基礎プログラミング演習 II 教材　(#18)

■ 関数

プログラム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可能です。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。
実際、C 言語では printf( ), scanf( ), sin( ) など標準的に多くの関数が用意されており、受講生は既にこれらを便利に利用しています。こうした関数を自分たちで定義することができるわけです。

□ 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

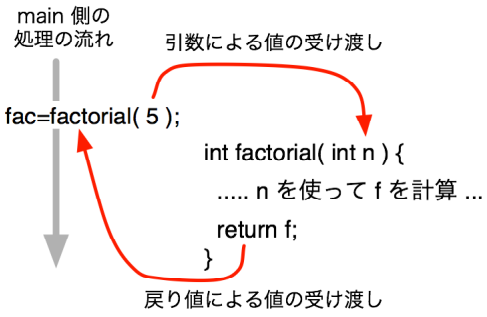
```
int factorial(int n)
{
  int i,f;

  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）

特徴：
・関数には名前があり、その名前で呼び出す。
・関数の処理は関数名に続くブロック { } に記述される。

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。
  ・ 引数によって値が渡され、
  ・ 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 として結果を戻す
・ 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。



> 階乗の定義から `i >= 1` とすべきかもしれませんが、結果が変わらないので、無駄な処理を省くために `i > 1` で済ませています。

□ 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。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。

□ 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 で戻す値の型は、関数自体の宣言文（ int factorial(…) の部分)の先頭にある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（縦に）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。

```
#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（それも整数型の）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.

入力した二数の階乗の和を作るプログラムを作ってください。

サンプルプログラムの factorial 関数を二度使えば良いでしょう。機能単位に切り出した関数を再利用することでプログラムの機能修正が容易に、また読みやすくなることを感じてください。

□ 複数の引数・一般書式

関数は以下のようにして定義します。引数は複数用意することができます。

```
型 関数名( 型 変数1, 型 変数2, … 型 変数z ) {
   ....
}
```

これをメインルーチン側は、**関数名(値1, 値2, ... 値z)** として呼び出します。
引数は複数用意できますが、そのときは呼び出す側の用意した値と、呼ばれる側の変数が左から順に組み合わされて代入されます。両者で値と変数の数と型が一致していることが重要です。

上の例では、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。「**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。

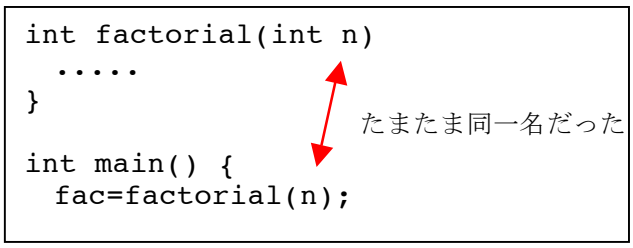
□ 用語の整理：仮引数・実引数

先に示したサンプルプログラムでは `fac=factorial( n );` のように変数一つを与えて呼び出しました。
しかし呼び出し側で設定するのは「値」であり、常に変数一つだけを記述するわけではありません。例えば以下のすべての書き方があり得ます。

```
fac=factorial( 10 );  定数
fac=factorial( i );  変数
fac=factorial( i + 2 );  計算式
fac=factorial( factorial( 3 ) + 3 );  関数（自分自身かもしれない）を含む計算式
```

サンプルプログラムでは、呼び出し側と関数定義側で同じ名前の変数(n)を使っていますが、これは偶然で、違った名前で問題ありません。

そもそもメイン側から引き渡されるのは変数ではなく「値」（変数だけでなく定数や式もあり得る）であることに注意してください。



★教科書 p.228 13.1.3 「引数とデータの引き渡し」参照
・maxPrint( ) 関数が二つの引数をとること（void は後述するので気にしない）
・用語（仮引数・実引数）の整理
・図 13.3 にあるように値が仮引数に対して渡されること
（例では変数から変数へのデータのコピーのように書かれているが、そもそも変数を一対一で引き渡すとは限らず、渡されるのはあくまで「値」である事に注目）
・「値渡し」という用語の理解は待つべし（ポインタと参照渡しの概念の比較で理解すべきです）

（しつこいですが）良くある勘違い：
サブルーチン側で仮引数に用いた変数の値を変更しても、メインルーチン側で実引数に使った変数の値は変更されません。（実引数が a+2 の場合何が起きる？ a + b なら？と考えると解りやすい）

□ 課題 2.

二つの数字を scanf で入力し、大きい方を出力（表示）するプログラムを作成してください。二つの引数を与えると、大きい方を戻り値に設定して返す関数を作って実現してくださいね。

□ 課題 3.

150 1.05 などと、単価と消費税率の二数を入力すると価格を出力するプログラムを作成してください。この課題も計算を行う関数を作って実現してください。（消費税率は関数内に固定的に書かず、引数として関数に渡すようにしてください。単価と税率の型の違いに注意。なお余り実用性の無いプログラムですが、練習課題なのでその点は気にしないで下さい。）

□ 課題 4

入力した二数の階乗の和を作るプログラムを作ってください。ただし、課題 1. とは異なり、「二数を引数に与えると、それぞれの階乗を求め、その和を戻り値に設定して返す」関数を定義して実現するように修正してください。（既に作成した階乗を求める関数を再利用しましょう）

特に最後の課題は「関数を用いることでプログラムをできるだけ見通しの良いものにする」意義を学ぶものです。「動けば良い」では無く、より良いコードにする工夫を検討して下さい。