計算機応用 **5E**

5E　2003.6.12

本日の授業のテーマ

これまでの学習内容をまとめます。

(1) 重要な UNIX コマンド
(2) プログラムのコンパイルと実行方法
(3) C 言語
    - C 言語の基礎
    - データの型
    - 制御文
    - 配列
    - 関数
    - ファイル処理
(4) プログラム例
    - Hello World !!
    - 繰り返し (while)
    - 繰り返し (for 文)
    - 制御文　(if 文)
    - 関数(値渡し)
    - 関数(アドレス渡し)
    - ファイル処理(データ書き出し)

1 重要な UNIX コマンド

- UNIX のファイル構造は、ツリー(木)構造です。
  - 今、自分が居るディレクトリーを、カレントディレクトリーと言います。カレントディレクトリーのパスを調べるコマンドは、「pwd」です。
  - カレントディレクトリーの 1 つ上のディレクトリーを親デレクトリーと言います。親ディレクトリーへ移動するコマンドは、「cd ..」です。
  - カレントディレクトリーの直ぐ下のディレクトリーを子ディレクトリー、あるいはサブディレクトリーと言います。例えば、子ディレクトリーhogehoge に移動するコマンドは、「cd hogehoge」です。
  - ユーザー各個人が使用(読み、書き、実行)を許されている最上位のディレクトリーをホームディレクトリーと言います。移動するコマンドは、「cd」です。
- カレントディレクトリーにあるファイルやサブディレクトリーの名前を調べるコマンドは、「ls」です。
- サブディレクトリ hogehoge の追加と削除のコマンドは、以下の通り。
  - 追加は、「mkdir hogehoge」です。
  - 削除は、「rmdir hogehoge」です。
- ファイルを削除するコマンドは、「rm hogehoge」です。これで、hogehoge と言うファイルが削除されます。
- ファイルやディレクトリーをコピーするコマンドは、「cp hogehoge hugahuga」です。これで、hogehoge というファイルあるいはディレクトリーの hugahuga という名のコピーが作成されます。
- ファイルやディレクトリーを移動するコマンドは、「mv hogehoge ../hugahuga」です。親ディレクトリー(..)にサブディレクトリーhugahuga が無い場合、hogehoge が親ディレクトリーに移動して、名前を hugahuga に変更されます。もし、親ディレクトリー(..)にサブディレクトリーhugahuga がある場合、hugahuga のサブディレクトリーとして、hogehoge と言う名前で移動します。
- 以前使用したコマンドを呼び出す機能をヒストリー機能と言います。「↑」や「↓」で使用できます。同じような長いコマンドを何回も打ち込む手間が省けます。

2 コンパイルと実行

- C 言語のプログラムが書かれたソースファイルには、「hogehoge.c」のように拡張子「.c」が必要です。
- C 言語のソースファイル「hogehoge.c」をコンパイルして、実行ファイル「hugahuga」を作成するコマンドは、次の通りです。
  - 数学関数が無い場合　　cc -o hugahuga hogehoge.c
  - 数学関数が有る場合　　cc -lm -o hugahuga hogehoge.c
- ターミナルに「実行ファイル名(例えば、hugahuga)」を打ち込んで[Enter]キーを押せば、プログラムは実行されます。

3 C 言語

今後、C 言語を用いての数値計算を学習する上で、C 言語の重要な事項をまとめます。

3.1 C 言語の基礎

・C 言語では、大文字と小文字は、区別されます。変数名 hogehoge と Hogehoge、hoGehoge は異なります。

・コメント文は、プログラムの内容をわかりやすくするために記述するものです。これは、人間のためのもので、コンパイラーは無視します。/*～*/で囲まれた部分が、コメント文です。行をまたいでも、それは有効です。

・識別子とは、変数、記号定数、関数などにつける名前のことです。名前に用いることができる文字は決まっています。英大文字「A～Z」と英子文字「a～z」、数字の「0～9」とアンダースコアー「_」です。

・C はフリーフォーマットで記述できますので、文の区切りの記号が必要です。その区切りの記号にセミコロン「;」を用います。

3.2 データの型

・変数は定義してから用いなくてはなりません。型を指定することにより、変数を定義できます。これは、コンパイラーがプログラムを実行するときに、必要な領域を確保するためです。

