プログラミング演習 A 教材 (#9)

■ 関数

プログラムのなかの一部分を機能ごとに切り出し、関数として分けて記述することができます。これによってより分かりやすいプログラムの記述ができます。またプログラムを機能ごとの単位に仕分けて書くことによって、機能単位の再利用が容易になり、共通の関数を使った別のプログラムを簡単に作れるようになります。大規模なプログラムを作る重要な手法の一つです。

□ 簡単な関数の例

以下に階乗 (factorial) を計算するプログラムの例を示します。入力し、実行して、たとえば 5 の階乗が正しく 120 (5 * 4 * 3 * 2 * 1) になることを確認して下さい。

入力処理

計算処理

出力処理

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

  printf("n=");
  scanf("%d", &n);

  fac=n;
  for(i=n-1; i>1; i--) {
    fac*=i;
  }

  printf("factorial = %d\n", fac);

  return 0;
}
```

11 以上の階乗を求めようとすると桁あふれを起こして正しい結果が出ないので注意。

プログラムをよく見ると、入力処理、計算処理、出力処理に分けられることがわかります。
今回はこのうち計算処理の部分だけを別の関数にします。

階乗の計算処理だけを factorial という名前の関数に分けて書いた例を以下に示します。二つのプログラムの関係をメインルーチン、サブルーチンと表現することもあります。

```
int main() {
  int n, fac;

  printf("n=");
  scanf("%d", &n);

  fac=factorial(n);

  printf("factorial = %d\n", fac);

  return 0;
}
```

呼び出し

```
int factorial(int n)
{
  int i,f;

  f=n;
  for(i=n-1; i>1; i--) {
    f*=i;
  }

  return f;
}
```

戻り

メインルーチン（main 関数）　　サブルーチン（factorial 関数）

処理の流れ：
・ プログラムは main( ) の上から順に実行される。
・ main の途中で factorial( ) 関数を呼び出すと、
・ factorial 関数に処理が移る。
  ・ 階乗の計算を行い、
  ・ return でメインルーチンに処理が戻る
  ・ そのとき return f としたため、階乗の結果が factorial( ) 呼び出しの結果として戻される。
・ メインルーチン側で factorial( ) 関数の結果（戻り値）を fac 変数に代入。
・ そのまま main の処理を続行する。

□ 関数の定義、値の引き渡し方

例では factorial 関数は以下のようにして定義しました。

```
int factorial(int n)
{
  ……
}
```

`int factorial(…)` の先頭の型 `(int)` は、`factorial` 関数の戻り値（後述）の型を示す。

この関数をメインルーチン側から呼び出すときは、例えば以下のようにします。

```
factorial( 値 );
```

値の部分には数値(100 等)や変数(n 等)、それらを組み合わせた計算式などが入ります。この値がサブルーチン側（factorial 関数側）で用意した変数 n に引き渡され、サブルーチン内での実行が始まります。

書法：(メインルーチン側)
関数名`( 値 1, 値 2, … 値 z )`

書法：(サブルーチン側)
型 関数名`( 型 変数 1, 型 変数 2, … 型 変数 z )`

こうした受け渡しに用いるカッコ内の値（や変数）を引数（ひきすう）と呼びます。引数は複数用意できますが、そのときは呼び出す側と呼ばれる側の用意した値と変数が左から順に組み合わされて代入されていきます。引き渡す引数の数と型がきちんと一致していることが重要です。
上の書法では、メインルーチンで引数に設定した「値 2」は、サブルーチンで用意した「変数 2」に設定されますが「値 2」のデータ型は「変数 2」の型に一致していなければなりません。

□ 戻り値

`return` 文の実行によって、処理はサブルーチンからメインルーチンに戻ります。このとき、「 `return 値;` 」と書くことで関数の結果そのものを設定することができます。

つまり呼び出すときに `fac=factorial(n);` のようにしておくと、サブルーチン側の return によって戻された結果（ n の階乗）が変数 fac に代入されます。

注意：
return で戻す値は、関数自体の宣言文 (int factorial(…) の部分)の先頭にある関数自体の型宣言と一致していること。

□ 実際のプログラムの記述

実際の C プログラムのなかでは、サブルーチンはメインルーチンと同じファイルのなかにずらっと並べて書きます。

サブルーチン（呼び出される側）はメインルーチンの呼び出しより前に記述されていなければなりません。

つまりサブルーチンは main より前（上）に書くことになります。

関数から関数を呼び出すこともできますが、そうした場合も常に呼ばれる側がより前（上）に記述されている必要があります。

（例ではメイン、サブで同じ名前の変数を用いていますが、これらは別の名前を用いても構いません。）

