第２回Ｃ言語講座

2012/5/30

メインテーマ：条件分岐とループ、関数

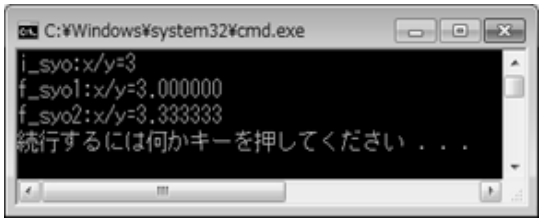
〇キャスト

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、
d_syo1の結果が3.0000
d_syo2の結果が3.3333
となってちゃんと計算がdouble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;
	double d_syo1 , d_syo2;

	i_syo = x / y ;
	d_syo1 = x / y ;
	d_syo2 = (double)x / y ; //キャスト(intをdoubleに)

	printf( "i_syo:x/y=%d¥n",i_syo);
	printf( "d_syo1:x/y=%lf¥n",d_syo1);
	printf( "d_syo2:x/y=%lf¥n",d_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。

〇if 文

```c
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)	||

```c
#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。

```c
#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("1つめの値を入力¥n" );
        scanf( "%d" , &x );
        printf("2つめの値を入力¥n" );
        scanf( "%d" , &y );
        //ここまでコピー
        //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( "これは%d回目のprintfです. ¥n" , i );
        }
        return 0;
}
```

〇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。

```
#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;
}
```

〇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。

```
#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;
}
```

第二回演習問題（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