

## 第4章 「プログラムの流れの繰返し」のまとめ

### 4.1 「do 文」のまとめ

#### do 文 (教 p.74)

```
do 文1 while ( 式1 ) ;
```

まず、文<sub>1</sub> (ループ本体と呼ばれる) を実行する。式<sub>1</sub> が \_\_\_\_\_ (真) である限り、\_\_\_\_\_ の実行を繰り返す。

注: 必ず 1 回はループ本体を実行する。

後述の while 文や for 文に比べると、do 文の実際の使用頻度は低い。

プログラムの実行が止まらなくなったときは、Ctrl-c で強制終了する。

#### 複合文 (ブロック) 内での宣言 (教 p.75)

(do 文に限らず) ブロックの中で宣言された変数は、その **ブロックでのみ有効** である。

#### 論理否定演算子とド・モルガンの法則 (教 p.77)

単項演算子の「  」は、真偽を逆にする演算子で、論理否定演算子とも言う。

#### 複合代入演算子 (教 p.80)

「\*=」, 「/=」, 「%=」, 「+=」, 「-=」, などのことである。例えば、sum += t は \_\_\_\_\_ とほぼ同じ意味になる。つまり、この代入を実行した後の sum の値は、実行する前の sum の値に t を加えた値になる。

#### 後置増分演算子と後置減分演算子・前置増分演算子と前置減分演算子 (教 p.81) (教 p.88)

a++	a の値を一つだけ増やす	(式全体の値は、_____ の値)
a--	a の値を一つだけ減らす	(式全体の値は、_____ の値)
++a	a の値を一つだけ増やす	(式全体の値は、_____ の値)
--a	a の値を一つだけ減らす	(式全体の値は、_____ の値)

Q 4.1.1 次のプログラムの断片の出力は何か? 答 \_\_\_\_\_

```
1  n = 3;
2  do {
3      printf("%d ", n);
4  } while (n-- > 0);
```

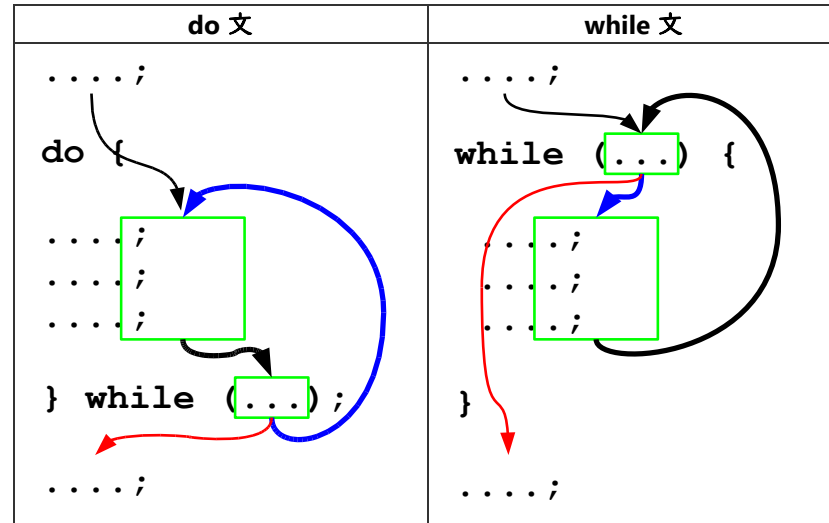
## 4.2 「while 文」のまとめ

### while 文 (教 p.82)

```
while ( 式1 ) 文1
```

式<sub>1</sub>が \_\_\_\_ (偽) でない限り、 \_\_\_\_ (ループ本体) の実行を繰り返す。

注: ループ本体が一度も実行されないことがある。



Q 4.2.1 次のプログラムの断片の出力は何か? 答 \_\_\_\_\_

```
1  n = 3;
2  while (n-- > 0) {
3      printf("%d ", n);
4  }
```

