ゲーム制作のための

# C言語講座～第１回～

2007/08/14

この講座では・・・、

第１回　データ入出力・型・演算・乱数
第２回　配列・条件分岐（if 文、switch 文）・繰り返し（for 文、while 文）
第３回　自作関数・構造体
第４回　ポインタ関連
第５回　ファイル入出力・その他色々

の計５回でC言語の基礎的な部分をやっていきます。ただ、少ない日数で教えるため、内容は薄く範囲も最低限のところになるかと思います。もし、深いところまで勉強したければ、本やインターネットを使って独学することをお薦めします。

とりあえず、ルール的なもの。

**「質問をしよう！」**
まあ、質問はどんどんして下さい。わからないまま先に進んでも、余計にわからなくなるだけなので。それに、こっちが間違えていることもありうるので・・。

**「講義が終わったら打とう！」**
プログラミングは打つことから始まります。何も打たずに作品はできませんし、紙の上だけでやっていっても、エラーに気がつかないでしょう。この資料にかかれたソースを打ってみたり、数値を若干変えてみたり色々打っていきましょう。

**「本は買おう！」**
C 言語を学ぶために、参考となる本を手に入れて独学するのも、十分良い方法です。しかし、近くの図書館から借りるというのはあまりお勧めではないかもしれません。０円で済む分、途中でやる気がなくなっても、損することはないでしょう。でも、本を買った場合、ちゃんと全部読まなければ、その値段だけ損することになります。だから、簡単に諦められないように、本は買ったほうがいいですよ。

あと、開発環境に関しては、何でもいいです（Visual　Studio を推奨しますが)。
コンパイラがあれば最低限できるはずなので。わからなければ、個別に質問して下さい。

では、早速今日の講義始めますよ。

## １－０　『プログラミング言語とは』

プログラムとは機械に対する命令です。ただ、人間の言葉をそのまま機械に伝えても、機械は理解できません。なので、コンパイラという翻訳機を使って、機械がわかる言葉に変えてから実行されます。これにより、コンパイラの翻訳のルールさえわかれば、２進数とかをいちいち知らなくても機械に命令することが可能になりました。以下が簡単なルールです。

・ソース（プログラムを書いた文章）を上から１行ずつ翻訳していく。
・大文字と小文字、全角と半角は別物扱いされる。
・１つでもスペルが間違っていたり、何かを忘れていたりするとエラーが起こる。

細かいルールに関しては、後々出てくると思いますので、少しずつ覚えていきましょう。

## １－１　Hello World

まず、この C 言語講座では、コンソール（Windows ではコマンドプロンプト）というところで動作するプログラムを作っていきます。コンソールは作ったプログラムを一番簡単に実行できるところだと思ってください。

データを出力できるようにならないと、そのソースはどんな結果を出したかを見ることができません。最初に適当な文字列を出力するプログラムを作りましょう。

source1-1.cpp　Hello World を出力するソース

```
#include<stdio.h>

int main()
{
        printf("Hello World¥n");

        return 0;
}
```

まあ、プログラムが初めての人は何がなんだかわからないですよね・・。とりあえず、同じように打ってみて、何が出るかを見てみるといいと思います。以下は解説です。

１行目：#include<stdio.h>
printf( )を使用するために必要なものです。後で説明します。

２、３、６行目：int main( )　｛ ｝
メイン関数といいます。コンソールで動作するプログラムでは、メイン関数がプログラムのスタート地点になっています。そのメイン関数の中身が「｛ 」の部分から「 ｝」の部分までであるということを示しています。

４行目：printf(“Hello World¥n”) ;
このソースの一番重要部分です。
printf （） とは、コンソールに括弧内のものを表示する命令です。
あと、文字列は「”」で囲むというルールがあります。Hello World は文字列なので、「”」で囲みましょう。
もう１つ、「¥n」は何かというと、改行を表します。このような¥で始まる特殊な命令をエスケープシーケンスといいます。

主なエスケープシーケンスその１

| | |
|---|---|
| ¥n…文字列の改行を示す | ¥0…文字列の終了を示す |
| ¥t…タブを示す | |

また、エスケープシーケンスは「¥」、「”」、「’」といった特殊な文字を文字列内に使うときにも使用します。「”」を普通に書くと、文字列の始まりや終わりと勘違いされてしまうので気をつけましょう。

**主なエスケープシーケンスその２**

| | |
|---|---|
| ¥¥…¥という文字 | ¥”… “という文字 |
| ¥’… ‘という文字 | |

あと、printf( )の後に「；」（セミコロン）をつけましょう。日本語で言う「。」と同じで、文の終わりに必ずつけます。

５行目：return 0；
関数の終了をしめすものです。今はそう覚えておいて下さい。

解説は以上です。初心者の人は今の段階では、#include とか int main()とかはプログラムを打つときに必ず必要なもの（おまじない？）と思っていても結構です。printf のことはマスターしておきましょう。

