

while 文…繰り返し処理

三池 克明

while 文を使えば同じような作業を繰り返すことができます。
これをループ処理と呼びます。
これはコンピュータが最も得意とする分野といえるでしょう。

—目 次—

1.	繰り返すなら.....	1
1.1.	同じ処理を繰り返す.....	1
1.2.	while 文の動きを見る.....	6
2.	活用例.....	7
2.1.	1、2、…、9、10の総和を求める.....	7
2.2.	1～10までの奇数の総和を求める.....	10
3.	do～while 文.....	11
3.1.	構文と処理の違い.....	11
3.2.	ゲーム風プログラムその2.....	12
4.	break 文と continue 文.....	14
4.1.	break 文.....	14
4.2.	continue 文.....	15
5.	演習問題.....	16

1. 繰り返すなら

1.1. 同じ処理を繰り返す

以下は「Hello World」を10行表示するプログラムです。
同じ処理を10回記述していますね。

hello2.cpp

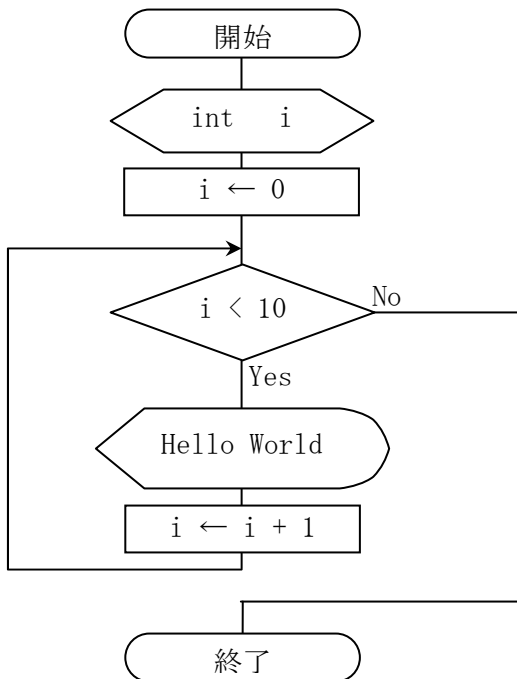
```
1: #include <stdio.h>
2:
3: main()
4: {
5:     printf("Hello World\n");
6:     printf("Hello World\n");
7:     printf("Hello World\n");
8:     printf("Hello World\n");
9:     printf("Hello World\n");
10:    printf("Hello World\n");
11:    printf("Hello World\n");
12:    printf("Hello World\n");
13:    printf("Hello World\n");
14:    printf("Hello World\n");
15: }
```

もし100回とか1000回繰り返すときも100行、1000行と記述しなければならないのでしょうか。

だとしたら大変面倒ですね。

そこでwhile文です。

以下のフローチャートは指定した条件を満たしている間（条件が真の間）処理を繰り返させます。



以下はそのプログラムです。
入力し、コンパイル、実行してみましょう。

while1.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i;
6:
7:     i = 0;
8:     while(i < 10){
9:         printf("Hello World\n");
10:        i++;
11:    }
12: }
```

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
```

それではコードをたどってみましょう。

```
6:
7:     while(i < 10) {
8:         printf("Hello World\n");
```