### 文字定数と putchar 関数 (教 p.86)

文字定数は 1 文字を \_\_\_\_\_ 「`'`」 ~ 「`'`」 で囲んだものことである。  
「`\n`」 や 「`\t`」 などの拡張表記は 1 文字として扱われる。

putchar 関数は、引数として受け取った文字を標準出力に出力する。

### 整数値を逆順に表示

Q 4.2.2 教科書 List 4-10 に似ている、次のプログラム (reverse.c) について...

```
1 #include <stdio.h>
2
3 int main(void) {
4     int no = 12345;
5
6     do {
7         printf("%d", no % 10); /* ① */
8         no /= 10; /* ② */
9     } while (no > 0); /* ③ */
10 }
```

```

11     return 0;
12 }

```

各行を実行後の no の値はどうか？

	1回目	2回目	3回目	4回目	5回目
7行め					
9行め					

また no の初期値が 12345 以外のときを考える。6～9行めの do～while 文の代わりに、教科書 List 4-10 と同じ次のような while 文に変更すると、no がいくつのときに、振舞いがどう変わるか？

```

1     while (no > 0) {
2         printf("%d", no % 10);
3         no /= 10;
4     }

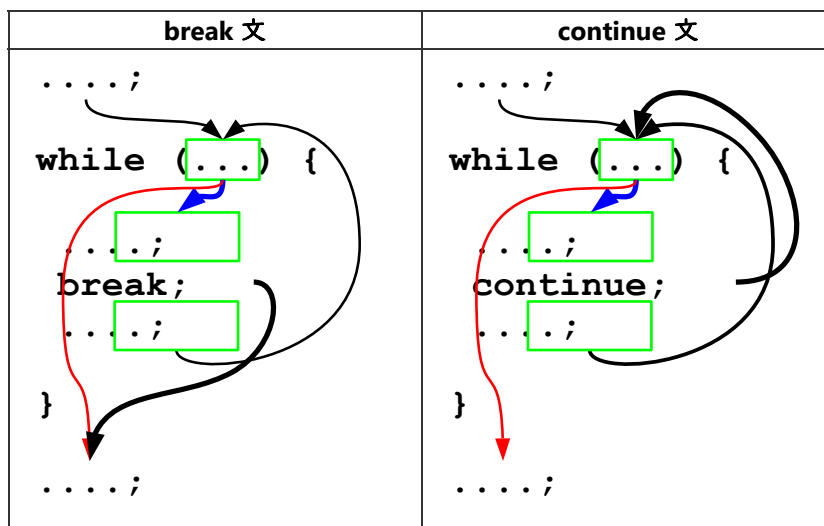
```

no が \_\_\_\_\_ のときに、変更前は、\_\_\_\_\_ が、変更後は、\_\_\_\_\_。

### break 文と continue 文 (教 p.92)

break 文は (もっとも内側の) 繰返し文 (do 文, while 文, for 文) を \_\_\_\_\_。  
(外側の繰返し文を一気に抜け出すことはできない。)

continue 文は (もっとも内側の) 繰返し文のはじめ (do 文, while 文の場合は条件式、for 文の場合は第3式) にもどる。



## 4.3 「for 文」のまとめ

### for 文 (教 p.94)

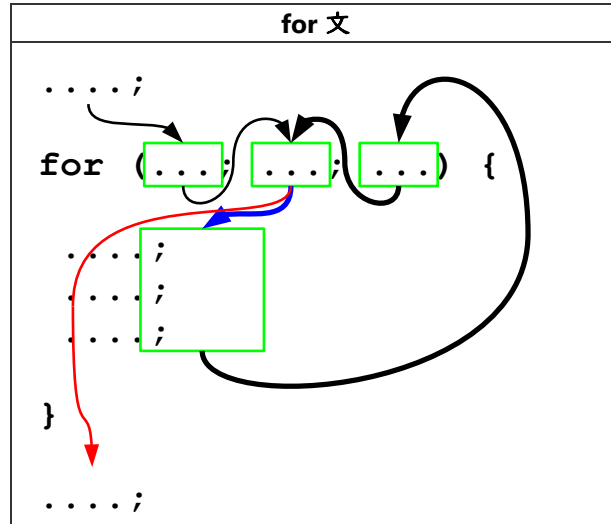
```

for ( 式1; 式2; 式3 ) 文1
for ( 宣言1 式2; 式3 ) 文1

