

第4章 「プログラムの流れの繰り返し」の まとめ

4.1 用語のまとめ

教 p.72

do ~ while 文

```
do 文1 while ( 式1 );
```

まず、文₁ (ループ本体と呼ばれる) を実行する。式₁ 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。

あとで紹介する while 文や for 文に比べると、do ~ while 文の実際のプログラムでの使用頻度は低い。

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。

教 p.73

複合文 (ブロック) 内での宣言 (do ~ while 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。

教 p.75

論理否定演算子 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。

教 p.78

複合代入演算子 *=, /=, %=, +=, -=, 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・前置増分演算子

教 p.79

a++ a の値を一つだけ増やす (式全体の値は、増分 ___ の値)

教 p.86

a-- a の値を一つだけ減らす (式全体の値は、減分 ___ の値)

++a a の値を一つだけ増やす (式全体の値は、増分 ___ の値)

--a a の値を一つだけ減らす (式全体の値は、減分 ___ の値)

Q 4.1.1 次のプログラムの断片の出力は何か? 答 _____

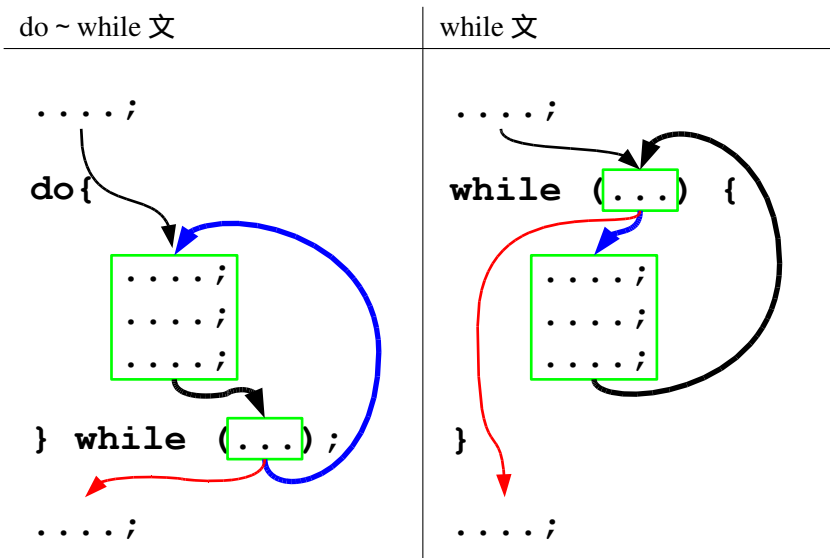
```
n = 3;  
do {  
    printf("%d_", n);  
} while (n-- > 0);
```

教 p.80 **while** 文

`while (式1) 文1`

式₁ 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 4.1.2 次のプログラムの断片の出力は何か? 答 _____

```
n = 3;
while (n-- > 0) {
    printf("%d_", n);
}
```

教 p.84

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

教 p.85

putchar 関数 引数として受け取った文字を標準出力に出力する。

教 p.90

for 文

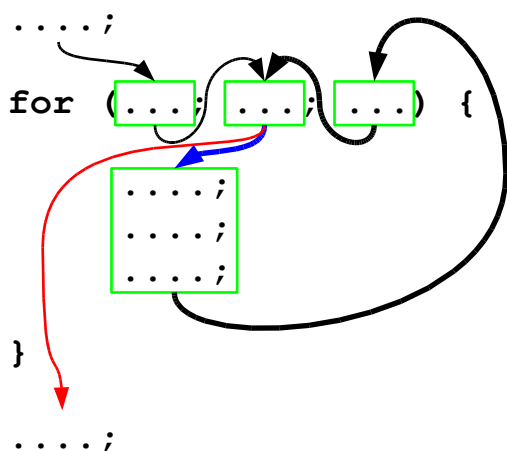
`for (式1; 式2; 式3) 文1`

ループに入る前にまず ____ 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₁ ~ 式₃ は省略可能である。式₂ を省略したときは、1 (真) と書くのと同じ意味になる。

for 文



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>

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。

教 p.92

一定回数の繰り返し 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 4.1.3 上記の形で、n が 0 や負の数だった場合はどうなるか？