```
#include <stdio.h>

int factorial(int n)
{
  int i,f;
  f=n;
  for(i=n-1; i>1; i--) {
    f*=i;
  }
  return f;
}

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

  fac=factorial(n);

  printf("factorial = %d\n", fac);
  return 0;
}
```

□ main の意味

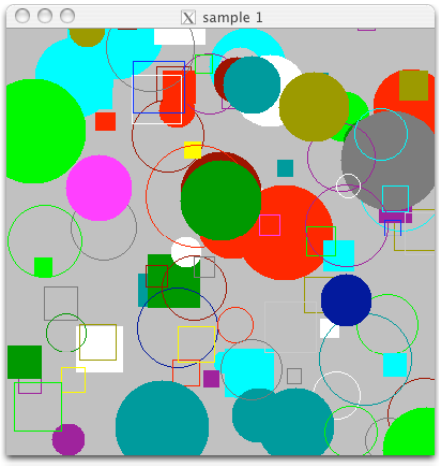
よく見ると main( ) もひとつの関数として機能していることが分かると思います。
実際、C 言語ではメインルーチンは特別な存在ではなく、単に実行時に最初に呼び出される関数がたまたま main( ) という名前に固定されている、というだけのことです。

□ 課題 1.

以前に作った「4 種類の図形を描き分ける」プログラムを、サブルーチンを使ってより読みやすくしてください。（次ページのひな形を見て空白を埋めていけばいいでしょう。）

ヒント：
・draw( ) という関数を作ってみました。
・draw( ) の引数は左から色、図形タイプ、座標位置(x, y)、図形の幅です。
・もし 1-4 以外の図形タイプを設定した場合は、draw( ) の戻り値を 0 以外にします
・dray( ) の戻り値次第で、必要ならエラー処理をしてメインルーチンの処理を中断します。

□ 課題 2.



draw( ) 関数を利用して違う動きをするプログラムを作りましょう。乱数を発生させる rand( ) 関数を利用して、ランダムな位置に 4 種類の図形を 100 個描くプログラムを作って下さい。

乱数発生のためのサンプルプログラムは次ページにあります。

サブルーチンをブラックボックスとして再利用できることを実感できると思います。

課題 1. ひながた

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

int draw(int win, int c, int type, float x, float y, float w)
{
  int rc=0; /* 戻り値をゼロに */

  … 略 …

  return rc;
}

int main() {

  … 略 …

  while( y < 380.0 ) {
    rc=draw(win, c, type, x, y, w); /* 描画 */
    if( rc != 0 ) { /* 戻り値がゼロでなければエラー処理 */
      printf(" 1 から 4 で指定してください。\n");
      gclose(win);
      return 0;
    }

      … 略 …

  }

  ggetch(win); /* キー入力を待つ */
  gclose(win); /* 描画ウィンドウを閉じる */

  return 0;
}
```

課題 2. のための乱数発生サンプルプログラム。(0-99 までの乱数を表示する。)

ポイント:

- 疑似乱数関数 rand( ) のために stdlib.h の include が必要。
- 最初に srand( ) 関数を一度だけ呼び出す必要あり。srand に与える引数はランダムな数値。Seed と言う。
- rand( ) 関数は 0 から 2147483647 までの整数を返す。

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

int main() {
  int seed, num, cnt;

  printf("random seed = ");
  scanf("%d",&seed);
  srand(seed);

  for(cnt=0; cnt<10; cnt++ ) {  /* 10 回ループ */
    num=rand() % 100; /* 余りは 0-99 */
    printf("%d\n",num);
  }

  return 0;
}
```