

# for 文...繰り返し処理

三池 克明

while 文をより高機能にしたのが for 文です。  
while 文との違いを理解しスマートに使い分けられるようにしましょう。

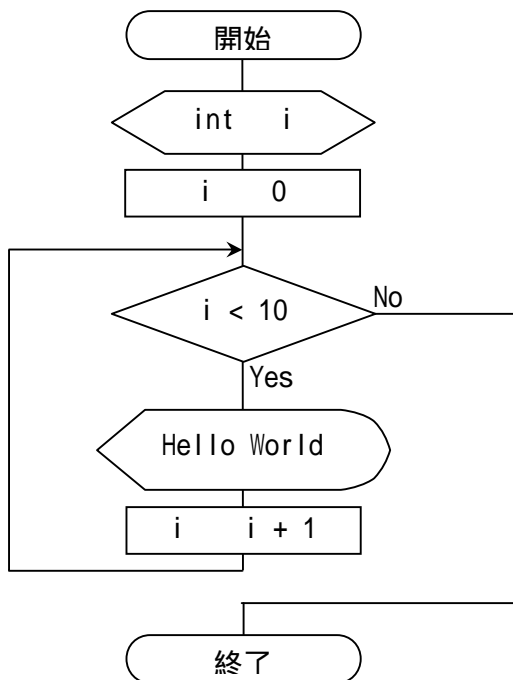
## 目 次

1.	繰り返すなら.....	1
1.1.	同じ処理を繰り返す (for 文版) .....	1
1.2.	for 文の動きを見る .....	5
1.3.	1、2、...、9、10の総和を求める (for 文版) .....	6
2.	活用例.....	8
2.1.	任意の自然数の総和を求める .....	8
3.	演習問題.....	15

## 1. 繰り返すなら

### 1.1. 同じ処理を繰り返す (for 文版)

以下は `while1.cpp` のフローチャートです。  
このソースプログラムを for 文で記述してみましょう。



以下が for 文で記述したソースプログラムです。  
入力し、コンパイル、実行してみましょう。

for1.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i;
6:
```

```
7:         for(i = 0; i < 10; i++){
8:             printf("Hello World\n");
9:         }
10: }
```

## 実行結果

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

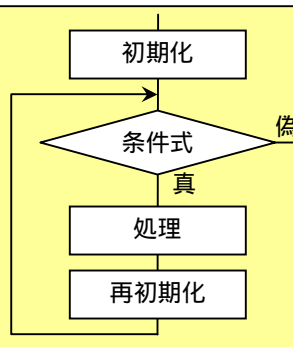
while1.cpp と同じ動作のようです。

以下が今回初めて使用した for 文の構文です。

このように while 文とは異なりループの直前とループ内の最後に処理を記述することができます。

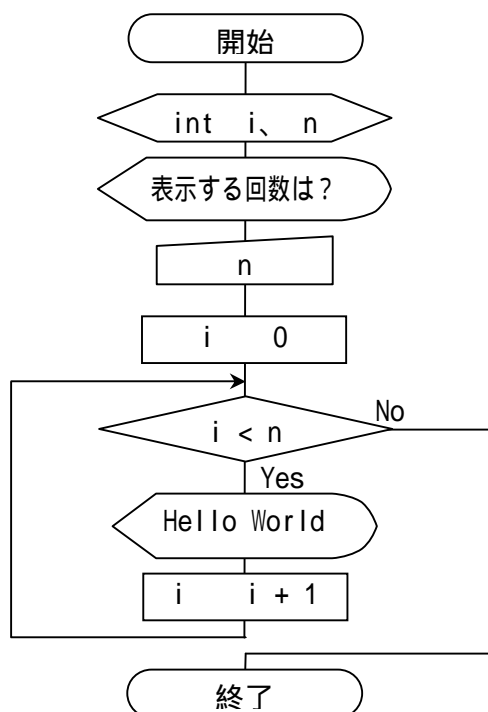
構文：

```
for(初期化; 条件式; 再初期化){
    処理
}
```



続いて while2.cpp も for 文で記述してみましょう。

以下は while2.cpp のフローチャートと、それを for 文で記述したソースプログラムです。



for2.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, n;
6:
7:     printf("表示する回数は? : ");
8:     scanf("%d", &n);
9:
10:    for(i = 0; i < n; i++){
11:        printf("Hello World\n");
12:    }
13: }
```

```
表示する回数は? : 14
```

```
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World
```