## １－２　変数

次に進みます。ゲームプログラミングにおいて、一番必要なものはデータの管理です。「○○というｱｲﾃﾑは○個持っている」「プレイヤーの HP はあと○○だ」というように、ゲームの進行状況において、変動する数値を扱わなければなりません。そこで、変数というものを使います。これは何かの値を入れている箱のようなものです。ただ、箱にも大きさや形など種類がたくさんあるように、変数にもいくつかの種類があります。

**主な変数の型**

| | |
|---|---|
| int | 整数を扱う型。（約－２１億～２１億までの整数を格納可能） |
| float | 実数を扱う型。（３．４×１０－３８乗～３．４×１０３８乗） |
| double | 実数を扱う型。（１．７×１０－３０８乗～１．７×１０３０８乗） |
| char | 文字を扱う型。 |

（あと、各型の前に unsigned をつけると、符号無しの型になります）

ちなみに、型に合ってない数値を入れると、勝手に変更されてしまうので注意してください。
（たとえば、整数を扱う型である int に小数を入れようとすると、小数点以下の値が切り捨てられます）
では、変数を作り、その値をコンソール上に出力してみましょう。

source1-2.cpp　変数宣言と変数の値の出力

```
#include<stdio.h>

int main()
{
        int seisu;              /*整数を格納する変数*/
        double syousu;          /*小数を格納する変数*/

        seisu = 10;
        syousu = 1.2;
```

```
        printf("%dは整数、%fは小数¥n" , seisu , syousu);

        seisu = 3.14;              /*小数「3.14」を「seisu」に代入*/

        printf("%dは整数、元は3.14¥n" , seisu);

        return 0;
}
```

---

では、解説します。(#include や int main など１度やったところは省略しますよ)

４、５行目：int seisu；　double syosu；
　変数宣言といって、あらかじめ関数の最初にどんな変数を使うか宣言しなければいけません。また、変数の名前は任意の名前をつけることができます。数学みたく、x や y を使ってもいいんですが、今回ではわかりやすくするために整数（seisu)、小数（syousu）という名前にしました。この２行で、int 型の seisu と、double 型の syousu を宣言しています。
　あと、「/*整数を格納する変数*/」の部分についてですが、これは「コメント」というものです。コンパイラがソースを翻訳するときに、この「/*」と「*/」で囲まれた部分を無視します。つまり、人間だけが読める部分ということです。これから先、長いソースを書いていくことになると思いますが、何の値を入れている変数かを書いたり、どんな処理をしているかを書いたりしておけば、簡単に把握することができます。コメントをつける癖をつけておくといいですよ。

６，７行目：seisu = 10；　syousu = 1.2；
　変数の中に値を代入しています。C 言語の世界では、「＝」は、右辺の値を左辺のところに代入するという意味を持っています。(数学の「＝」とはちょっと意味が違うので注意)
　この２行で seisu には１０が、syousu には１．２という値を入れました。

８行目：printf("%d は整数、%f は小数¥n" , seisu , syousu);
　前回出てきた printf です。ちょっと形が変わっていますので、その部分だけ説明します。
　%d とか%f というのが、文字列の中に出てきました。これは型フィード文字といい、任意の値を表示するときに使います。

**主な型フィード文字**

| | |
|---|---|
| c …文字 | s …文字列 |
| d …整数 | f …小数 |

　型フィード文字を使ったとき、文字列の後に何の値を入れるかを書きます。ただし、どこまでが値を指しているかをわかるように，（コンマ）で区切って下さい。型フィード文字が複数あるときは、８行目と同じように１つずつ順番に書いていってください。（１つ１つコンマで区切るのを忘れずに）

　９、１０行目は６～８行目とやっていることは同じです。どのような結果になるかは、自分で打ち込んで試してみてください。

### １－３　入力（scanf）

これで、コンソール画面に出力することができました。次は入力です。

source1-3.cpp　入力

---

```
#include<stdio.h>

int main()
{
        int data;                    /*ユーザーに入力してもらった値を格納する変数*/

        printf("好きな整数を入力してください¥n");

        scanf("%d" , &data);     /*scanf関数（入力）*/

        printf("%dが入力されました¥n" , data);

        return 0;
}
```

---

６行目：scanf("%d" , &data);
　入力をする関数です。printf と見た目はほとんど変わりません。printf では、指定した値を型フィードの形に合わせて出力していました。scanf では、ユーザーが入力したものを型フィードの形に合わせて指定したところに代入しています。ただ、data の前に「&」がついています。これは４日目に説明する「ポインタ」に関するものです。とりあえず、scanf を使うときは頭に「&」をつけると覚えておいてください。

### １－４　演算

　今度はデータの演算を行います。「３つあったｱｲﾃﾑを１つ消費して、残りが２つになる」とか「ダメージ計算は、A の攻撃力÷B の攻撃力×・・・」というように、数値計算は非常に重要な部分です。ただ、数学の四則演算と大差ないため、簡単にできると思います。下記は演算記号の表です。

