ゲーム制作のための

C言語講座～第２回～

2007/08/15

昨日はお疲れ様です。あれから、プログラミングを打ちましたか？まだの人はソース丸写しでもいいので、やってみましょう。最初は覚えられなくても、打つことで段々覚えられるようになります。できることなら、プログラミングを日課にするくらいやるといいですよ。（そういう私は・・・。）

今回は条件分岐（if 文、switch 文）と配列、繰り返し（for 文、while 文）です。前回と今回の２回をきちんとマスターできれば、簡単なゲーム・アプリケーションを作ることは可能です。（ただし、ソースが莫大な量に・・）きちんと吸収できるよう頑張ってください。では、講義を始めます。

２－１　if 文

条件分岐の if 文です。多分、ゲームおよびアプリケーションを作る上でもっとも重要な命令でしょう。ゲームで言えば、ストーリーの分岐に始まり、当たり判定、プレイヤーの操作、イベントの有無、フラグ・・・・。まあ、言いはじめるとキリがないかもしれません。とりあえず、重要なものなのでしっかり覚えてください。

if 文の書き方は if（ 条件式 ）です。条件式とは２つの値（式）との関係がその条件に合っているかいないかを判断する式です。

基本的な条件式

記号	使用例	意味
＝＝	A＝＝B	A と B の値が同じ
！＝	A！＝B	A と B の値が違う
＞＝or＜＝	A＞＝B（A＜＝B）	A の値が B の値以上（以下）
＞or＜	A＞B（A＜B）	A の値が B の値を超えている（未満）
＆＆	A＞B　＆＆　A＞C	A＞B　かつ　A>C（両方とも成り立つ）
｜｜	A＞B　｜｜　A＞C	A＞B　または　A＞C（どちらか片方でも成り立つ）

条件式が正しければ、中括弧内の内容を実行します。正しくなければ、内容を実行せずに下に進みます。では、１つソースを書いてみましょう。

source2-1a.cpp　if 文

```
#include<stdio.h>
int main()
{
	int data = 10;
	data += 15;				/*data = data + 15;*/
	if(data >= 20)		/*この時dataは２５*/
	{
		printf("dataは２０以上です¥n");
	}
```

```
        else
        {
                printf("dataは２０未満です¥n");
        }
        return 0;
}
```

４行目で int 型の変数「data」を宣言し、初期値として１０を代入しています。次の行で１５を加算して、「data」は２５という数値になりました。

そして、６行目：if(data >= 20)で、「data」という変数が２０以上か判定しています。もちろん、２５なので、この条件式は合っています。なので、中括弧内の printf(“data は２０以上です¥n”)；が実行されます。

それともう１つ、if 文によく使われるものがあります。それは else です。これは正しくない時のみ、実行されます。このソースの例だと「data>=20 ではない」⇒「data<20」のときに else 以下の中括弧の内容が実行されます。実際に、data を２０未満の数値に置き換えたソースを書いて実行してみてください。

そして、else に if をくっつけた形もあります。ソースを見ながら説明します。

source2-1b.cpp　else if

```
#include<stdio.h>
int main()
{
        int data = 10;
        data += 15;                        /*data = data + 15;*/
        if(data >= 30)            /*この時dataは２５*/
        {
                printf("dataは３０以上です¥n");
        }
        else if(data >= 20)
        {
                printf("dataは２０以上３０未満です¥n");
        }
        else
        {
                printf("dataは２０未満です¥n");
        }
        return 0;
}
```

１０行目：else if というのがそれです。この if 文が実行されるのは前の if 文に書いてある「data>=30」を満たしてないときです。つまり、３０未満の時ですね。その中で、「data>=20」という条件に合っていれば、中括弧の内容が実行されます。

一番下の else は上２つの if 文の条件に合っていないとき「data>=30」かつ「data>=20」⇒「data<20」

の時のみ実行されます。

それと注意して欲しいのが、数学でよく「２＜x＜５」とか使っていたと思います。しかし、C 言語でそのまま書くと、思い通りの動作がしないと思います。ですから、「＆＆」を使って、２つに分けて書いて下さい。（この場合だと「２＜x　＆＆　x＜５」）