こちらもしっかりと同じ動作をしていることがわかります。

このように for 文と while 文はともにループ処理を表しますが、while 文がループの継続条件のみを記述するのに対し、for 文は初期化ならびに最初期化の処理を記述することができます。

---

## 1.2. for 文の動きを見る

---

ここで for 文の動きを見てみましょう。

for3.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i;
6:
7:     for(i = 0; i < 5; i++){
8:         printf("i = %d です。¥n", i);
9:     }
10:    printf("for の外では i = %d です。¥n", i);
11: }
```

実行結果

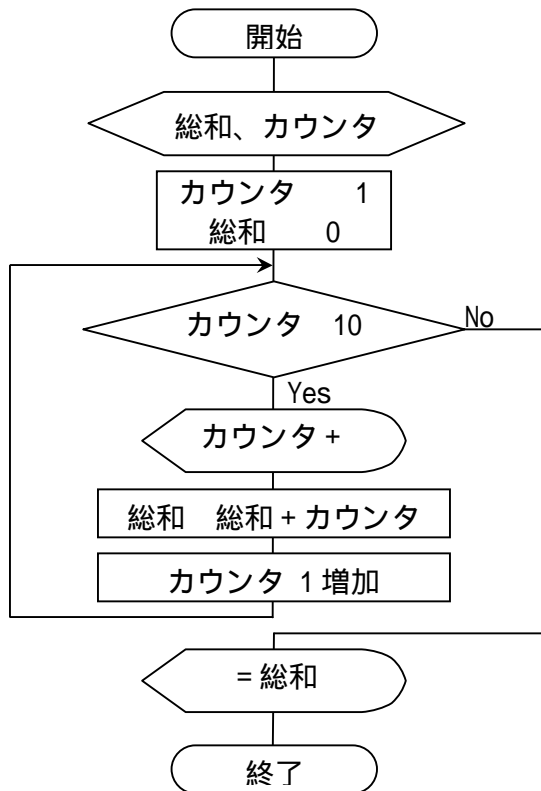
```
i = 0 です。
i = 1 です。
i = 2 です。
i = 3 です。
i = 4 です。
for の外では i = 5 です。
```

while 文と同じ実行結果であることがわかります。

### 1.3. 1、2、...、9、10の総和を求める (for 文版)

while6.cpp を for 文で記述してみましょう。  
以下はその変数リストとフローチャートです。

必要な変数	ソースコードでの変数名	型
総和	sum	int
カウンタ	i	int



```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, sum;
6:
7:     for(sum = 0, i = 1; i <= 10; i++){
8:         printf("%d+", i);
9:         sum += i;
10:    }
11:    printf("¥b=%d です。¥n", sum);
12: }
```

## 実行結果

1+2+3+4+5+6+7+8+9+10=55 です。

ここで7行目を見てみましょう。

