関数についての復習と特訓

１．printf

おなじみの関数である printf について復習しましょう。

（１）引数（入力）

ご存じのように printf はフォーマット（"%d"や"%c"等）と、それに対応する式（x とか x＊x 等）を入力とします。

```
main(){
int x;
x=0;
printf("%d",x);
}
```

実行画面

```
info:[/home/jstaff/wada] :./a.out
0info:[/home/jstaff/wada] :
```

（２）戻り値（出力）

上のプログラムにおける、関数 printf の戻り値（出力）は何でしょう？ 0 だと思った人は、関数についての理解が不足しています。C 言語において（数学でも同じですが）関数 f の戻り値は、基本的には次のようにして確認できます。

（イ）x=f(引数)

（ロ） x の値を表示する

さて、printf の戻り値が整数型であることを考えながら以下の問題を解いて下さい。

練習問題５． １

以下の命令を実行したときの、printf の戻り値を調べよ。

（１）printf(" ")

（２）printf("%d %d ",0,0)

（３）printf("%d %d",0,0)

これらの結果から printf は、戻り値としてどのような値を返すか類推せよ。

上の問題から、最初のプログラムにおける printf の戻り値は 1 であることが分かると思います。実際には printf の戻り値を利用することは余り多くありませんが、関数概念の理解のために、例題として採りあげました。

（３）副作用

関数を理解するには引数、戻り値の他に”副作用”という概念が必要になることは以前に学びました（副作用を考えるところがＣ言語と数学との違いです)。副作用とは、戻り値を返すという動作以外の動作のことです。printf で言えば、引数の値をフォーマットに乗っ取って画面に表示することです。

printf の仕様をまとめると以下のようになります。

printf
引数：フォーマットと式
戻り値：出力文字数（整数型）
副作用：フォーマットに乗っ取って画面に文字を出力する

練習問題５．２

次の関数の仕様を書きなさい。

（１）練習問題３．３の triangles

（２）Ｃ言語演習第１週の power

練習問題５．３

以下の関数を利用する main 文を考えよ。

```
void wa(double x, double y)
{
        printf("%lf", x + y);
}
```

２．関数の合成

関数の合成という概念は数学のそれと全く同じです。二つの関数 f, g に対して、g の処理結果を f の引数として処理したいとき、合成を使います。書式は

f(g(引数)) ;

とします。動作としては

```
x=g(引数) ;
f(x);
```

と同じです。合成を使うメリットは

（１）見た目が見やすくなって頭が整理しやすいこと。
（２）変数の個数が少なくて済むこと

等があります。合成をする上で、当然留意すべきは、g の戻り値が f の引数のデータ型と一致しているか否かです。g の戻り値が実数型で、f の引数は整数型の場合、思った通りの結果が出ない場合があります。また、f は引数を２個必要とするのに g の戻り値は一つという場合は単純に合成することはできません。f の引数が２つの場合には、

```
f(引数 1,g(引数 2)) ;
```

とすることもあります。

練習問題５．４
論理関数ＮＯＴとＡＮＤを作成し、さらに main においてそれらを合成して NAND を作れ。

３．関数を用いたプログラム
（１）実引数と仮引数
関数を呼び出すときにやりとりするデータのことを引数と呼ぶことは以前に学びました。以下のプログラムを例にとって説明しましょう。

```
double wa(double x, double y)
{
        double          z;
        z = x + y;
        return z;
}
```

```
main()
{
        double          a, b, sum;
        a = 11.1;
        b = 22.2;
        sum = wa(a, b);
        printf("%lf", sum);
}
```

プログラム１．二つの実数の和を計算する関数

上のプログラムの main のなかで、wa という関数を呼び出しています。引数は a,b そして x,y です。さて、wa(a,b)という記述は、変数 a,b に入っている値を考えれば、wa(11.1,22.2)と同じです。一方、関数定義の部分に現れる x と y という引数について考えてみると、これは、wa という関数の処理方法を記述するための”仮の”引数です。