```

ループに入る前にまず \_\_\_\_ (または \_\_\_\_ ) を実行する。  
 \_\_\_\_ が非0 (真) である間、 \_\_\_\_ (ループ本体) と \_\_\_\_ を繰り返し実行する。

詳細: 式<sub>1</sub>~式<sub>3</sub>は省略可能である。式<sub>2</sub>を省略したときは、1 (真) と書くのと同じ意味になる。



左のような for 文は右に示す while 文と (ほぼ) 等価である。

for 文	(ほぼ) 等価な while 文
<pre>for (A ; B ; C) {   ループ本体 }</pre>	<pre>____ ; while ( ____ ) {   _____   _____ ; }</pre>

では、なぜ左の書き方が好まれるか? — 繰り返しを制御する変数に対する処理が、一箇所にまとまっていて、一目でどのような繰り返しか理解しやすいからである。

### for 文による一定回数の繰り返し (教 p.97)

for 文には、良く使う決まり文句的な形がある。

- ① for (i = 0; i < n; i++) ... i が \_\_\_\_\_ n 回繰り返す
- ② for (i = 1; i <= n; i++) ... i が \_\_\_\_\_ n 回繰り返す
- ③ for (i = n; i > 0; i--) ... i が \_\_\_\_\_ n 回繰り返す
- ④ for (i = n - 1; i >= 0; i--) ... i が \_\_\_\_\_ n 回繰り返す

Q 4.3.1 上記の形で、n が 0 や負の数だった場合はどうなるか?

Q 4.3.2 上記のそれぞれの形でループを抜けたあとの i の値は何か? (ただし、n は非負とする。)

①      ②      ③      ④

**Q 4.3.3** つぎのような階乗の計算をするプログラム fact.c について...

```

1 #include <stdio.h>
2
3 int main(void) {
4     int i, n, fact = 1;
5
6     printf("正の数を入力してください。 ");
7     scanf("%d", &n);
8     for (i = 1 /* ① */; /* 通常 1 行を 3 行にわけた。*/
9           i <= n /* ② */;
10          i++ /* ③ */) {
11         fact = fact * i;
12     }
13     printf("あなたの入力した数の階乗は %dです。\\n", fact);
14
15     return 0;
16 }

```

変数 i, fact の値の変化 (n が 4 のとき) はどうなるか？

行	1 回め		2 回め		3 回め		4 回め	
	i	fact	i	fact	i	fact	i	fact
9								
11		→		→		→		→
10	→		→		→		→	

次のプログラム (polygon.c) は正 n 角形の座標を順に出力する。

```

1 #include <stdio.h>
2 #include <math.h> /* sin, cos のために必要 - 教 p.217*/
3
4 int main(void) {
5     int n, i;
6
7     printf("nを入力して下さい: "); scanf("%d", &n);
8     for(i = 0; i < n; i++) {
9         double theta1 = 2 * 3.1416 * i / n;
10        double theta2 = 2 * 3.1416 * (i + 1) / n;
11        printf("%.3f %.3f %.3f %.3f\\n",
12              100 * cos(theta1), 100 * sin(theta1),
13              100 * cos(theta2), 100 * sin(theta2));
14    }
15
16    return 0;
17 }

```

**式文と空文 (教 p.101)**

文 (statement) に以下を追加する。

分類	一般形	補足説明
空文	;	"何もしない"文、{} と書いても意味は同じ。

**4.4 「多重ループ」のまとめ**

## 2 重ループ (教 p.102)

for 文や while 文などのループ本体が、また for 文や while 文などの繰返し文を含んでいることを二重ループという。二重・三重・... ループをまとめて、多重ループという。特別な文法や実行規則があるわけではない。

**Q 4.4.1** 次の二重ループのプログラム (triangle.c) は、数字の三角形を出力する。

```
1 #include <stdio.h>
2
3 int main(void) {
4     int i, j, n;
5     printf("n を入力して下さい: "); scanf("%d", &n);
6     for (i = 1/*①*/; i <= n/*②*/; i++/*③*/) {
7         for (j = 1/*④*/; j <= i/*⑤*/; j++/*⑥*/) {
8             printf("%d", j % 10);/*⑦*/
9         }
10        printf("\n");/*⑧*/
11    }
12    return 0;
13 }
```