```
6:
7:     for(sum = 0, i = 1; i <= 10; i++){
8:         printf("%d+", i);
```

この場合は、

- 初期化           sum = 0 と i = 1
- 条件式           i <= 10
- 最初期化        i++

となります。

初期化だけ式が二つありますが、このように「,(カンマ)」を間に挿入することで複数の処理を記述することができます。

これは最初期化でも同様です(条件式ではできません)。

## 2. 活用例

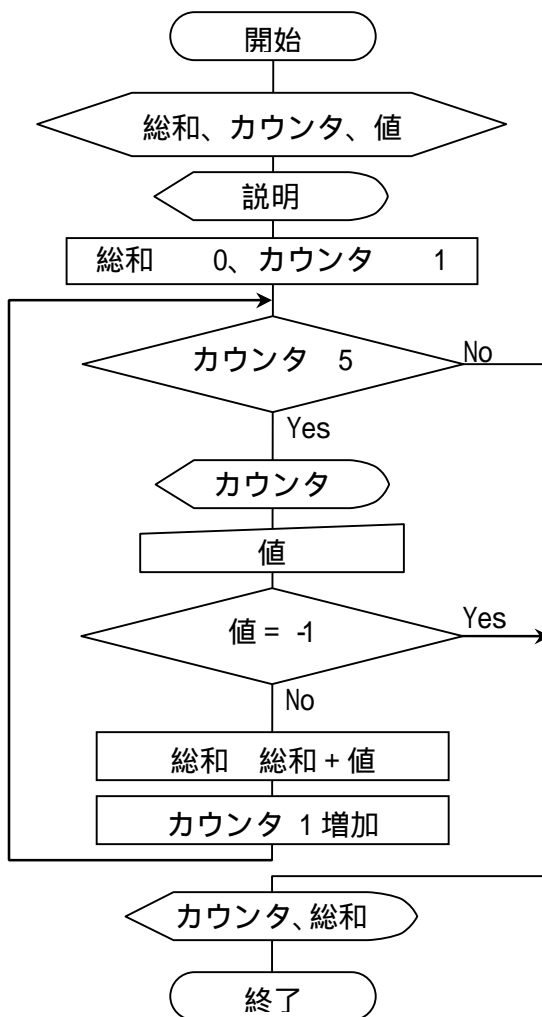
本節では for 文を使ったプログラムを挙げ、解説します。  
while 文同様、長い処理であっても簡単に記述することができるのを理解できればよいでしょう。

### 2.1. 任意の自然数の総和を求める

今回は任意の個数（最大 5 個）の自然数（正の整数）を入力させ、その個数と合計を表示させるプログラムを作ってみましょう。

以下はそのプログラムの変数リストとフローチャート、そしてソースプログラムです。

フローチャート上の 変数名	ソースコード上の 変数名	型	概要
総和	sum	int	自然数の総和
カウンタ	i	int	カウンタ (入力した整数の個数も兼ねる)
値	v	int	キーボードで入力された値



for5.cpp

```

1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, v, sum;
6:
7:     printf("任意の個数(最大5個)の自然数を入力してください。¥n");
8:     printf("終了するときは-1を入力してください。¥n");
9:
10:    for(sum = 0, i = 1; i <= 5; i++){

```

```
11:         printf("%d 個目 : ", i);
12:         scanf("%d", &v);
13:         if(v == -1){
14:             break;
15:         }
16:         sum += v;
17:     }
18:
19:     printf("データの個数 : %d、合計 : %d¥n", i, sum);
20: }
```

### 実行例 1

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

1 個目 : 10

2 個目 : 20

3 個目 : -1

データの個数 : 3、合計 : 30

### 実行例 2

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

1 個目 : -1

データの個数 : 1、合計 : 0

### 実行例 3

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

1 個目 : 1

2 個目 : 2

3 個目 : 3

4 個目 : 4

5 個目 : 5

データの個数 : 6、合計 : 15

おや？ データの個数が常に1多いですね。  
for文の初期化に問題があるのでしょうか？  
ためしに変数 i の初期値を0にしてみましょう。

for6.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, v, sum;
6:
7:     printf("任意の個数(最大5個)の自然数を入力してください。¥n");
8:     printf("終了するときは-1を入力してください。¥n");
9:
10:    for(sum = 0, i = 0; i <= 5; i++){
11:        printf("%d 個目 : ", i);
12:        scanf("%d", &v);
13:        if(v == -1){
14:            break;
15:        }
16:        sum += v;
17:    }
18:
19:    printf("データの個数 : %d、合計 : %d¥n", i, sum);
20: }
```

実行例 1

任意の個数(最大5個)の自然数を入力してください。

終了するときは-1を入力してください。

0 個目 : 10

1 個目 : 20

2 個目 : -1

データの個数 : 2、合計 : 30

## 実行例 2

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

0 個目 : -1

データの個数 : 0、合計 : 0

## 実行例 3

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

0 個目 : 1

1 個目 : 2

2 個目 : 3

3 個目 : 4

4 個目 : 5

5 個目 : 6

データの個数 : 6、合計 : 21

データの個数は問題ないのですが別の問題がおきてしまいました。

今度は「~個目」の数字が 1 小さいのと、入力できる自然数が最大 6 個になってしまいました。

解決法は色々ありますが今回は以下の方法で解決させました。

for7.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, v, sum;
6:
7:     printf("任意の個数(最大 5 個)の自然数を入力してください。¥n");
8:     printf("終了するときは -1 を入力してください。¥n");
9:
10:    for(sum = 0, i = 0; i < 5; i++){
11:        printf("%d 個目 : ", i + 1);
```