今後は

a,b のように実際に値を持っている引数（実引数）と、

x,y のように実際に値を持っていない引数（仮引数）

があることに注意して下さい。実引数と仮引数が見た目に同じでもプログラムは問題なく動作しますが、当分は両者を違う記号で書く事を心がけて下さい。

（２）引数の処理方法（データ渡しのしくみ）

関数を使ったプログラムを書く為には、関数にデータを引数として渡したときにデータがどの様に処理されるかを知っておく必要があります。

（２．１）数値を渡す場合

上のプログラム１を見て下さい。main の中で関数 wa を呼び出した際、以下のような処理が行われます。

main 部　　|　　関数部

---

a の値をコピー

実引数（変数 a=11.1）----------> 仮引数（変数 x）

b の値をコピー

実引数（変数 b=22.2）----------> 仮引数（変数 y）

和を計算（z=x+y）

z の値をコピー

変数 sum　　<----------　　変数 z

練習問題５．５

（１）以下のプログラムを実行するとどうなるか考えよ。

```
double wa(double x, double y)
{
        double          z;
        z = x + y;
        x=0;
        y=0;
        return z;
}
main()
{
        double          a, b, sum;
        a = 11.1;
        b = 22.2;
        sum = wa(a, b);
        printf("%lf¥n", sum);
        printf("%lf %lf¥n ", a,b);
}
```

上のプログラム、もしくはその前の説明から重要な教訓が得られます。

関数内で行った（仮）引数への代入操作は、関数を終了すると全て無効になる。

ということです。

（２．２）配列名を渡す場合
関数に配列名を渡すプログラムを考えます。

```
int add(int a[2])
{
        return a[0]+a[1];
}
main()
{
        int             b[2];
        b[0] = b[1] = 1;
        printf("%d¥n",add(b));
}
```

実行結果は容易に予想できるでしょう。

実行結果
2

次のプログラムの場合はいかがでしょう。

```
void input_zero(int a[2])
{
        a[0] = 0;
        a[1] = 0;
}
```

```
main()
{
        int             b[2];
        b[0] = b[1] = 1;
        input_zero(b);
        printf("%d¥n",b[0]);
        printf("%d¥n",b[1]);
}
```

実行結果

0

0

この実行結果を見て不思議に思いませんか。先ほど得た教訓によると、

関数内で行った引数への代入操作は、関数を終了すると全て無効になる

はずではありませんか。そこでもう一度上のプログラムを検討しましょう。

Ｃ言語プログラムをコンパイルした後の実行ファイルでは、配列の名前は数値で表されています。この事実に注意しながら説明します（どのような仕組みで数値が割り当てられるかは３年生以降で学びます)。ここでは配列 b の配列名が 2147 であったとしましょう。

main | input_zero

代入（コピー）

実引数（ b の配列名）----------> 仮引数（ 配列名 a）

2147

a[0] = 0

（配列名 2147 の 0 番目に 0 を代入せよ）

a[1] = 0

（配列名 2147 の 1 番目に 0 を代入せよ）

関数 input_zero 内で 「配列 2147 の中身を変更せよ」という命令を出しているので、この段階で配列 b の中身は変更されてしまうわけです。

上のことから、

配列に入れてある数値や文字は、配列名を受け渡す関数を使えば書き換えられる

と言うことが分かります。

練習問題５． 6
次のプログラムの実行結果を予想せよ。その理由も述べよ。

```
void swap(int x, int y)
{
        int             tmp;
        tmp = x ;
        x = y ;
        y = tmp ;
}
main()
{
        int             a, b;
        a = 11;
        b = 22;
        swap(a, b);
        printf("%d %d¥n ", a,b);
}
```

練習問題５． 7
二つの整数型変数の値を入れ替える関数はどのように作ったら良いだろうか。配列名を引数にする（受け渡す）関数によって実現せよ。

●補足　２次元配列を関数に渡す方法について

配列を引数とする関数について復習しましょう。

一次元配列を引数とする関数は例えば以下のように作成します。

```
int funct(int a[3]){
.....

}
```

引数になる配列のサイズが、未定の時は、以下のように
サイズを省略することができます。

```
int funct(int a[]){
.....

}
```

さて、２次元配列を関数に受け渡す場合は以下のようにします。

（例）４x３の整数型配列を引数とし、整数値を返す関数 funct の場合、

```
int funct(int a[4][3]){

.....

}
```

この関数を使うときには

```
funct(a)
```

の様に記述します。一次元の時のように引数のサイズを省略したい場合もあるかもしれませんが、それはできません。ただ、以下のように第一成分のみなら、省略可です。

```
int funct(int a[][3]){

.....

}
```

以上