ループ処理

ループ処理はある条件を定め、その条件が成立している間は繰り返し処理を行う方法である。日常的な出来事でいえば、

「お金がある限り買い物を続ける」
「品数が 10 品になるまで買い物を続ける」

といった事柄が繰り返し処理に当たる。C 言語でこれらの繰り返し処理を実現する文として、**while 文**、**for 文**、**do-while 文**といった文が用意されている。「お金がある限り」や「品数が 10 品になるまで」などのループを続けるための条件のことを**継続条件**という。これらの継続条件の判定方法には

- **見張り方式**
- **カウンタ方式**

がある。**見張り方式**はある条件が満たされている間は処理を繰り返す方式なので、「お金がある限り」や「品数が 10 品になるまで」などがこの方式の継続条件に当たる。また、**カウンタ方式**ではループ内にループした数をカウントする変数を設定し、この変数を用いて継続条件を決める方式である。そのため、「品数が 10 品になるまで」といった条件はカウンタ方式でも示すことができる。一般的には繰り返す数があらかじめ決まっている場合にはカウンタ方式が利用される。継続条件が常に「真」となるようなループを作ることもでき、これは**無限ループ**という。

while 文

while 文は「**条件式が“真”であれば本体を実行し続ける。条件式が“偽”になれば終了する**」という処理に使われる。そのため、前判定繰り返しともいう。ここで、本体とは while に続く{}で囲まれたブロックに記述された文の総称であり、while 文に限らず、for 文や do-while 文でも同様である。以下に書式と使用例を示す。

書式

```
while(条件式){
        文;
}
```

使用例

```
#include <stdio.h>

int main(void)
{
        int num = -1;
        int total = 0;

        while( num != 0 )
        {
                printf("数値を入力してください:");
                scanf("%d", &num);
                total = total + num;
        }

        printf("入力した値の合計値は%dです¥n", total);

        return 0;
}
```

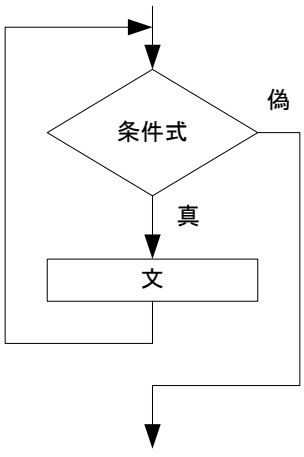

図 1　while 文

使用例のプログラムはキーボードから数値を入力し、入力された数値の合計値を求めて表示するプログラムである。このプログラムではキーボードから 0 を入力すると、total = total + num ; が実行された後にループが終了し、そこまでに入力された数値の合計が出力される。
このプログラムで重要なのは、初期化と呼ばれる次の処理である。

```
int num = -1;
int total = 0;
```

変数は宣言されると値を格納される場所が確保されるが、その中身である値はコンパイラによって自動的に決まる。そのため、値を入れてあげないとめちゃくちゃな値からプログラムがスタートしてしまう。このプログラムの場合、

```
while( num != 0 )  や  total = total + num;
```

で変数を利用しているので、あらかじめ値を入れておかないと正しく動作しない。

for 文

for 文は繰り返しの回数が予め分かっているときに一般的に使われる。「**初期値は○○、条件式が“真”であれば本体を実行し、条件を再設定して繰り返す。条件式が“偽”になれば終了する**」という構文に使われる。これも前判定繰り返しである。以下に書式と使用例を示す。

書式

```
for(式 1 ;  式 2 ;  式 3){
        文 ;
}
```

使用例

```
#include <stdio.h>

int main(void)
{
        int i;
        int num;
        int total = 0;

        for( i = 0; i < 5; i++ )
        {
                printf("数値を入力してください:");
                scanf("%d", &num);
                total = total + num;
        }

        printf("入力した値の合計値は%dです¥n", total);

        return 0;
}
```

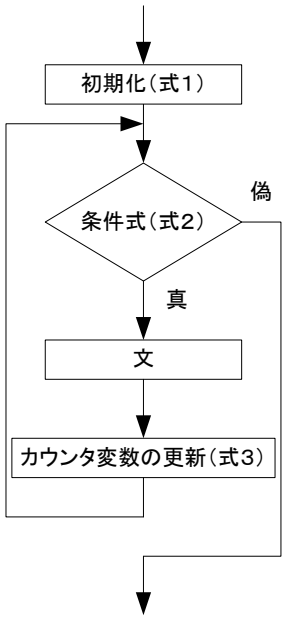

図２　for 文

このプログラムは while 文を用いて以下のように書き換えることができる。

使用例

```
#include <stdio.h>

int main(void)
{
```

```
        int i;
        int num;
        int total = 0;

        i = 0;                          /* for文の式1にあたる処理 */
        while(i < 5)                    /* for文の式2にあたる処理 */
        {
                printf("数値を入力してください:");
                scanf("%d", &num);
                total = total + num;
                i++;                    /* for文の式3にあたる処理 */
        }

        printf("入力した値の合計値は%dです¥n", total);

        return 0;
}
```