7行目の「while(i < 10)」は条件式 $i < 10$ が真の間、中カッコ {} 内の処理を繰り返すよう指示しています。

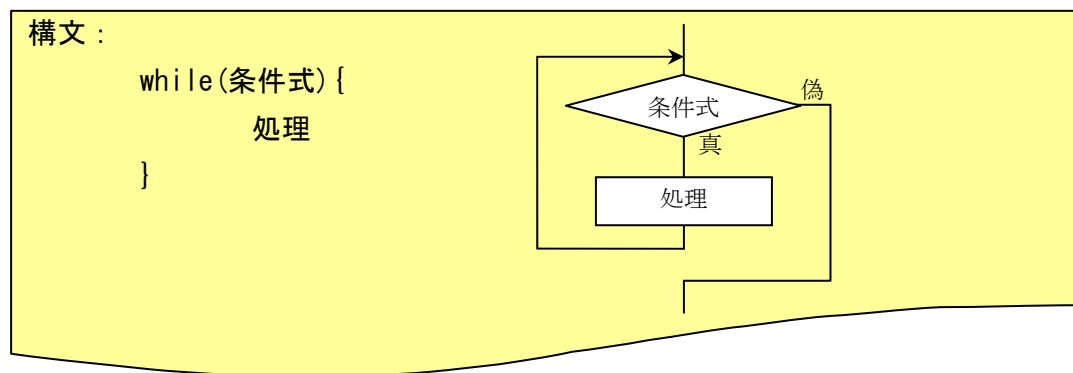
```
7:     while(i < 10) {
8:         printf("Hello World\n");
9:         i++;
10:    }
```

8～10行目が while 文で繰り返す処理です。

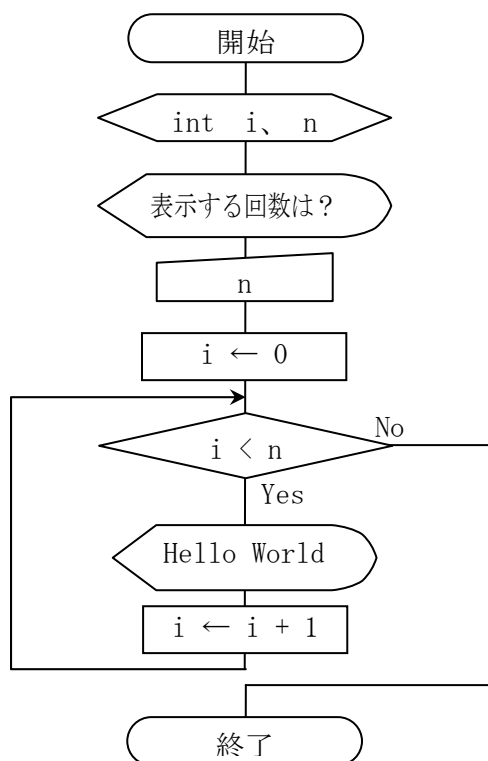
メッセージを表示してから、変数 i をインクリメント（1増加）させているのがわかります。

よって二回目、三回目…と処理が繰り返されるたびに i の値が1ずつ増えていきます。そして i が10になったとき $i < 10$ は偽になるため while 文から処理が抜け出します。

また、今回はじめて使用した while 文の構文は以下のとおりです。



なお、このように繰り返す処理のことをループ処理と呼びます。
続いて while1.cpp を表示回数を指定できるように改良してみました。
以下がそのフローチャートとソースプログラムです。



```
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:    i = 0;
11:    while(i < n){
12:        printf("Hello World\n");
13:        i++;
14:    }
15: }
```

実行例

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

このように繰り返す回数を自由に指定できるようになりました。

1.2. while 文の動きを見る

ここでもう少し while 文の動きをみてみましょう。

以下のソースプログラムを入力し、コンパイル、実行してください。

while3.cpp

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

実行結果

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

変数の動きがわかりましたか？

2. 活用例

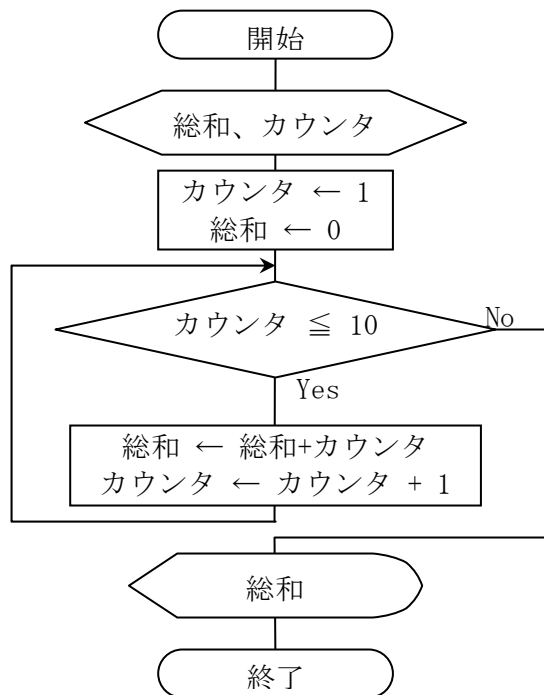
本節では while 文を使ったプログラムを挙げ、解説します。
長い処理であっても簡単に記述することができるのを理解できればよいでしょう。

