[プログラミング入門Ｉ　３回目　２００３・９・１１]　繰り返しと配列

前回はプログラムの基本である変数、入出力、分岐を説明しました。さて、今回はいろいろな繰り返しについて説明しましょう。コンピュータはプログラムに書かれた内容を高速に実行します。１万回でも１００万回でも、繰り返しデータを処理することによって複雑なことでもあっという間に実行してくれます。

goto 文：実行の順番を変える

いまではあまり推奨されないやり方ですが、“万能”の制御文である goto 文から説明することにします。（なぜ、推奨されないかについては後で説明します。）まず、講義の一番最初にプログラムは書かれている順番に実行されると説明しました。**goto 文**はその順番を変えることができます。次のプログラムを見てください。

```
#include <stdio.h>
main()
{
    int x;
    x = 0;
 again:
    printf(“x=%d¥n”,x);
    x = x + 1;
    goto again;
    return 0;
 }
```

上のプログラムで、識別子に：（コロン）終わる部分（文）は**ラベル**と言います。ラベルは実行文ではなく、プログラムの位置を示します。Goto 文は、ラベルで示されているところから実行することを指示します。このプログラムでは、まず**x**に 0 を代入し、次に**x**の値を出力します。そのあとで、**x**の値に**1**を足したものを**x**に代入します。つまり、変数**x**の値に**1**を加えます。次の文は goto 文ですから、again:でしめされたところから、実行をします。**x**の出力、ここでは**x**の値に**1**を加え、その後、goto 文ですから、「永遠に」**x**を**1**づつ加えながら、**x**の値をプリントを繰り返すことになります。これでは、まずいので、99 でとめることを考えてみましょう。

while 文：繰り返しを書く

while 文は、ある条件が成立している間、文を実行するという繰り返し文です。では、0 から 99 までの数字を出力するプログラムをつくりましょう。

```
#include <stdio.h>
main()
{
    int x;
    x = 0;
    while(x < 100){
       printf(“x=%d¥n”,x);
       x = x + 1;
    }
    return 0;
 }
```

while 文は以下のように書きます。

While(条件式)　条件式が成立している間、実行する文

上のプログラムでは、while の中の文の中で x を**1**づつ増やして、条件式のところで x < 100 でなかったら、while 文の実行を終了します。これを、goto 文で書くと以下のようになります。

```
main()
{
    int x;
    x = 0;
  again:
```

```
    if(x < 100){
        printf("x=%d¥n",x);
        x = x + 1;
        goto again;
    }
    return 0;
}
```

どちらも同じことをするプログラムなのですが、どちらがわかりやすいでしょうか？ある条件が成立する間は、何かを実行するという形はいろいろな場面に現れてきます。そのときに、それがわかるように記述するというのが while 文です。

for 文：変数を 0 から１づつ増やす繰り返し

さらに、上で書いたようにある変数の値をある値からある値まで変化させるというもよく現れるパターンです。このようなときに便利なのが for 文です。では、上の例を for 文で書いてみましょう。

```
main()
{
    int x;
    for(x = 0;x < 100; x = x + 1)
      printf("x=%d¥n",x);
    return 0;
}
```

for 文は以下のような形式で使います。

for(初期化するための式；条件式　；繰り返し式)　繰り返し実行する文

for 文は、３つの式を書きます。最初の初期化をするための式は、ループの最初に実行される式です。その後、条件式が成り立っている間、繰り返し実行する文を実行と繰り返し式を繰り返し実行します。プログラムでは、はじめに、初期化するための式 x = 0 で、変数ｘを 0 にセットし、printf と x = x + 1 を条件式 x < 100 が成り立っている間実行します。つまり、上の while のプログラムと同じことをするわけです。While 文と goto 文を比べたのと同じように、while で書くのに比べてずっと簡単に書くことができます。また、このようなパターンは多くのプログラムに現れるため、for 文で書いておくことでコンパクトでわかりやすいプログラムになります。

