[プログラミング序論 II　9回目(最終回)　２００２・１１・１４]

これまでのまとめ

これまで、C言語の機能については一通り説明しました。まとめてみると、

- 配列、多次元配列
- 文字、文字の配列、文字列
- 関数定義の仕方、関数呼び出しの書き方、手続きとしての関数
- 再帰呼び出し
- いろいろな変数、局所変数、大域変数、静的変数
- 関数の引数は値渡しであること
- ポインタとは。ポインタの使い方、＊演算子、&演算子
- ポインタによる関数の値の返し方
- 配列とポインタとの関係、ポインタについての演算
- データ型とは
- 構造体、構造体の意味、定義の仕方、フィールドの参照、. 演算子
- 構造体の配列、参照、変数の宣言
- typedef 宣言とデータ型
- 構造体の引数、返り値の返し方、構造体へのポインター、->演算子
- sizeof 演算子
- 共用体

配列はデータをならべて、数字で取り出すためのデータ構造です。関数はある機能を人まとまりにして名前をつけておく機能で、プログラムは関数の集まりとして作ります。この２つはコンピュータを扱うためにはプログラミング言語になくてはならない機能です。特に、関数の再帰呼び出しの考え方は重要です。入力が決まっていれば、やるべきことをいちいち目の子でかいてそれ専用のプログラムをかくことができますが、プログラムを任意の入力でも動くようにすることは上手なプログラムを書く上で重要なことです。再帰的な関数は、自分を自分自身の一部で表現するもので、例えば、入力が n の時の動作を n-1 の時の動作を用いてかくなど、いろいろな場面に応用できます。

ポインタとはコンピュータを効率的に扱うための仕組みです。プログラムで宣言されたデータはメモリに格納されます。ポインタの実体はメモリ上のアドレスですが、考えとしてはなにかのデータを「ポイント」するものです。＊演算子や&演算子についてちゃんと理解しておきましょう。

構造体やそれをつかったデータ型は、プログラムを論理的にわかりやすくかくための仕組みです。プログラミングとは、コンピュータを動かす手順を書くものですが、プログラマにとってはコンピュータにさせたいことを表現する手段でもあるのです。わかりやすく表現するために、いろいろなデータを組み合わせて構造体を定義できます。構造体は、int や double など基本データ型と同じようにデータ型になります。

言語自体を学習することはもちろん必要ですが、使いこなせるようにならないと意味がありません。重要なのは、自分のやりたいことをプログラミングできるようになることです。

最後に、C言語で便利なプリプロセッサとC言語の基本ライブラリであるに入出力について簡単に説明します。

Cのプリプロセッサ

C言語では、プログラムのテキスト上に他のファイルのテキストをいれたり、ある語を他の語に置き換えたりするプリプロセッサというものがあります。プログラムで、＃から始まる文がプリプロセッサに対する指令で、以下のものがあります。

- `#include “ファイル名”`　ファイル名で指定されたファイルのテキストで、この＃がある部分に置き換えます。ファイル名が< …> でかかれた場合には、システムに定義されたファイルを指定するもので、unix では、/usr/include にあるファイルで置き換えられます。
- `#define xxx yyy`　テキストに現れる xxx を yyy で置き換えます。
- `#define f(x,y,…) …x …y`　f(a,b)の形式で現れる文字列を、… a … b　で置き換えます。

C言語では、ファイルは１つではなく、いろいろな部分を別々のファイルに書いて、それをあとであわせて、１つのプログラムにすることができます（分割コンパイルという）。その場合には、共有する構造体や大域変数を１つのファイルにして、xxx.h という.h（ヘッダファイルという）から始まるファイルにしておくことがよくあります。このような時には、#include “xxx.h”として読み込むわけです。例

えば、Cの printf を使うときに書いておくおまじない#include <stdio.h>は、システムの関数が定義されている/usr/include/stdio.h を取り込むという意味です。

#define は、文字列の置き換えです。#define f(x,y) …. は、見た目は関数定義同じようにみえますが、これは単なる文字列の置き換えです。たとえば、

```
#define z a
#define foo(x,y) x + y
goo(){
   int z,b,c;
   c = foo(z,b);
}
```

は、

```
goo(){
   int a,b,c;
   c = a + b;
}
```

と同じです。

Cの入出力関数

C言語ではファイルや端末からの入出力は言語の一部ではなく、関数ライブラリとして提供されています。scanf や printf もその一部です。以下のものがあります。

- **fopen(ファイル名、モード指定)** 文字列として与えられたファイル名のファイルをオープンします。モードとして、読み込みの時には文字列"r"を指定し、書き込みの時には"w"を指定します。この関数は、**ファイル構造体 FILE** へのポインタを返します。
- **fclose(FILE *ファイル構造体)** ファイル構造体に対応するファイルを閉じます。
- **fgetc(FILE*ファイル構造体)** ファイル構造体に対応するファイルから１文字読み込みます。ファイルが終わりの時にはファイルの終わりを示す-1(EOF)が返されます。
- **fputc(FILE *ファイル構造体、文字)** ファイル構造体に対応するファイルに１文字書き込みます。
- **fprintf(FILE *ファイル構造体、…)** ファイル構造体に printf をつかって書き込みます。のこりの引数の並びは printf と同じです。
- **fscanf(FILE *ファイル構造体、…)** ファイル構造体から scanf をつかって読み込みます。のこりの引数の並びは、scanf と同じです。

これらの関数を使うためには、ファイル構造体が定義してある stdio.h を include しなくてはなりません。例えば、a.txt というファイルに hello と書きたい場合には、以下のようにします。

```
#include <stdio.h>
main(){
  FILE *fp;
  fp = fopen("a.txt","w");
  fprintf(fp,"hello");
  fclose(fp);
}
```

次に、a.txt から、b.txt にコピーする例を示します。

```
#include <stdio.h>
main(){
   int c;
   FILE *fp,*fq;
   fp = fopen("a.txt","r"); fq = fopen("b.txt","w");
   while((c = fgetc(fp)) != -1) fputc(fq,c);
   fclose(fp); fclose(fq);
}
```

fgetc、fputc でつかう文字は int で宣言されていることに注意。