初心者用
C 言語講座

第 3 版 1999 年 10 月 13 日

文責：斎藤輪太郎

1. はじめに

この講座ではプログラムをほとんど組んだことがない初心者を対象に、Ｃ言語を使ってプログラミングの基本を説明していきます。

Ｃ言語は基本的にどんな動作をするプログラムでも組むことができる言語（汎用性がある）で、実行速度もかなり速く、また実用性がかなり高いこともあって、様々なソフトウェア開発に多く使われるプロフェッショナルな言語です。

では一緒にがんばりましょう。

2. 画面に文字を表示

早速プログラムを書いてみましょう。以下のようなファイルを作って下さい。ファイル名は hello.c とします。

```
#include <stdio.h>

main(){
        printf(“Hello, world!¥n”);
}
```

※各行のセミコロン；を忘れないようにして下さい。

そして次にコンパイルという作業を行います。以下のコマンドをたたいて下さい。

```
cc hello.c
```

すると、a.out というファイルができあがるので、

```
./a.out
```

と打ち込んでみましょう。Hello, world と表示されましたか？

ではプログラムの中味を説明していきます。

（再掲）

```
#include <stdio.h>                ①

main(){                           ②
     printf(“Hello, world!¥n”);   ③
}
```

①は文字列の表示などを行うことを宣言しています。。
②はここからプログラム本体が始まることを示しています。プログラム本体は main(){から始まり、}で終わります。
③Hello, world と出力することを宣言しています。printf は画面への出力を行います。最後の¥n は改行を行うことを意味しています。

どうですか？簡単ですね。

課題：printf を使って自分の名前を表示してみましょう。

3. 変数

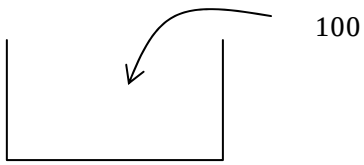
変数とはある値を(一時的に)保持しておく入れ物のことです。例えば次のプログラムを見て下さい。

```
main(){

    int num;
    num = 100;

}
```

これは num という変数に 100 という数値を代入しています。



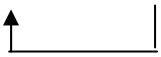
変数 num

int num は num という変数が整数を格納することを宣言しています。変数を使うときは必ずこのような宣言が必要になります。
次のプログラムを打ち込んで動作を確認しましょう。

```
#include <stdio.h>
main(){
    int num;
    num = 100;
    printf("I got number %d¥n", num);
}
```

printf(“%d¥n”, num)は num という変数の中身を表示する命令です。%d は整数の表示を意味します。実際に表示する変数名は上の例のように、ダブルクオテーション “ の後に書きます。

printf(“I got number %d¥n”, num);



```
I got number 100
```

注意：変数の宣言は main の最初の方で行います。

小数を宣言するときは、double を使います。

```
#include <stdio.h>
main(){
    double num;
    num = 12.34;
    printf(“I got number %lf¥n”, num);
}
```

num は今度は小数なので、printf で出力するときは、%lf をつけます。

文字を宣言するときは、char を使います。

```
#include <stdio.h>
main(){
    char c;
    c = ‘X’;
    printf(“I got letter %c¥n”, num);
}
```

c は文字なので、printf で出力するときは、%c をつけます。

課題：変数に 123 を代入し、それを表示してみましょう。

4. 演算

Ｃ言語では、四則演算など様々な演算を比較的簡単に行うことができます。

```
#include <stdio.h>
main(){
   double x, y, z;                    ①
```

```
   x = 100.0;                          ②
   y = 200.0;                          ③
   z = (x + y) / 2.0;                  ④
   printf(“Average is %lf¥n”, z);      ⑤
}
```

① ３つの変数 x,y,z を宣言しています。
② x に 100 を代入しています。小数の演算になることを明確にするために、100.0 と表記しています。
③ y に 200 を代入しています。
④ z に(x + y)/2 (このケースでは(100 + 200) / 2)を代入しています。
⑤ z の値を表示します。

④の等号の右辺を様々な形に変えることにより、様々な演算を行うことができます。
足し算は+, 引き算は-, 掛け算は * , 割り算は / で表します。括弧 (…) は普通の数式と同じように使用することができます。

課題：100.0 / 8.1 * 60 * 60 を変数を使って計算しましょう。

5. 条件分岐の基本 if 文

今までのところでは、記述した行は全て実行されましたが、今度は条件によっては実行されない行を記述する手法を学びましょう。以下のプログラムを打ち込んで実行してみて下さい。

```
#include <stdio.h>
main(){
   int x;
   x = 20;
   if(x > 10){ printf(“Hello!¥n”); }
   else { printf(“Hi!¥n”); }
```

