.1e-12;` などと非常に小さな値を用意して、
`if((d > 10.0-e)&&(d < 10.0+e)) break;` などとして判定します。

■ 数学関数

□ 数学関数の使用例

C 言語には sin( ) 関数や cos( ) 関数など、数学的な計算を行ってくれる関数がひと揃い用意されています。こういった関数を数学関数と呼んでいます。代表的な関数としては以下のようなものがあります。

sin　正弦を求める	sqrt　平方根を求める
cos　余弦を求める	pow　累乗を求める
tan　正接を求める	log　対数を求める
fabs　絶対値を求める	exp　指数を求める

他の関数、詳しい使い方などについては教科書 p.466 math.h を参照すると良いでしょう。

y = sin(r); のようにして使えば y に角度 r の時の正弦を計算してくれます。

ただし sin 等の三角関数に与える引数は、
・ double 型の実数であること
・ 角度は、ラジアン単位（弧度）で指定する
ことに注意して下さい。
sin 関数の戻り値は double 型で返されます。

C 言語における三角関数はどれもラジアン単位（弧度法）です。(度数法で言う 360 度を 2 πラジアンとする)
つまり sin 関数に 60 度に相当する角を渡したければ、
60 / 360 * 2 * 3.141592 ≒ 0.018278
を与えることになります。

□ 数学関数を使うために

数学関数を使うために先頭に以下の一行を書いて下さい。

#include <math.h>

これによって数学関数を定義したヘッダファイル math.h がプリプロセッサによって取り込まれます。書かなかった場合のエラーを一度見ておくと良いでしょう。
Termnal で man 3 sin などとすると、どの関数がどのヘッダを必要とするか分かります。

また、MacOSX では不要ですが、幾つかのコンパイラでは -lm オプションが必要になります。

$ cc sample.c -lm

これは数学関数のためのライブラリを使う指示で、これがないと下記のように「sin という名前は未定義 (undefined) である」といったエラーが出る場合があります。

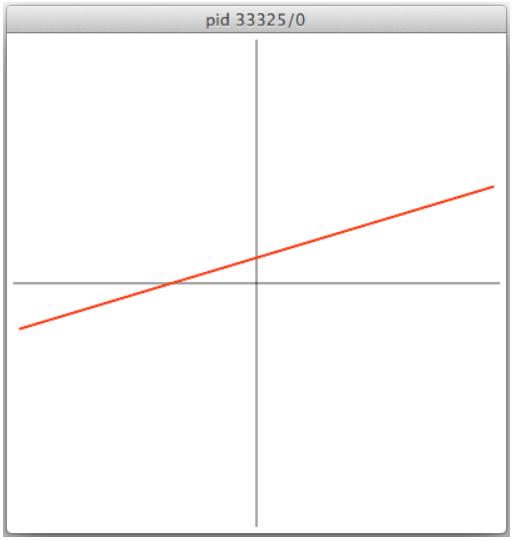
```
$ cc –o sample sample.c
/tmp/ccHZhcFU.o: In function `main':
/tmp/ccHZhcFU.o(.text+0x3f): undefined reference to `sin'
collect2: ld returned 1 exit status
$
```

□ 前準備（一次関数のグラフを描く）

以下に y = ax + b　(a=0.3, b=20.0) の一次関数グラフを描くプログラムを示します。

押さえて欲しいポイント：
- HgLine( ) 関数で座標軸を描いています。
- HgBox( ) 関数で大きさ 1x1 の四角形を描くことで、計算した座標位置に点を打っています。
- 原点をグラフィクスウィンドウの真ん中（200, 200）に置いているので、HgBox( ) 関数の x, y 座標位置指定にはそれぞれ 200.0 を加算しています。

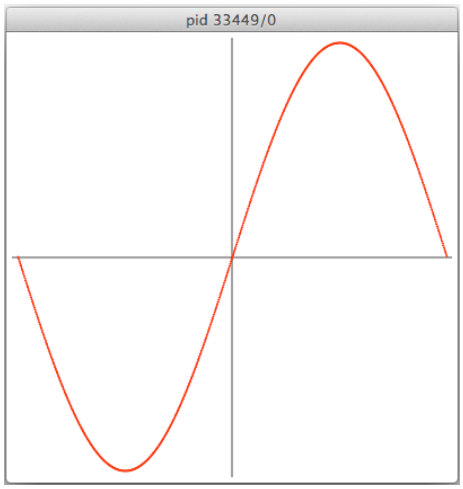
プログラムと実行結果




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

int main() {
  int x;
  double y, a, b; // y = ax + b

  HgOpen(400.0, 400.0);
  HgSetWidth(1.0);
  HgSetColor(HG_BLACK);

  HgSetFillColor(HG_WHITE);
  HgLine(5.0, 200.0, 395.0, 200.0);
  HgLine(200.0, 5.0, 200.0, 395.0);

  a=0.3;
  b=20.0;
  HgSetColor(HG_RED);
  for(x=-190; x<190; x++) {
    y=a * x + b;
    HgBox(x + 200.0, y + 200.0, 1.0, 1.0);
  }