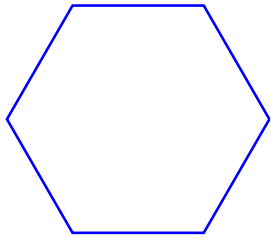
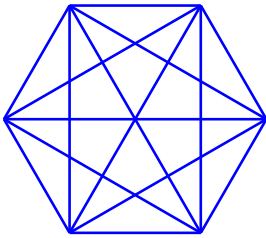
```
n を入力して下さい: 4
1
12
123
1234
```

n = 3 のとき、ループ内の式、文はどの順で実行されるか？

①② \_\_\_\_\_

**Q 4.4.2** 次の二重ループを使ったプログラム (diamond.c) は、ダイヤモンド図形という図形の座標を順に出力する。

```
1 #include <stdio.h>
2 #include <math.h> /* sin, cos のために必要 - 教 p.217 */
3
4 int main(void) {
5     int n, i, j;
6
7     printf("nを入力して下さい: "); scanf("%d", &n);
8     for (i = 0/*①*/; i < n - 1/*②*/; i++/*③*/) {
9         double th1 = 2 * 3.1416 * i / n/*④*/;
10        for (j = i + 1/*⑤*/; j < n/*⑥*/; j++/*⑦*/) {
11            double th2 = 2 * 3.1416 * j / n/*⑧*/;
12            printf("%.3f %.3f %.3f %.3f\n",
13                100 * cos(th1),
14                100 * sin(th1),
15                100 * cos(th2),
16                100 * sin(th2));/*⑨*/
17        }
18    }
19
20    return 0;
```

正 $n$ 角形 ( $n = 6$ )	ダイヤモンド図形 ( $n = 6$ )
	

$n = 4$  のとき、ループ内の式、文はどの順で実行されるか？

①② \_\_\_\_\_

### コンマ演算子 (教 p.233)

式<sub>1</sub>, 式<sub>2</sub>

という式は、式<sub>1</sub>、式<sub>2</sub>をこの順に評価し、\_\_\_\_の値を捨て、\_\_\_\_の値(と型)を持つ。

注: 関数を呼び出すときに引数を区切るコンマ(例: `printf("%d %d", i, j)`)はコンマ演算子ではない。

コンマ演算子の例 (comma.c)

```

1 #include <stdio.h>
2
3 int main(void) {
4     int i, j;
5     for (i = 0, j = 6; i < j; i++, j--) {
6         printf("i = %d, j = %d, ", i, j);
7     }
8     return 0;
9 }
```

**Q 4.4.3** このプログラムの出力はどうなるか？

\_\_\_\_\_

また  $i = 0, j = 5$  の場合はどうか？

\_\_\_\_\_

## 4.5 「プログラムの要素と書式」のまとめ

キーワード (教 p.108)

if や else など C 言語にとって特別な意味のある単語を \_\_\_\_\_ (keyword) と呼ぶ。変数名などに使用することはできない。(ただし、変数名などの一部に使用するのは構わない。)

### 自由形式 (教 p.110)

C 言語では、原則としてレイアウト (空白の数や改行) はプログラム の意味に影響を及ぼさない。空白がいくつ連続しても空白 1 文字と同じであり、改行も空白と同じである。

注意すべきところ:

- 前処理指令 (#include ..., #define ...) などの途中では、改行できない。
- 文字列リテラル、文字定数の途中でも、改行できない。

## 文法のまとめ

### 文 (statement)

に以下を追加する。

分類	一般形	補足説明
do 文	do 文 while ( 式 );	(教 p.74)
while 文	while ( 式 ) 文	(教 p.82)
continue 文	continue ;	(教 p.93)
for 文	for ( 式 ; 式 ; 式 ) 文	(教 p.94)
for 文	for ( 宣言式 ; 式 ) 文	(教 p.94)

### 式 (expression)

に以下を追加する。

分類	一般形	補足説明
後置演算	式 後置演算子	C の後置演算子は ++, -- のみ (教 p.81)
コンマ演算子	式, 式	(教 p.233)