Hello!と表示されましたか？では次に x = 20;を x = 5 に変えて実行してみましょう。今度はHi!と表示されましたか？

if 文は条件によって実行する文を変えます。書式は以下のとおりです。

```
if(条件){
```

```
    条件に合ったときの処理;
}
else {
    条件に合わなかったときの処理;
}
```

先程の例では、x > 10 が条件となります。従って x が 10 より大きいなら、printf(“Hello!¥n”);の方が実行され、そうでなければ、printf(“Hi!¥n”);のほうが実行されることになります。なお、else 以下のところは省略が可能です。

課題：例にならって n の値をプログラム中で決めて下さい。そしてその値が 100 より大きいなら Big number!、100 以下なら Small number!と表示するようにして下さい。

6. 繰り返しの基本 while 文

今までのところではプログラムの実行順序は上から下でした。ここではプログラムの流れを下から上へ戻す手法を学びましょう。以下のプログラムを打ち込んで実行してみて下さい。

```
#include <stdio.h>
main(){
  int x;
  x = 100;
  while(x <= 300){
     printf(“The number I got is now %d.¥n”, x);
     x = x + 50;
  }
}
```

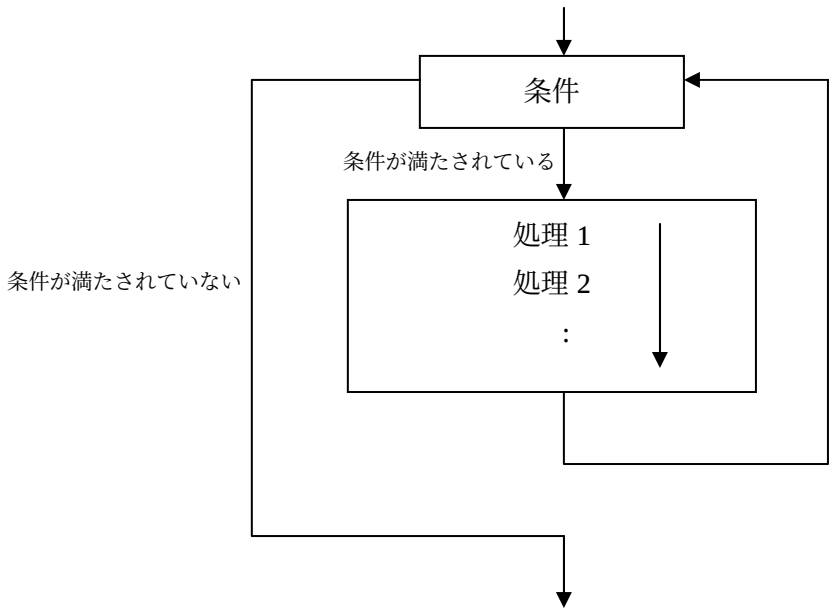
以下のような出力が得られると思います。

```
The number I got is now 100.
The number I got is now 150.
The number I got is now 200.
The number I got is now 250.
The number I got is now 300.
```

ちょっと難しくなってきましたが、なぜこのようになるのか、まず while 文から説明しましょう。while 文はある条件が満たされている間は指定した命令を実行しつづけるものです。書式は以下のようになっています。

while(**条件**){
　処理 1;
　処理 2;
　　:
}

まず**条件**が満たされているか調べられます。そして満たされているなら、{　}の中の**処理**が上から順番に下まで実行されます。そして再び**条件**が満たされているか調べ、満たされていればまた上から順番に**処理**が実行されます。これが繰り返されます。**条件**が満たされなくなったときに while 文の実行が終わります。



これを先ほどのプログラムに当てはめると、まず x が 100 に設定され、x が 300 以下の間は while 文の中が繰り返されることが分かります。x = x + 50 は x の値を 50 増やすという意味なので、x の値は while 文の中で 100,150,200,250,300 と変化することになります。それに伴い、その数が printf によって計５回表示されることになります。

課題：while 文を使って 1,3,5,7,..21 までの数を表示しましょう。
課題：while 文を使って 1 + 3 + 5 + … + 21 の答えを求めましょう。

7. ユーザ入力

これまでは変数の値はプログラムの中で決められていました。しかし、scanf を使うと、プログラムが走った後にユーザが値を入力することでそれを変数の値とすることができます。

scanf(“%d”, &n);

とすると、int 型変数 n にユーザが入力した値を代入することができます。ここでも、%d は入力の型が整数であることを示しています(&は変数の値をユーザ入力によって変えるようにするための記号ですが、ここでは細かい説明は省きます)。次のプログラムを見てみましょう。

```
#include <stdio.h>

main(){

   int x, y;
   printf(“Input first number: “);
   scanf(“%d”, &x);
   printf(“Input second number: “);
   scanf(“%d”, &y);
   printf(“%d + %d = %d¥n”, x, y, x + y);

}
```