```
12:         scanf("%d", &v);
13:         if(v == -1){
14:             break;
15:         }
16:         sum += v;
17:     }
18:
19:     printf("データの個数 : %d、合計 : %d¥n", i, sum);
20: }
```

### 実行例 1

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

0 個目 : 10

1 個目 : 20

2 個目 : -1

データの個数 : 2、合計 : 30

### 実行例 2

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

1 個目 : -1

データの個数 : 0、合計 : 0

### 実行例 3

任意の個数(最大 5 個)の自然数を入力してください。

終了するときは -1 を入力してください。

1 個目 : 1

2 個目 : 2

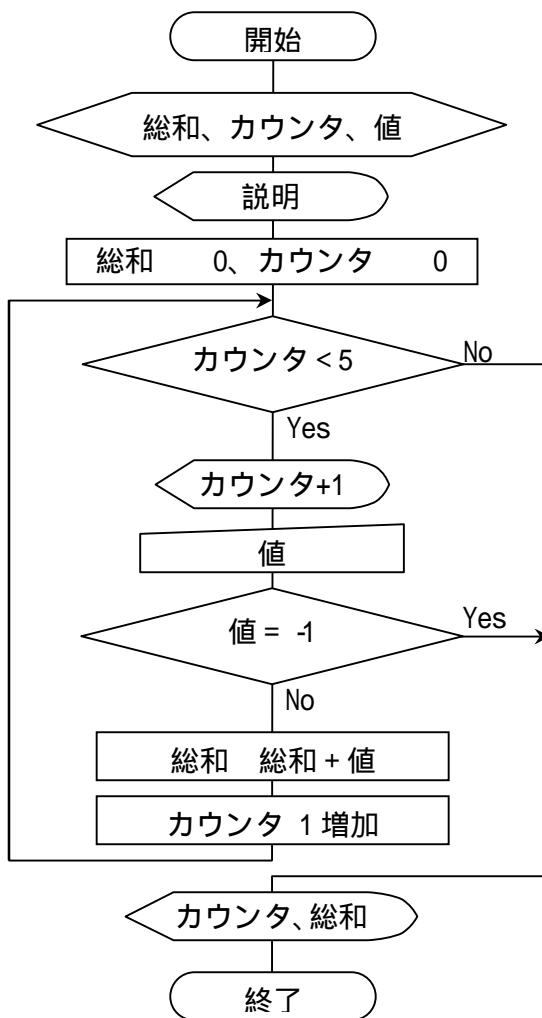
3 個目 : 3

4 個目 : 4

5 個目 : 5

データの個数 : 5、合計 : 15

今度こそ完成したようです。  
またこの完成版のフローチャートは以下の通りです。  
どこが変わったのかを確認しておきましょう。



### 3. 演習問題

- 7-1. 開始値と終了値を入力させ、開始値～終了値までの整数の総和を for 文を使って求めるプログラム「ensyu7 -1.cpp」 を作りなさい。なお、以下の実行イメージに従うこと。

実行例

開始値 : 12

終了値 : 23

12+13+14+15+16+17+18+19+20+21+22+23=210 です。

ヒント :

「ensyu6 -4.cpp」を for 文で処理させれば完成です。

- 7-2. 「for4.cpp」を改造して 100～140 までの偶数の総和を求めるプログラム「ensyu7 -2.cpp」を作りなさい。
- 7-3. 「ensyu7 -2.cpp」を改造して 12～60 までの 6 の倍数の総和を求めるプログラム「ensyu7 -3.cpp」を作りなさい。
- 7-4. 「for5.cpp」を改造して入力した自然数の個数、合計、平均を算出するプログラム「ensyu7 -4.cpp」を作りなさい。なお、以下の実行イメージに従うこと。

実行例

任意の個数 (最大 100 個まで)の自然数を入力してください。

終了するときは -1 を入力してください

1 個目 : 1

2 個目 : 4

3 個目 : 2

4 個目 : -1

データの個数 : 3、合計 : 7、平均 : 2.333333