・使用頻度が高い型は、以下の通りです。

型名	型指定子	変数宣言例
文字型	char	char a, b;
整数型	int	int i, j;
倍精度実数型	double	double x, y;

・C 言語では、変数の適用範囲は厳密に決められています。自動変数と外部変数があり、適用範囲が異なります。

自動変数　関数の中で定義され、その関数の中だけで使用できる。関数がコールされるとメモリー上に変数が配置される。その関数の処理が終わるとその変数は消滅する。通常良く使うのはこれである。

外部変数　関数の外で定義され、どの関数でも使用できる。プログラムが起動されるとメモリー上に変数が配置される。プログラムが終了するまで、変数は維持される。

3.3 制御文

・通常プログラムは、上から下へと実行されます。しかし、実行の流れを変えたい場合が多々あります。そのプログラムの実行の順序を制御するのを制御文といいます。

・使用頻度の高い制御文は、次の 3 個です。

```
if(条件 1){文 1}else if(条件 2){文 2}else{文 3}
```

条件 1 が真の時、文 1 が実行されます。条件 1 が偽の場合、次の条件 2 の真偽を判断し、真ならば文 2 を実行します。else if 文はいくらでも書くことができます。全ての if 又は else if の条件が偽ならば、else の文 3 を実行します。

for(初期値;継続条件式;再設定式){文}

実行順序は、以下の通り。

① 初期値の設定
② 継続条件が正ならば③へ、偽ならば for 文は終了
③ 文を実行
④ 再設定式を実行
⑤ 再び②を実行

do(条件式)while{文};

実行順序は、以下の通り。

① 条件式が正ならば②へ、偽ならば do 文は終了
② 文を実行。①へ戻る。

3.4 配列

・配列は、同じようなデータが多くある場合に使います。多くのデータに一つずつ名前をつけると大変です。1 万個のデータがあった場合、1 万個の名前を付けた変数を用意しますか?。下の例で、変数を用いての大量のデータ処理が不可能ということが分かるでしょう。

・1 万個の変数で領域を用意する場合の宣言

```
double aaa, aab, aac, aad, aae, aaf ・・・・;
                    ・
                    ・
                    ・
double oun, ouo, oup, ouq;
```

・配列で 1 万個の領域を用意する場合の宣言

```
double a[10000]
```

・配列を使う場合も宣言が必要です。宣言の例は、以下の通りです。

配列の次元	要素数	宣言例
1 次元	100	double x[100]
2 次元	100×100	double x[100][100]
3 次元	100×100×100	double x[100][100][100]

・配列添字は 0 から始まります。したがって、double x[1000]と宣言した場合、使える配列は、x[0]～x[999]までです。

・添字である数字でデータの指定ができるため、メモリからのデータの読み書きが単純化できます。

3.4 関数

・C 言語は、関数の集まりです。その中で main 関数は特別で、そこから実行されます。

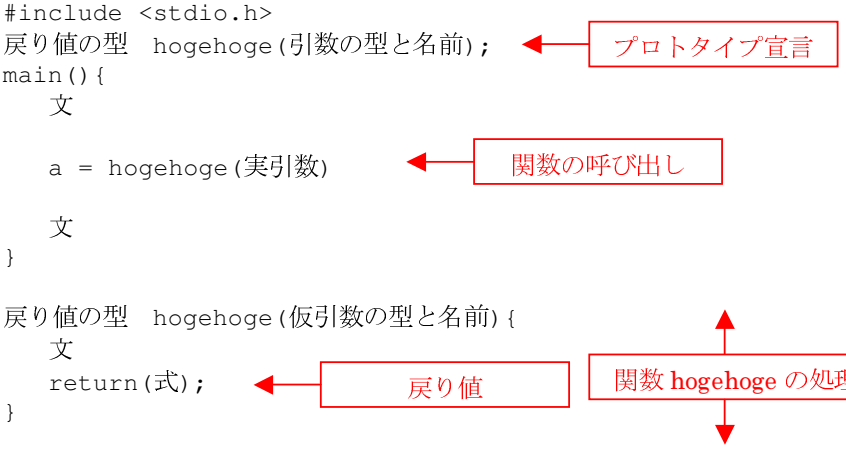
・プログラマーが関数を作成する場合、以下のように記述します。

```
#include <stdio.h>
戻り値の型  hogehoge(引数の型と名前);
main(){
    文

    a = hogehoge(実引数)

    文
}

戻り値の型  hogehoge(仮引数の型と名前){
    文
    return(式);
}
```



