第２・３回Ｃ言語講座

2011/5/16＆5/23

メインテーマ：条件分岐とループ、関数

①変数の型（＋前回の復習）

整数を扱う→int 型変数
小数を扱う→float 型変数
文字（一文字）を扱う→char 型変数
※C での文字列は数回後に扱います。

入力→scanf
出力→printf
printf/scanf での変数の入出力：
%d → 整数　　例：printf("%d", 3);
%f → 小数　　例：printf("%f", 3.14);
%c → 文字　　例：printf("%c", 'a');

```c
#include <stdio.h>
//基本的にmain(){…}の中にプログラムを書いていく
int main(void)
{
	//変数はmainの先頭で宣言
	char c ;　//文字型（文字のみ扱える）
	printf("何か文字を入力してください¥n");
	scanf( "%c" , &c );　　//入力
	printf( "入力文字→%c¥n" , c );　//出力
	return 0;
}
```

○キャスト

ある型を別の型に強制的に変換する。
(変換したい型名)変換するもの
で強制的に型を変換できる。

右の場合、
f_syo1の結果が3.0000
f_syo2の結果が3.3333
となってちゃんと計算がfloat型に変換されていることがわかる。

結果↓


```c
#include <stdio.h>
int main(void)
{
	int x = 10 , y = 3 , i_syo;
	float f_syo1 , f_syo2;

	i_syo = x / y ;
	f_syo1 = x / y ;
	f_syo2 = (float)x / y ;　//キャスト(intをfloatに)

	printf( "i_syo:x/y=%d¥n",i_syo);
	printf( "f_syo1:x/y=%f¥n",f_syo1);
	printf( "f_syo2:x/y=%f¥n",f_syo2);
	return 0;
}
```

②条件分岐（if 文、switch 文）

条件によって行う処理を変更できる（分岐を行える）制御文。

○if 文

```
if( 条件 )
{
    //条件が合っていた時の処理
}
else //省略可能
{
    //条件に当てはまらなかった時の処理
}
```

条件には主に以下のものが使える。

数学での記号	C 言語での記号
＞　＜	＞　＜
≧　≦	>=　<=
＝　≠	==　!=
( ) { } [ ]	( ) ←この括弧のみ
AND(かつ)	&&
OR(または)	||

```
#include <stdio.h>
int main(void)
{
        int num ;
        printf( "値→" );
        scanf( "%d" , &num );
        //num≧10とnum＜10で処理を分ける
        if( num >= 10 )  //num≧10のときの処理
        {
                printf( "10以上の値です¥n" );
        }
        else    //num≧10でない(num＜10)ときの処理
        {
                printf( "10未満の値です¥n" );
        }
        return 0;
}
```

○switch 文

```
switch( 変数 )
{
    case 定数:
        //処理
        break;
    default:
        //処理
        break;
}
```

switch に書かれた変数の値によって処理を分岐する。

case の後ろに書かれた定数と変数の値が一致すればその case から break が書いてある部分までの処理を実行する。

case のどの値とも一致しなかった場合は default に書かれている処理を実行する。

default は不要なら省略可能で、その場合条件に一致しなかったら何もしない。

```
#include <stdio.h>
int main(void)
{
        int selection ;  //選ばれた計算方法
        int x , y ;      //計算用の値
        int result = 0;  //計算結果
        printf( "計算方法を選んでください¥n" );
        printf( "0:加算 1:減算 2:乗算¥n" );
        scanf( "%d" , &selection );
        printf("2つの値を入力¥n" );
        scanf( "%d %d" , &x , &y );
        //変数selectionの値によって処理を分ける
        switch ( selection )
        {
        case 0:          //加算( selection == 0 )
                result = x + y ;
                break;
        case 1:          //減算( selection == 1 )
                result = x - y ;
                break;
        case 2:          //乗算( selection == 2 )
                result = x * y ;
                break;
        default:         //選択肢にないものが選ばれたとき
                printf( "不正な選択肢です¥n" );
                break;
        }
        printf( "計算結果= %d¥n" , result );
        return 0;
}
```