このプログラムの下線を引いた部分の処理を確認すると、それぞれが for 文の式 1 から式 3 の処理とまったく同じ役割を果たしていることがわかる。この for 文や while 文で用いた変数 i はループの実行回数の管理に用いるため、**カウンタ変数**と呼ばれる。

do-while 文

これは後判定繰り返しで、あらかじめ繰り返し回数が分からないときに使われることが多い。「**本体を実行し、条件式が“真”であれば本体を実行し続ける。条件式が“偽”になれば終了する**」という構文に使われる。よって、do-while 文は繰り返しをする回数が決められていなくて、最低 1 回は繰り返すべき処理を行う場合に使用すると便利である。以下に書式と使用例を示す。

書式

```
do{
        文；
} while(条件式)；
```

使用例

```
#include <stdio.h>

int main(void)
{
        int num;
        int total = 0;

        do{
                printf("数値を入力してください:");
                scanf("%d", &num);
                total = total + num;
        }while( num != 0 );  // 注：最後にセミコロン

        printf("入力した値の合計値は%dです¥n", total);

        return 0;
}
```

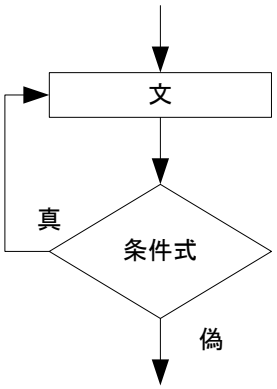

図 3
do-while 文

着目すべきところはやはり初期化部分の

```
        int num;
        int total = 0;
```

である。while 文の場合には、

```
        int num = -1;
        int total = 0;
```

であった。do-while文では先に本体が実行されるので、scanf("%d", &num);で値を入力し、合計を求める処理を行ってから、継続条件の判定を行うため、初期化を行わずにint num;としても、プログラム上問題無い。それではdo-while文を用いてfor文のような処理を行わせた処理を以下に示す。

使用例

```
#include <stdio.h>

int main(void)
{
        int i;
        int num;
        int total = 0;

        i = 0;                          /* for文の式1にあたる処理 */
        do
        {
                printf("数値を入力してください:");
                scanf("%d", &num);
                total = total + num;
                i++;                    /* for文の式3にあたる処理*/
        } while(i < 5);                 /* for文の式2にあたる処理*/

        printf("入力した値の合計値は%dです¥n", total);

        return 0;
}
```

多重ループ

ループ処理は以下のようにすると多重化することができる。ここに示した例では２重ループであるが、もちろん３重以上のループも作成できる。また、while 文、for 文、do-while 文は自由に組み合わせることができる。

書式

```
for(式１; 式２; 式３){
        for(式４; 式５; 式６){
        文１ ;
    }
}
```

使用例

```
#include <stdio.h>

int main(void)
{
        int i, j;
```

```
        for (i = 0; i < 5; i++) {
                for (j = 0; j < 3; j++) {
                        printf("%d ", i + j);
                }
        }

        return 0;
}
```

演習

テキストで示した使用例６つ全て作成し、実行結果を確認しなさい。

課題３　次のプログラムを作成しなさい。

kadai3-1（30 点）

(1)for 文を用いて、1～1000 までの合計値を求め、表示する。

(2)while 文を用いて、1～1000 までの合計値を求め、表示する。

(3)do while 文を用いて、1～1000 までの合計値を求め、表示する。

kadai3-2（30 点）

以下のような出力をする掛け算九九の表を作成する。

出力例

```
  1  2  3  4  5  6  7  8  9
  2 ･･･････････････････････ 18
  3 ･･･････････････････････ 27
  4 ･･･････････････････････ 36
  5 ･･･････････････････････ 45
  6 ･･･････････････････････ 54
  7 ･･･････････････････････ 63
  8 ･･･････････････････････ 72
  9 ･･･････････････････････ 81
```

ヒント
for 文を用いた２重ループを用い、九九の各値は計算式で求める。**表示桁数を調整して**見栄え良く出力すること。

kadai3-3（40 点）

ここまでに習ったことを使ってオリジナルのプログラムを作成する。時間に余裕がある人は複数作成して提出してもよい。ただし、少なくとも一つは作成して提出すること。

このプログラムについては、どのような動作を行うものかを簡単でよいので説明を付けること。

kadai3-3 は習った内容を理解し、作成していれば 25 点。

以下は 40 点の例です。

（１）プログラムにある程度の行数がある。

（２）とても良く応用できている。

（３）複数のプログラムを作成している。