２－２　switch 文

これも if 文と同じ条件分岐のための命令です。これは、ある値の大きさによって、場合分けをします。書き方は switch(式 or 変数)という形で書きます。とりあえず、ソースを見ましょう。

source2-2.cpp　switch 文

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main()
{
	int data;
	srand((unsigned)time(NULL));
	data = rand() % 3;
	switch(data)		/*dataの値によって条件分岐します*/
	{
		case 0:			/*０のとき*/
			printf("パー¥n");
			break;
		case 1:			/*１のとき*/
			printf("グー¥n");
			break;
		case 2:			/*２のとき*/
			printf("チョキ¥n");
			break;
		default:		/*それ以外のとき*/
			printf("謎のエラーが発生しました¥n");
			break;
	}
	return 0;
}
```

９行目：switch(data)というところで、data の値によって分岐されるようになっています。そのあとに１１、１４、１７行目に case というものがあります。これが、data の値がその数値だった時、それ以降の命令を実行するというものです。

そして、printf で出力した後に break というものがあります。これは処理を中断して抜ける命令です。switch 文の中にあるので、break まで来ると switch 文から抜けることになります。つまり、switch 文の中括弧のすぐ下のところに移動してそこから処理を再開することになります。

break を抜かしてしまうと、そのまま下の処理まで実行してしまいます。たとえば、0 だった時には、「パー」「グー」「チョキ」「謎のエラーが・・・」の４文が同時に出てきてしまいますので注意してく

ださい。

２－３　配列

配列とは何かという話に入る前に、すこしメモリについて説明しておきます。変数の値のようなデータはメモリという記憶装置の中に記録されます。そのデータを計算する CPU に持っていって、演算をした結果をメモリに記録するという動作を繰り返して、プログラミングが実行されています。そこで、メモリにたくさんのデータを格納できるように、メモリ上に番地（アドレス）を設けて、どのアドレスに何のデータが入っているかを OS が管理しています。ただ、変数宣言をした時に、その変数のデータのアドレスが決定しますが、複数宣言した時としても、それらのアドレスが隣同士とは限りません。

ですが、配列というもので宣言すると、複数の変数が連続的にメモリ上に確保できます。これを後々やる for 文やポインタ、構造体などと組み合わせることでかなり効率的なプログラミングが組めるようになります・・・・。

と、言ってもわかりにくいと思うので、簡単に説明すると、変数は値を入れる「箱」のようなものだと前に言っておきました。配列は複数の「箱」を１つにくっつけて、大きな１つの「箱」にすることで、効率よくプログラムを組もうと考えられたものだと思ってください。どういう使い方をするのかをソースに書いてみました。

source2-3.cpp　配列

```
#include<stdio.h>
int main()
{
        int data[3];
        data[0] = 5;
        data[1] = 10;
        data[2] = 15;
        printf("１つ目は%d¥n２つ目は%d¥n３つ目は%d¥n", data[0] , data[1] , data[2]);
        return 0;
}
```

変数宣言同様、配列の宣言も関数の最初に行います。４行目：int data[3]がそうです。３は配列の個数です。つまり、配列の宣言は「型 配列名[配列の個数]」となっています。

５～７行目に各配列に値を代入しています。ただ、変数宣言同様に配列も初期化することができます。書き方は int data[3] = {5 , 10 , 15} ;　という風に書きます。中括弧の中に配列の数と同じ数だけの値を入れてください。

ちなみに、第１回目に少し出てきた「文字列」ですが、これは文字を扱う変数「char 型」の配列です。文字列が１つの変数に入るわけではなく、１つ１つの文字が配列に入っていて、それが１つにくっつけているだけなのです。まあ、このことについては、最終日に詳しくやろうかと思います。

２－４　for 文

では、繰り返し文について、説明します。繰り返し文とは一定区間をある条件が成立するまで繰り返すものです。C 言語では for 文と while 文の２つが使われています。まず、for 文の使い方を説明します。下のソースを見てください。

source2-4.cpp　for 文