  HgGetChar();
  HgClose();

  return 0;
}
```

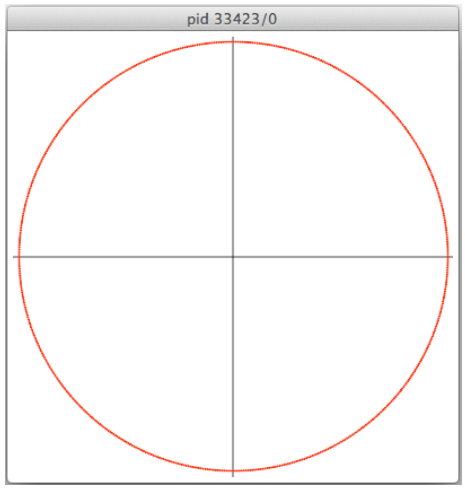
□ 課題 1.

右図のような sin カーブを描いてください。



□ 課題 2.

右図のような円を描いてください。ただし HgCircle 関数などを使うのではなく、円周上に多くの点を打って円に見えるようにします。



$y = \sqrt{r^2 - x^2}$ として y を求めるのではなく、角 θ を 0〜2π まで変化させ、sin, cos 関数を用いて x, y を求めて描くように。この説明が分からない場合は聞いてください。

□ 定数

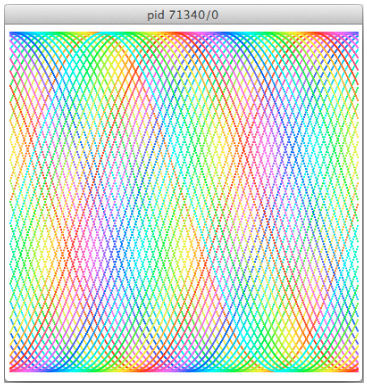
math.h をインクルードすることによって、幾つかの数学関数で利用できる定数が定義されます。
例えば円周率は下記のように定義されています。

```
#define M_PI  3.14159265358979323846264338327950288
```

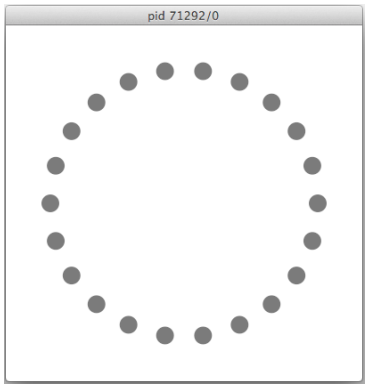
(興味のある受講生は math.h ヘッダファイルを参照すると良いでしょう。ただし Xcode における場所はこんなところになります。探してください。`/Applications/Xcode.app/Contents/Developer/latforms/MacOSX.platform/Developer/SDKs/MacOSX10.10.sdk/usr/include/math.h`)

□ 課題 3.

三角関数など数学関数を利用して、何か美しい図形を描いて下さい。
例えば下記のようなものなど。各自工夫して、きれいなものを作って下さい。(二種類)

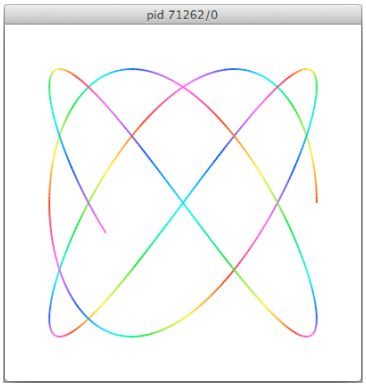


sin カーブを少しずらしながら
色を変えて描いたもの

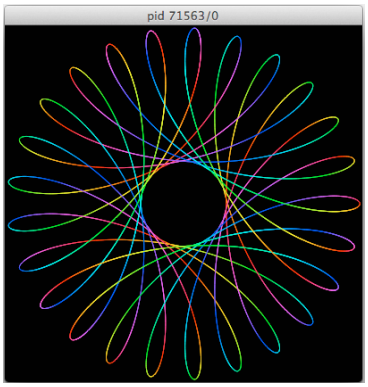


円周上に中心位置をずらし
ながら図形を描いたもの

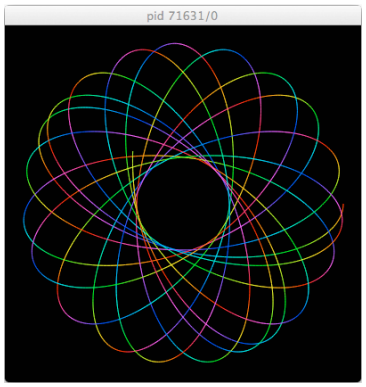
スパイラル



リサージュ曲線（描いている途中）
徐々に色を変えてアニメーションとしてみた



スピログラフその 1



スピログラフその 2（描いてる途中）