③ループ

同じ処理を何回も繰り返す制御文

○for 文

```
for(初期化;継続条件;処理)
{
    //ループする処理
}
```

初期化の処理は for 文が始まる際に一回だけ実行される。
最初と 1 ループごとに継続条件を調べ、合っていなかったらループを抜ける。(右の場合 i >= 10 になったらループを抜ける)

※ i++ はインクリメントという。
i = i+1 と基本的には同じ意味。
(逆に i--はデクリメントという)

```c
#include <stdio.h>
int main(void)
{
        int i; //ループ用変数
           //初期化 継続条件  1ループ毎の処理
        for( i = 0 ; i < 10 ; i++ )
        {
                printf( "%d¥n" , i );
        }
        return 0;
}
```

○while 文

```
while( 継続条件 )
{
    //処理
}
```

for 文と違いループの条件判断のみを行う。
for 文がカウントの要る処理（特定回数をカウントさせる）のに適しているのに対し、while 文はカウントの要らない処理を行うのに適している。

```c
#include <stdio.h>
int main(void)
{
        int i = 0;
        while( i < 10 )  //iの値が未満ならループ
        {
                printf( "10以上の値が入力されたら終了します¥n" );
                scanf( "%d" , &i );
        }
        return 0;
}
```

○do～while 文

基本的には while 文と同じ。
ただし、条件の判定は毎ループが終了した後なので、必ず一回は do～while の処理が行われる。

この場合、while(条件); と while の括弧の後ろにセミコロンが必要なのに注意。

```c
#include <stdio.h>
int main(void)
{
        int i = 20; //条件に合っていない
        do
        {
                printf( "10以上の値が入力されたら終了します¥n" );
                scanf( "%d" , &i );
        } while( i < 10 );
        return 0;
}
```

④関数

C 言語のプログラムは「関数」と呼ばれる小さなプログラムの集合体で書かれます。
今まで扱ってきた printf()も scanf()もコードを書いていた main()も実は関数。
関数はいくつかの命令文の集まりで、例えば printf 関数なら画面に文字を表示するのに必要な命令を書き並べてできています。

関数は以下の形式をとります。

『戻り値　関数名（引数)』(例： int main(void)：戻り値 int 型、関数名 main、引数なし　)

戻り値：関数の処理が終わった際に関数内の return 文によって返されるもの。
関数名：関数の呼び出しに使う名前。main , printf , scanf など
引数（ひきすう）：関数に渡す情報。例：printf("Hello World!");の表示する文字列"Hello World!"のこと

※引数に何も取らない(特定の処理を行うため、外部からの情報が要らない場合)や戻り値に何も返さない場合、その部分を void と書きます。main 関数はそこからプログラムが始まるので引数が必要ないため、main(void)となっています。
(引数をとる main 関数もありますが、この講座では扱わないので気になったら調べてみて下さい)

〇関数のインクルード
C 言語の規格よってあらかじめ用意されている関数を標準ライブラリ関数といいます。
これらの関数を使うには対応する#include から始まる行を書く必要があります。
(対応する「～.h」ファイルをインクルードする(含める)。「～.h」には関数を使うための情報が書いてある。)

例えばすでに暗黙のうちに使っていますが、printf 関数や scanf 関数を使うには#include <stdio.h>と書く必要があります。
標準ライブラリ関数には数学関数や文字列を操作する関数などがあります。
例を以下に示しておくので気になったら自分で調べてみたり講師に聞いたりしてもらえれば。

#include <stdio.h>
主にファイル関係を扱う関数群。printf , scanf などのコンソール画面への入出力や fopen , fclose ,fprintf , fscanf などのファイルを扱う関数（後半で触れます）など