もちろん、for 文の中に書く式は０から９９まで１づつ繰り返す式だけでなく、このパターンにあうものであれば、どのような式でもかくことができます。

記述を簡単にする演算子

プログラムには、よく現れるパターンがあります。たとえば、上の x=x+1 はそのひとつです。これは x++と書くことができます。これをつかうことによって、上の for 文は、

for(x = 0; x < 100; x++) …

と書くことができます。そのような演算子をまとめておきます。

変数++ ポストインクリメント。変数に１加える。但し、式としての値は１加える前の値となる。

++変数　プリインクリメント。変数に１加える。式としての値は１加えた後の値となる。

変数－－ ポストデクリメント。変数を１減らす。但し、式としての値は１減らす前の値となる。

－－変数　プリデクリメント。変数を１減らす。式としての値は１減らした後の値となる。

また、**変数　演算子= 式**は、変数 = 変数 演算子　式と同じ意味になります。たとえば、 x += 10 は、x = x + 10 と同じ意味になります。

break 文と continue 文：ループの停止と中断

while や for 文は、条件式が偽になるとループの実行を終了しますが、途中で実行を止めたい場合があります。ループ内で、break 文が実行されるとそのループの実行を直ちに実行を中断します。（例については教科書の１６８ページを参照してください。）

それに対して continue 文は、今実行している回のループの実行する文の実行をやめて、次の回のループに移ります。 for 文の場合には、次の回が実行される前に、繰り返し式が実行されてから、つぎの回の実行されることに注意してください。（例については、教科書の１７２ページを参照してください）

配列とは(一次元配列)

変数は、値を入れておく箱だと説明しました。**配列**とは、データを入れる箱を並べて、**何番目か（インデックス）で参照するためのデータ型**です。これがないと、表のようなデータを扱うプログラムが書くことができません。int や float の基本データ型の配列は、以下のように宣言します。

基本データ型　配列の名前[要素の数];

基本データ型は、整数の配列であれば、int ですし、浮動小数点数であれば、float や double を使います。要素とは配列の中に入っているデータのことで、例えば１０個の整数の配列 A は、

```
int A[10];
```

と宣言します。この要素の数（つまり、何個のデータが入るか）を**配列のサイズ**と言います。

参照は、以下のように書きます。

配列名[何番目かを指定する式]

ここで、注意しなくてはいけないのは、何番目かを数えるのにＣ言語では、０から数えます。つまり、最初の要素は A[0]です。したがって、最後の要素は、A[9]です。要素の指定のためには式を書くことができるので、変数 i をつかって、A[i]とも書くことができます。

変数と同じく、式の中に書くと配列の要素を参照し、左辺に書くと配列への代入になります。例えば、次の式

```
A[i*2+1]+10
```

は、i*2+1 番目の配列Ａの要素を取り出し、10 を足すということになり、次の文

```
A[i+3] = 100;
```

は、i+3 番目の配列Ａの要素のところに 100 を入れるということになります。

では、配列を使う例として、サイズ１０の配列に 10 個の数を入力して、それの合計を計算するプログラムをつくりましょう。

```
#include <stdio.h>
main()
{
    int i,k,s;
    printf(“Please input 10 numbers?”);
    for(i=0; i<10; i++){
       scanf(“%d”,&k);
       a[i]=k;
    }
    s = 0;
    for(i=0; i<10; i++) s += a[i];
    printf(“sum is %d¥n”,s);
    return 0;
 }
```

最初のループでは、printf でメッセージを出力した後、i を１づつ増やしながら、scanf で k に整数を入力して、それを a[i]に代入しています。二つ目の for ループでは、ｓに a[i]を足しこんでいます。s+=a[i]は、s = s + a[i]と同じであることに注意してください。

変数、配列の初期化

あらかじめ、変数に最初に値をセットしておく（初期化）と便利なことがあります。初期化は、以下のようにして変数の宣言と同時におくことができます。

```
int a=10;
```

これは、変数 a の宣言と同時に a には 10 をいれておくという意味になります。

配列も以下のようにして初期化することができます。

```
int test[4] = {1,3,4,1};
```

これは、配列の宣言と同時に、サイズ４の配列に順番に 1,3,4,1 という値を入れるという意味です。なお、このような初期化は宣言する時しかつかえません。プログラムの実行途中で、上のようにして代入はできませんので、注意しましょう。