```c
#include<stdio.h>
int main()
{
	int data[9];
	int i;
	printf("九九の三の段のを出力します。¥n");
	for(i = 1 ; i <= 9 ; i++)
	{
		printf("%d¥n" , i * 3);
	}
	printf("iの値は%dです。¥n" , i);
	for(i = 0 ; i < 9 ; i++)
	{
		data[i] = 5 * (i + 1);
	}
	printf("九九の五の段を出力します。何番目の数値を出力しますか？¥n");
	scanf("%d" , &i);
	printf("%d番目の五の段は%dです。¥n" , i , data[i - 1]);
	return 0;
}
```

まず、７行目に for 文があります。括弧の中身は（初期値；条件式；変化式）という構成になっています。で、下の８～１０行目までの中括弧が繰り返される内容です。for 文の動き方は・・・、

① 初期値のところで、変数に値を代入します。
② 条件式を見て、それが正しいかどうか判断します。
③ 条件式が正しければ、中括弧の内容を実行します。
④ 変化式にある演算を行います。
② 条件式を見て、判断。
③ 中括弧の内容を実行。
:
:
② 条件式を見て、判断。⇒正しくない⇒繰り返し処理終了⇒中括弧の下に移動

というように、最初に①を行い、後は②が成立しなくなるまで、②～④を繰り返していきます。

９行目の printf で、i×３の値を出力しています。繰り返すたびに i は１ずつ増えていくので、３、６、９、１２、・・・、２７と表示されます。

１１行目の printf で、for 文を抜けた直後の i の値を出力しています。大学の講義で C 言語をやる人は注意してください。テストに１回は出るような問題です。上の法則に従っていけばわかると思います。（決して、９ではありませんよ！）

１２行目の for 文は配列を利用したものです。昨日の講義で変数名に「seisu1」「seisu2」とか使いました。ただこの１や２は「名前＝文字」です。数値ではありません。でも、配列の順番を表す[]の中身は「番号＝数値」です。なので、１４行目のように data[i]と変数を[]の中に書いても問題がありません。この for 文も i が１つずつ増えているので、繰り返されるごとに次の配列に値を代入していく形になっています。

２－５while 文

もう１つの繰り返し、while 文をやっていきます。これもソースを先に書いてみました。

source2-5a.cpp　while 文

```
#include<stdio.h>
int main()
{
	int data[9];
	int i = 0;
	while(i < 9)
	{
		data[i] = 5 * (i + 1);
		i++;
	}
	printf("九九の五の段を出力します。何番目の数値を出力しますか？¥n");
	scanf("%d" , &i);
	printf("%d番目の五の段は%dです。¥n" , i , data[i - 1]);
	return 0;
}
```

前のソースの２つ目の for 文とやっていることはまったく同じです。for 文を while 文に書き直しただけです。while 文の後の括弧には条件式だけを書きます。初期値や変化式は書きません。動作としては、

① 条件式が正しいか判断する。
② 正しいなら、中括弧内を実行する。
　　：
　　：
① 条件式の判断⇒正しくない⇒繰り返し処理の終了⇒中括弧の下に移動

となっています。初期値や変化式がない分簡単に見えますね。ですが、while 文の前で i を初期化したり、繰り返しの中に i を変化させる命令を書いていたりするので、実際作りやすさはあまり変わりません。まあ、初期化・変化式が必要なら for 文、必要ないなら while 文で書くのがやりやすいかと思います。

あと、do～while 文というものもあります。これは、中括弧の中身を１回実行してから、while の後にある条件式を見て、繰り返すかを判断します。使うことはあまりないかもしれませんが一応載せておきます。

source2-5b.cpp　do-while 文

```
#include<stdio.h>
int main()
{
	int data[9];
	int i = 0;
	do
	{
		data[i] = 5 * (i + 1);
		i++;
```

```
        }while(i < 9);
        printf("九九の五の段を出力します。何番目の数値を出力しますか？¥n");
        scanf("%d" , &i);
        printf("%d番目の五の段は%dです。¥n" , i , data[i - 1]);
        return 0;
}
```

最初に do が来て、その後すぐに中身を書きます。中括弧の終わりに while と書き、条件式を書きます。ここで注意するのが、「;」が必要だということです。忘れないようにしましょう。

２－６　応用１～繰り返しと break～

switch 文で使用した break を繰り返しの for 文、while 文に使用することができます。for 文、while 文の中で、break が実行されると、その時点で、繰り返し処理が中断され、中括弧の下へ移動します。

source2-6.cpp　繰り返しと break