以下が実行例です。

```
% ./a.out
Input first number: 10 ↵        ← ユーザ入力
Input second number: 20 ↵       ← ユーザ入力
10 + 20 = 30
%
```

上の例では scanf(“%d”, &x);は、ユーザの入力を変数 x に格納することを表しています。

課題：上の例にならってユーザから２つの数を入力してもらい、その積を出力するプログラムを書きましょう。

8.配列

似たような型のデータをたくさん扱いたいときに配列が便利です。配列はデータを一次元に並べて管理します。

例えば、

```
int array[5];
```

と宣言すると、array[0]から array[4]までの変数ができます。それぞれの変数には、

```
array[3] = 100;
```

のように値を入れることができます。
イメージ的には下の図のように数を入れる箱が５つできることになります。

array[0]	array[1]	array[2]	array[3]	array[4]

配列の宣言の一般的な書式は、

```
変数の型　変数名[ 配列中の要素数 ];
```

です。

次のプログラムを見てみましょう。

```
#include <stdio.h>

main(){
    int array[3];                    ①
    int x,n;

    n = 0;
    while(n < 3){                    ②
        printf("Please input number :");
        scanf("%d", &x);             ③
        array[ n ] = x;              ④
        n = n + 1;                   ⑤
    }
    n = 0;
    while(n < 3){                    ⑥
```

```
        printf("No. %d -> %d¥n", n, array[n]);   ⑦
        n = n + 1;
    }
}
```

以下がプログラムの実行結果の例です。

```
Please input number : 25 ↵        ユーザ入力
Please input number : 21 ↵        ユーザ入力
Please input number : 39 ↵        ユーザ入力
No. 0 -> 25
No. 1 -> 21
No. 2 -> 39
```

① 要素数が３の配列を宣言します。
② n が３より小さい間、while 文の中を繰り返します。３回繰り返すことになります。
③ 変数を入力します。
④ それを配列のｎ番目（０番目、もしくは１番目か２番目）に代入します。
⑤ n の値を１つ増やします。
⑥ 要素の中の数値を表示する為のループに入ります。
⑦ 要素番号と、その中の数値を表示します。

課題：１０個の数を入力してもらい、その合計を表示するプログラムを作成しましょう。
課題：１０個の数を入力してもらい、その平均を表示するプログラムを作成しましょう。

9.文字列

C 言語などで文字列とは、複数の文字が連なったもののことです。例えば、'a'、't' などは文字ですが、"Hello","Good"などは文字列です。C 言語では文字列を変数で扱う場合、文字の配列として扱われます。

例えば７文字以内の文字列を格納する変数を宣言したい場合､文字は char で宣言するので、

```
char str[7];
```

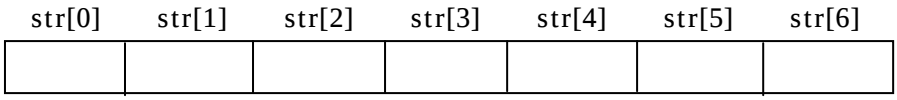
などどします。str は変数名です。
ここに実際に文字列を格納するときは、プログラムの最初の方に

```
#include <string.h>
```

という一行を入れた後、

```
strcpy(str, “Hello”);
```

とします。strcpy は右側の文字列を左側の配列へ代入する命令（関数）です。
イメージ的には各文字は以下のような形で配列に格納されます。¥0 は文字列の終わりを意味します。



str[1]は文字’e’を表すので、

```
printf(“%c¥n”, str[1]);
```

で’e’が出力されます。実際に確かめてみて下さい。文字列を一気に表示するには、

```
printf(“%s¥n”, str);
```

のように、%s を使って文字列を表示することを指定します。

途中から文字列を表示させることもできます。
例えば上の例で、

```
printf(“%s¥n”, &str[3]);
```

とすれば、”lo”と出力されます。&はかみくだいて説明すると、３番目の文字ではなく、３番目から始まる文字列であることを明確にします。以下に文字列変数の表現とその意味についてまとめておきます。プログラムを書く上で、文字と文字列ははっきり区別しなければならないことに注意して下さい。

```
str　… 文字列全部
str[3] … ３番目の文字
&str[3] … ３番目から始まる文字列
```

課題：Mothers という文字列を配列に格納し、表示して下さい。次に上の例にならってこの配列の１番目（０番目を最初の文字とする）から文字列を表示して下さい。

文字列処理に関する命令（関数）は他にもいくつかあります。

```
strcmp(str1, str2);  /* 文字列 str1 と str2 を比較。一致なら 0 となる */
strncmp(str1, str2, n); /* 文字列 str1 と str2 の最初の n 文字を比較し、
                           一致なら 0 */
strcat(str1, str2); /* 文字列 str1 の後に str2 をつなげる */
```

```
strlen(str1); /* 文字列 str1 の長さを返す
```