2.1. 1、2、…、9、10の総和を求める

1+2+3+…+10 の計算をするプログラムを作ってみましょう。

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

フローチャートでの変数名	ソースプログラムでの変数名	型
総和	sum	int
カウンタ	i	int



while4.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, sum;
6:
7:     i = 1;
8:     sum = 0;
9:     while(i <= 10){
10:         sum += i;
11:         i++;
12:     }
13:     printf("1～10の総和は%dです。¥n", sum);
14: }
```

実行結果

1～10の総和は55です。

while4.cpp を途中式も表示されるように改良しました。
それが以下の while5.cpp です。

while5.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, sum;
6:
7:     i = 1;
8:     sum = 0;
9:     while(i <= 10){
10:         printf("%d+", i);
11:         sum += i;
12:         i++;
13:     }
14:     printf("=%dです。¥n", sum);
15: }
```

実行結果

1+2+3+4+5+6+7+8+9+10+=55 です。

式を表示させるようにしたのですが最後で失敗してしまいました。
惜しいですね。

この失敗に対応したのが以下の while6.cpp です。

while6.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, sum;
6:
7:     i = 1;
8:     sum = 0;
9:     while(i <= 10){
10:         printf("%d+", i);
11:         sum += i;
12:         i++;
13:     }
14:     printf("%b=%d です。%n", sum);
15: }
```

実行結果

1+2+3+4+5+6+7+8+9+10=55 です。

14 行目の関数 printf() で「%b」を表示させることで直前の余計な「+」を削除しています。これで式も参照できるようになりました。

また、このプログラムは 7 行目で開始値、9 行目で終了値が決まるのがわかります。ですから、この 2 つの数値を修正すれば、例えば -20~32 の総和を求めることも可能ですし、scanf() で入力した変数にすれば動作上で開始値と終了値を指定させることも可能です。

2.2. 1～10 までの奇数の総和を求める

続いて while6.cpp を改造して奇数のみの総和を求めるようにしてみました。

while7.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    i, sum;
6:
7:     i = 1;
8:     sum = 0;
9:     while(i <= 10) {
10:         printf("%d+", i);
11:         sum += i;
12:         i+=2;
13:     }
14:     printf("¥b=%d です。¥n", sum);
15: }
```

実行結果

1+3+5+7+9=25 です。

12 行目の「i++;」を「i+=2;」に修正しただけです。

ここで増分値を 1 から 2 にすることができました。

また、開始値は 1 なので変数 i は 1、3、5…と奇数で増えていきます。

これにより奇数の総和が求められるようになるわけです。

この部分を修正すれば 3 きざみや 5 きざみにすることが可能になることがわかります。

また開始値も調整すれば偶数だけでなく 3 の倍数や 6 の倍数の総和を求めることも可能です。

3. do~while 文

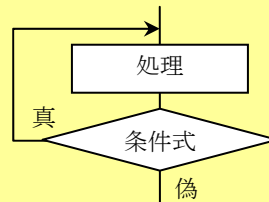
本節では while 文とよく似た構文を持つ do-while 文について解説します。
広義では while 文と do-while 文の 2 つをあわせて「while 文」と呼称することもあります。