**演算記号一覧（基本）**

| | |
|---|---|
| ＋（プラス）…足し算 | －（マイナス）…引き算 |
| ＊（アスタリスク）…掛け算 | ／（スラッシュ）…割り算 |
| ％（パーセント）…割った余りを求める演算 | ＝（イコール）…代入（右の値を左へ入れる） |

　では、これらを使ってソースを書いてみましょう。

source1-4a.cpp　基本演算１

---

```
#include<stdio.h>

int main()
{
        int seisu1 , seisu2;

        seisu1 = 100;

        seisu1 = seisu1 + 3;
```

```
        printf("seisu1 = %d¥n" , seisu1);

        seisu2 = 3 * 5 - 60 / (2 + 1) % 6;

        printf("seisu2 = %d¥n" , seisu2);

        return 0;
}
```

---

まあ、説明の必要性はないと思います。seisu2 の値に関しては四則演算の法則がわかっていれば、すぐわかると思います。

あと、こういう書き方もあります。

| 記号 | 意味 | 例 | 例と同じ意味のもの |
|---|---|---|---|
| ＋＋ | １を足す | A＋＋ | A＝A＋１ |
| －－ | １を引く | A－－ | A＝A－１ |
| ＋＝ | 足して代入する | A＋＝B | A＝A＋B |
| －＝ | 引いて代入する | A－＝B | A＝A－B |
| ＊＝ | 掛けて代入する | A＊＝B | A＝A＊B |
| ／＝ | 割って代入する | A／＝B | A＝A／B |
| ％＝ | 余りを求めて代入する | A％＝B | A＝A％B |

source1-4b.cpp　基本演算２

```
#include<stdio.h>

int main()
{
        int seisu1 = 100;               /*変数宣言時に初期値を入れられる*/
        int seisu2 = 50;

        seisu1 += seisu2;               /*seisu = seisu1 + seisu2*/
        seisu1 %= 4;                    /*seisu = seisu % 4*/
        seisu1 ++;                      /*seisu = seisu + 1*/

        printf("seisu1 = %d¥n" , seisu1);

        return 0;
}
```

---

ちなみに、変数宣言の時に値（式）を代入することは可能です。これを変数の初期化と言います。演算部分は表を参考にしながら計算していけば、できると思います。

## １－５　乱数

今日の最後の部分です。乱数を出すプログラムを組みます。乱数とは、どれか１つの値をランダムに選んでくる数です。例えば、サイコロとか、じゃんけんとか出る値に法則性がないほうがいいものに使えます。乱数を使うためには rand()を使います。

---

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main()
{
        int seisu;

        srand((unsigned)time(NULL));

        seisu = rand();
        printf("乱数を出します・・・。¥n値は%dになりました。¥n", seisu);

        seisu = rand() % 10;
        printf("０～９までの乱数を出します・・・。¥n値は%dになりました。¥n", seisu);

        return 0;
}
```

---

では、解説していきます。

２，３行目：#include<stdlib.h>　#include<time.h>

乱数の計算方法が記載されている stdlib.h と時間計測関係のプログラムが書いてある time.h をインクルードする必要があります。忘れずに書きましょう。

７行目：srand((unsigned)time(NULL));

まず、srand()について説明。これは、rand()を使う際に乱数計算するための値を初期化するものです。あくまで、コンピュータは計算機ですので、乱数も計算で出しています。そのため、できるだけ法則性が無くなるように、初期化したときの値を毎回変動させなければなりません。そこで、srand の括弧の部分に、時間を測定する(unsigned)time(NULL)を入れることで、プログラムが起動するたびに、時間が異なるためランダムな値が出ているようにしています。

８行目：seisu = rand();

seisu という変数に rand()で求めた値を代入します。rand()で返す値は０～３２７６７までのどれかになっています。ただ、３万以上の範囲がある乱数を利用することは実際ないと思います。ですので、次のような改良を加えます。

１０行目：seisu = rand() % 10;

rand()で出た値を何かの定数で割って余りを出せば、小さな範囲の乱数が作れます。今回では１０で割って余りを出すことで、０～９までの１０個の範囲の乱数を出しています。

## １－６　まとめ

今日のまとめです。

| 内容 | 関数・書き方 | 必要なヘッダー<br>（#include を使うもの） |
|---|---|---|
| 画面出力 | printf(“文字列”)<br>printf(“～ %d ～” , 変数名 or 値) | stdio.h |
| 文字入力 | scanf(“%d” , &変数名) | stdio.h |
| 乱数 | srand((unsigned)time(NULL))<br>rand() | stdlib.h<br>time.h |
| コメント | /*コメントの内容*/（C++では「//」もある） | |

・変数宣言は関数の一番始めにやる（C++では途中でも可能）
・変数の型や型フィード文字を間違えないように。
・文の終わりは「;」をつける。
（演算記号、変数の型などは４～６ページ参照で・・）

## １－７　演習

ユーザーが入力した値と、乱数で出てきた値の２つの合計と平均を求めるプログラムを作りなさい。
（平均は小数で表示して下さい）