Q 4.1.4 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 _____

空文 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、{}と書いても同じ。

教 p.96

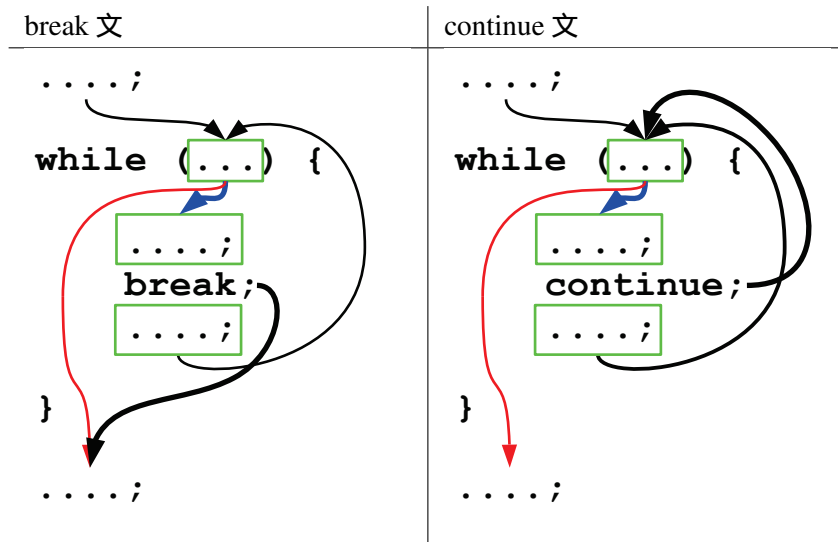
多重ループ for文やwhile文などのループ本体が、またfor文やwhile文などの繰返し文になっていることを二重ループという。二重・三重・…ループをまとめて、多重ループという。特別な文法や実行規則があるわけではない。

教 p.97

break文 (もっとも内側の)繰返し文(do~while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。)

教 p.101

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



教 p.102

キーワード 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、変数名の一部に使用するのは構わない。)

教 p.104

自由形式 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。

コンマ演算子

式₁, 式₂

という式は、式₁、式₂をこの順に評価し、____の値を捨て、____の値(と型)を持つ。

4.2 プログラム例

整数値を逆順に (reverse.c) (教科書 List 4-10 に類似)

```

1  #include <stdio.h>
2
3  int main(void) {
4      int no = 12345;
5
6      do {
7          printf("%d", no % 10);
8          no /= 10;
9      } while (no > 0);
10
11     return 0;
12 }
```

no の値の変化

	1 回目	2 回目	3 回目	4 回目	5 回目
7 行	_____	_____	_____	_____	_____
9 行	_____	_____	_____	_____	_____

Q 4.2.1 do ~ while 文の代わりに while 文を使うと、振舞いはどう変わるか？

典型的な for 文 (階乗の計算) (fact.c)

```

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; /* 通常 1 行に書くところを 3 行にわけた。*/
9           i <= n;
10          i++) {
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 }

```

i, fact の値の変化 (n が 5 のとき)

行	1 回目 i, fact	2 回目 i, fact	3 回目 i, fact	4 回目 i, fact	5 回目 i, fact
9	_, _	_, _	_, _	_, _	_, _
11	_, →_	_, →_	_, →_	_, →_	_, →_
10	→_, _	→_, _	→_, _	→_, _	→_, _

典型的な for 文 (正 n 角形の座標の出力) (polygon.c)

```

1  #include <stdio.h>
2  #include <math.h>
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 }

```

二重ループ (数の三角形) (triangle.c)

```
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; i <= n; i++) {
7          for (j = 1; j <= i; j++) {
8              printf("%d", j % 10);
9          }
10         printf("\n");
11     }
12     return 0;
13 }
```

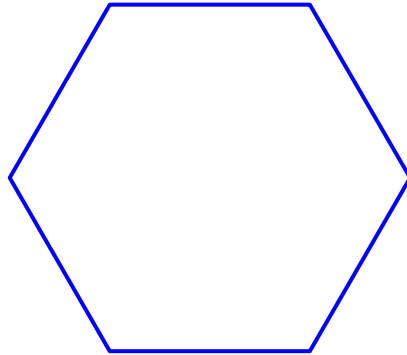
1
12
123
1234

Q 4.2.2 $n = 3$ のとき、ループ内の式、文はどの順で実行されるか？

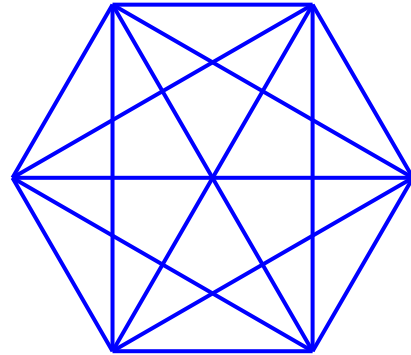
二重ループ (ダイヤモンド図形の座標の出力) (diamond.c)

```
1  #include <stdio.h>
2  #include <math.h>
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; i < n; i++) {
9          double theta1 = 2 * 3.1416 * i / n;
10         for (j = i + 1 /* 注意! */; j < n; j++) {
11             double theta2 = 2 * 3.1416 * j / n;
12             printf("%.3f_%.3f_%.3f_%.3f\n",
13                 100 * cos(theta1), 100 * sin(theta1),
14                 100 * cos(theta2), 100 * sin(theta2));
15         }
16     }
17
18     return 0;
19 }
```

正 n 角形 (n=6)



ダイヤモンド図形 (n=6)



Q 4.2.3 n = 4 のとき、ループ内の式、文はどの順で実行されるか？

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\n", i, j);
7      }
8      return 0;
9  }
```

Q 4.2.4 このプログラムの出力はどうなるか？

4.3 文法のまとめ

文 (statement) に以下を追加

分類	一般形	補足説明
do~while 文	do 文 while (式);	(教 p.72)
while 文	while (式) 文	(教 p.80)
for 文	for (式; 式; 式) 文	(教 p.90)
continue 文	continue ;	(教 p.101)

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.79)