#include <math.h>
数学関係の関数群。sin , cos , tan などの三角関数や累乗の pow , 平方根を求める sqrt , 対数 log など。

#include <string.h>
文字列を扱う関数群。２～３回後に触れます。

#include <time.h>
時間関係の関数群。現在時刻を取得する time 関数など。

〇自作関数

当然ながら自分で関数を作ることもできます。

右のプログラムでは戻り値が float 型（面積を返すため）で関数名が circle、引数に半径 r をとる関数を自作しています。

プログラムは基本的に上から実行されるため、main の後に関数を書く場合は右のようにプロトタイプ宣言を書いて、「こんな関数があるよ」と教えてあげる必要があります。

自作関数本体が main 関数より上にある場合はプロトタイプ宣言は必要ありません。

※main 関数自体はどこに書いてもそこから始まるため、場所は問いません。

```c
#include <stdio.h>
//円の面積を求める自作関数プロトタイプ宣言
float circle(int r);

int main(void)
{
	int num;
	float result = 0;
	printf("半径の値を入力してください¥n");
	scanf( "%d" , &num );
	result = circle( num );  //自作関数の呼び出し＆戻り値の取得
	printf( "半径%dの円の面積は%fです。¥n", num , result );
	return 0;
}

//自作関数の本体
float circle(int r)
{
	return r*r*3.141592f;
}
```

自作関数本体が上にある場合。

```c
#include <stdio.h>
//２乗を返す
int square(int x)
{
	return x*x;
}
int main(void)
{
	int num , result;
	printf("値を入力してください¥n");
	scanf( "%d" , &num );
	result = square( num );  //自作関数の呼び出し&戻り値の取得
	printf( "%dの2乗は%dです。¥n", num , result );
	return 0;
}
```

演習問題（基本的に問題に書かれた動作をすれば正解とします）

①

キーボードから入力された値について以下の処理を行うプログラムを作れ（入力される値は正の整数とする）

a) 1～入力された値までのすべての3の倍数を表示するプログラムを作れ

b) 1～入力された値までの整数の合計を表示せよ

a)ヒント：3の倍数＝3で割った余りが0

②

#include <stdlib.h>を書くことによって rand 関数(戻り値 int 型、引数なし、ランダムな値を返す関数）を使うことができる。

この関数とループを使って乱数を 10 個表示するプログラムを作れ。

※疑似乱数について

この問題の実行結果は毎回同じになるが、それはコンピューターが「種」と呼ばれる値を用いて計算によって乱数を算出しているため。(「疑似乱数」という)

rand 関数はデフォルトの種の値が0になっているのでこの値を変更しないと毎回同じ値が出てくることになる。

乱数の種を変更するには srand 関数（引数に設定したい値を入れる）を使う。

③

以下のような簡単なハイアンドローゲームを作れ

ランダムで 0 から 10 の数字を発生させ、その数字が 5 より大きいか小さいかを当てる。(5 をどちらに含むかは製作者に任せる。)

数字が 5 より大きかったら 1、小さかったら 0 を入力してもらい、それ以外の値が入力されたらプログラムを終了させる。

余裕があればループを用い、0 か 1 以外の値が入力されるまで何回でも遊べるようにしてみること。

ヒント： 0～10 の乱数は rand()%11 と書いてよい。

④(+α)

組み合わせ（コンビネーション）の計算をするプログラムを作れ。ただし、組み合わせの計算には自作関数を作って計算すること。

※組み合わせの計算方法

$$nCr = \frac{n!}{(n-r)!\,r!} = \frac{[n \times (n-1) \times (n-2) \times \cdots \times \{n-(r-1)\}]}{r \times (r-1) \times \cdots \times 2 \times 1}$$

ものすごく余裕があれば再帰関数を使って実装すること。

※再帰関数とは

関数の中で自分自身と同じ関数を呼び出すこと。詳しくは口頭で。