以下は strcmp を使った簡単な例です。実際に打ちこんで動作を確認してみましょう。

```
#include <stdio.h>
#include <string.h>

main(){
   char str1[20], str2[20];

   scanf(“%s”, str1);
   scanf(“%s”, str2);
   if(strcmp(str1, str2) == 0)printf(“Same strings¥n”);
   /* str1 と str2 を比較し、同じ(0)なら Same strings と表示 */
   else printf(“Different strings¥n”);
}
```

課題：２つの文字列をユーザに入力してもらい、それをつなげて出力するプログラムを作成しましょう。strcat を使うと便利です。

10.関数

ここでは簡単に関数について解説します。
Ｃ言語の関数とはかみ砕いて言えば、ある処理のまとまりを記述するものです。その処理のまとまりに対しては関数名が付けられ、これが呼ばれることによってそのまとまった処理が実行されます。

関数は以下のように定義します。、

```
関数名(){
   その関数の処理
     :
     :
}
```

またプログラムから呼び出すには、

```
関数名();
```

とします。

以下の関数は Hello と表示するものです。

```
print_hello(){
    printf(“Hello¥n”);
}
```

呼ぶ時は、

```
print_hello();
```

とします。

以下のプログラムを打ち込んで動作を確認しましょう。

```
#include <stdio.h>

/* 関数の定義 */
print_hello(){

    printf(“Hello”);

}

main(){

   print_hello();
   printf(“End of this program.¥n”);

}
```

プログラムは通常通り、main の先頭から実行されます。しかし、print_hello();の行のところで上の関数の定義のところに飛びます。そしてその中の一連の処理が実行された後、main に戻り、”End of this program”が表示されます。

関数の利点のひとつは、似たような処理がプログラム中で何回も現れるとき、それを関数に置き換えることでプログラムの行数を少なくできるということです。例えば、１０行で構成される全く同一の命令群がプログラム中５個所に出現するなら、この命令群を関数に

することによって(10 行×5) – (5 行（関数呼び出し） + 10 行(関数の定義)) = 35 行節約することができる計算になります。

課題：”Hello, world”と５回表示する関数を作りましょう。

関数に値を渡すこともできます。

```
print_sum(int x, int y)
{
    printf(“%d¥n”, x + y);
}
```

と定義しておいて、

```
print_sum(10,20);
```

と呼び出すと、print_sum 関数の中で x, y にそれぞれ 10, 20 が代入され、足し算が行われて 30 と表示されます。

関数に値を渡す（引数と呼ぶ）ときの形式は以下のようになります。

```
関数名(引数１の型 引数名１、引数２の型 引数名、…){
    引数を使った処理
      :
      :
}
```

課題：与えられた数 x, y の積を表示する関数を作成しましょう。

11.ファイル操作

ここでは他のファイルから情報を取りこむ方法を説明します。
例えば、あるファイルに書かれている内容をそのまま表示するにはどうすればいいのでしょうか（ＵＮＩＸの cat コマンドの働きをするプログラム）。以下にその手順を示します。

①まず、ファイルを扱う変数を宣言します。

```
    FILE *fp;
```

これはファイルを扱うための窓口になるものです。

②ファイルをオープンします。

```
fp = fopen(“File1”, “r”);
```

これは File1 という名前のファイルを読みこむことを宣言します。

③ ファイルからデータを読み込みます。いろいろな方法がありますが、例えば一行読みこみには以下のようにします。

```
fgets(line, 1000, fp);
```

これは File1 から一行を読みこみ、line という配列に格納します。但し、事前に以下のように line という配列を宣言しておく必要があります。

```
char line[1000];
```

④ ファイルの読みこみが終了したら、ファイルをクローズします。

```
fclose(fp);
```

さて、ファイルの中身を表示するプログラムの例を見てみましょう。

```
#include <stdio.h>

main(){
   char line[1000];
   FILE *fp;

   fp = fopen(“File1”, “r”);
   while(fgets(line, 1000, fp) != NULL){
      printf(“%s”, line);
   }
   fclose(fp);
}
```

以下の行

```
while(fgets(line, 1000, fp) != NULL){
```

が重要なポイントです。fgets は一行を読みこむ命令ですが、ファイルの最後まで読み終わってしまったときは、NULL という値になります。!=は等しくないという意味です。したがって、fgets(line, 1000, fp) != NULL はまずファイルから一行を読みこんでみて、“ファイルの内容を読みきっていなかったら”という意味になります。これに while がつ

くと、結果的にファイルの内容を読み終わるまで一行ずつファイルの中身を読みこんでいくことになります。

line に読みこんだ一行が入りますが、line は文字列として扱われるので、

```
    printf(“%s”, line);
```

で一行を出力することができます。

課題：行番号を付けてファイルの中身を表示しましょう。