・関数へのデータの渡し方に、2 種類あります。

値渡し	呼び出す側と呼ばれる側の関数が各々変数を用意します。仮引数は、実引数のコピーとなります。呼ばれた関数が処理をしても、呼び出した側の実引数の変数は、影響がありません。
アドレス渡し	呼び出す側の実引数は、アドレスです。呼ばれる側は、ポインターを用意して、実引数のアドレスを受け取ります。呼ばれた関数が処理をすると、呼び出した側の実引数の変数にも影響があります。

3.5 ファイル処理

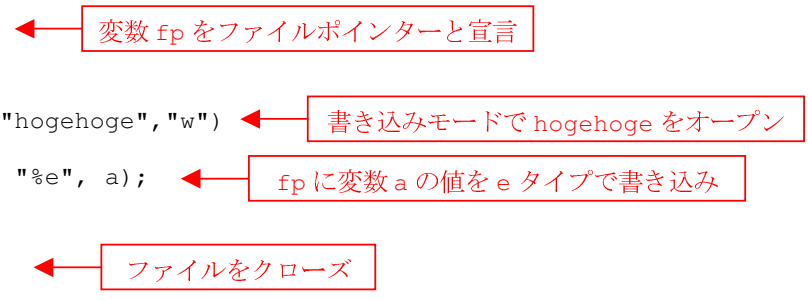
・ファイルへのデータの書き出しの手順は、以下のとおりです。

```
#include <stdio.h>
main(){
    FILE *fp
    文

    fp = fopen("hogehoge","w")

    fprintf(fp, "%e", a);
    文

    fclose(fp)
}
```



4 プログラム例

4.1 Hello World

・おなじみの超基本文

```
#include <stdio.h>
main()
{
   printf("Hello World!!");
}
```

4.2 繰り返し (while)

・1～100 の和を求めます。

```
#include <stdio.h>
main()
{
   int a, b;

   a = 1;
   b = 0;

   while(a<=100){
        b += a;
        a++;
   }

   printf("b = %d\n",b);
}
```

4.3 繰り返し (for 文)

・1～100 の和を求めます。

```
#include <stdio.h>
main()
{
   int a, b;

   a = b = 0;

   for(a=1; a<=100; a++){
        b += a;
   }

   printf("b = %d\n",b);
}
```

4.4 制御 (if 文)

・1～100 の和を求めます。

```
#include <stdio.h>
main()
{
   int a, b;

   a = b = 0;

   next: b+=a;

   if(a<100){
        a++;
        goto next;
   }

   printf("b = %d\n",b);
}
```

4.5 関数(値渡し)

・5+6 の計算を関数 sum で計算しています。

```
#include <stdio.h>
int sum(int x, int y);
main()
{
   int a,b,wa;
   a=5;
   b=6;

   wa=sum(a,b);

   printf("%d+%d=%d\n",a,b,wa);
}

int sum(int x, int y)
{
   return(x+y);
}
```

4.6 関数**(**アドレス渡し**)**

・関数 swap によって、実引数の値を入れ替えています。アドレス渡しであるため、関数側での操作が実引数に反映されます。

```
#include <stdio.h>
void swap(int *a, int *b);
main()
{
   int a, b;

   a=1;
   b=-1;

   swap(&a,&b);
   printf(" %d  %d\n",a,b);
}

void swap(int *a, int *b)
{
   int c;

   c=*a;

   *a=*b;
   *b=c;
}
```

4.7 ファイル処理**(**データ書き出し**)**

・ファイルに変数の値と三角関数の値を書き出しています。

```
#include <stdio.h>
#include <math.h>
main()
{
   FILE *out;
   double pi=4*atan(1);
   double theta, s, c, t;
   int i;

   out = fopen("calresult","w");

   for(i=0;i<=100;i++){
        theta = i*pi/100;
        s = sin(theta);
        c = cos(theta);
        t = tan(theta);

        fprintf(out,"%f\t%f\t%f\t%f\n",theta, s, c, t);
   }

   fclose(out);
}
```