```
#include<stdio.h>
int main()
{
        int ans;
        while(1)
        {
                printf("村人：勇者様、あのドラゴンを倒してください！¥n");
                printf("1 - 「はい」　2- 「いいえ」¥n");
                scanf("%d" , &ans);
                if(ans == 1)
                {
                        break;
                }
        }
        printf("村人：あ、ありがとうございます！！¥n");
        return 0;
}
```

この村人は「はい」を選択しない限り、永遠にドラゴンを倒してくださいと連呼します。（どっかの RPG で似たようなのがあったような・・・。）

解説をすると、まず、５行目の while の条件式に１が入っています。これは常に真（正しい）ことを差しています。つまり、中括弧の内容が無限ループするということです。

それで、１０行目の if 文が ans の値が１のときに break が実行されるようになっています。ちなみに、if 文と break に関わりがないため、この break は while の処理を中断するものになっています。for 文でも同じように使えるので、自分で打ってみて下さい。

２－７　応用２～入れ子～

今日の講座の最後の部分になりました。入れ子というのは箱の中に小さな箱が入っているやつです。まあ、プログラミングの場合だと if 文の中に if 文を入れたり、for 文の中に for 文を入れたりする方法

です。

source2-7a.cpp　if 文の入れ子

```
#include<stdio.h>
int main()
{
	int data = 20;
	if(data > 10)
	{
		if(data > 20)
		{
			printf("dataは２０を超えています¥n");
		}
		else
		{
			printf("dataは１１以上２０以下です¥n");
		}
	}
	else
	{
		printf("dataは１０以下です¥n");
	}
	return 0;
}
```

if 文の中に if 文が入っているタイプです。5 行目の if 文の条件が満たされた時のみ、7 行目の if 文に行きます。つまり、1 度条件を絞ってさらにもう 1 度条件を絞っているわけです。else の中ももちろん入れ子にすることは可能です。やってみてください。あと、入れ子を使わないで分岐することもできますね。（条件式に＆＆とかを使って、細かい条件にすれば）それもやってみてください。

source2-7b.cpp　for 文の入れ子

```
#include<stdio.h>
int main()
{
	int data[9][9];
	int i , j;
	for(i = 0 ; i < 9 ; i++)
	{
		for(j = 0 ; j < 9 ; j++)
		{
			data[i][j] = (i + 1) * (j + 1);
		}
	}
	printf("２つの値を掛け算します。１～９までの数値を入力してください¥n");
```

```
        scanf("%d" , &i);
        printf("もう１つの値を入力してください¥n");
        scanf("%d" , &j);
        printf("結果…%d¥n" , data[i - 1][j - 1]);
        return 0;
}
```

今度は、for 文の入れ子です。４行目：int data[9][9]；では配列の個数を書くところが２つあります。これは普通の配列（１次配列）自体をさらに配列にした２次配列というものです。まあ、１次配列を１つの箱と考えて、同じ箱同士をくっつけて１つのもっと大きな箱にしています。

そして、for 文の入れ子の説明です。６行目と８行目に for 文があります。この２つの動作は・・・、

i＝０のとき、j を０～８まで繰り返す。
i＝１のとき、j を０～８まで繰り返す。
　　：
　　：
i＝８のとき、j を０～８まで繰り返す。

と動いています。つまり６行目の for 文がループするたびに、８行目の for 文が９回ループしています。

よって、６行目のループが終わった時には、 data[i][j] = (i + 1) * (j + 1); という命令が、９×９＝８１回のループをしていることになります。

２－７　まとめ

内容	関数・書き方
条件分岐	if(条件式)｛真のときの実行｝else｛偽のときの実行｝ switch（値・式）｛case 値：～break；｝
繰り返し	for(初期値；条件式；変化式)｛内容｝ while(条件式)｛内容｝
配列	型 配列名[配列の個数]（例：int data[5]） 配列は０番目からスタートするので注意

２－８　演習

じゃんけんゲームを作る。（１度ゲームをしたら、またやるかどうかを聞いて、やるなら再度できるようにする。あと、関係ない値を入力したら、強制負けにするなど工夫してみよう。）