3.1. 構文と処理の違い

do-while 文の構文は以下のとおりです。

構文：

```
do {  
    処理  
} while(条件式)
```



while 文との違いは処理の後に条件判断を行う点です。
これにより最低一回は処理をさせることが可能です。

3.2. ゲーム風プログラムその2

ゲーム風プログラム `switch6.cpp` を改造して行動の選択を繰り返せるようにしました。

それが以下の `while8.cpp` です。

`while8.cpp`

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    cmd;
6:
7:     do {
8:         printf("どうしますか?¥n");
9:         printf(" 1. 調べる¥n");
10:        printf(" 2. 寝る¥n");
11:        printf(" 3. 移動¥n");
12:        printf(" 4. 終了¥n");
13:        printf("¥n 番号を選択してください(1-4):");
14:        scanf("%d", &cmd);
15:        switch(cmd) {
16:            case    1:
17:                printf("何も見つからなかった。¥n");
18:                break;
19:            case    2:
20:                printf("あなたはぐっすり寝た¥n");
21:                break;
22:            case    3:
23:                printf("あなたはこの場を立ち去った。¥n");
24:                break;
25:            case    4:
26:                printf("終了します。お疲れ様でした。¥n");
27:                break;
28:            default:
29:                printf("その番号は選択できません¥n");
30:        }
31:        printf("¥n¥n");
32:    } while(cmd != 4);
33: }
```

どうしますか？

1. 調べる
2. 寝る
3. 移動
4. 終了

番号を選択してください(1-4):2

あなたはぐっすり寝た

どうしますか？

1. 調べる
2. 寝る
3. 移動
4. 終了

番号を選択してください(1-4):4

終了します。お疲れ様でした。

このように行動の選択を繰り返せるようになり、また終了させることもできるようになりました。

4. break 文と continue 文

while 文を使えばループ処理（繰り返し処理）を記述できますが、場合によっては強制的にループを抜けたり、あるいはループの先頭に処理を戻したい場合もあります。

そのときには本節で解説する break 文や continue 文を使います。

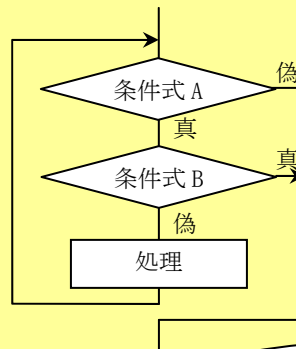
4.1. break 文

switch 文の中カッコ {} から処理を抜けるのに使用した break 文ですが、while 文の中カッコ {} から処理を抜け出すこともできます。

構文は以下のとおりです。

構文：

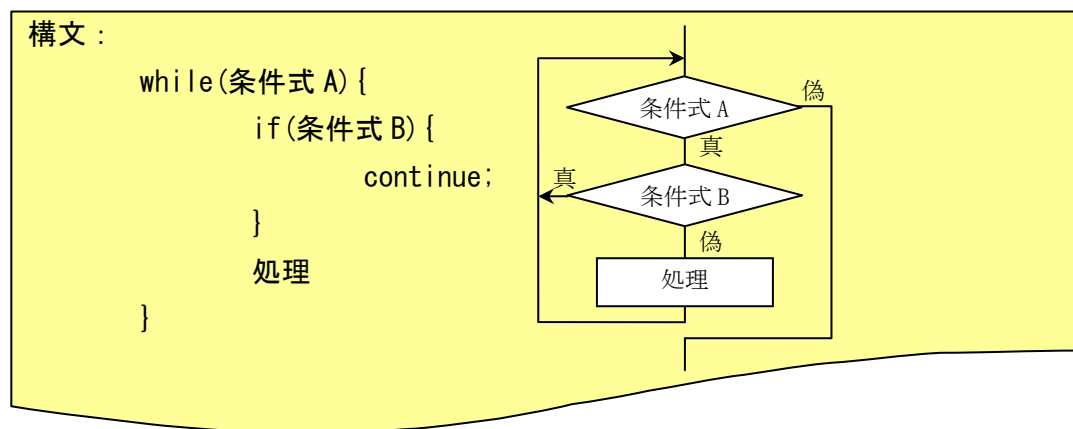
```
while(条件式 A) {  
    if(条件式 B) {  
        break;  
    }  
    処理  
}
```



通常、break 文はこのように if 文などの分岐処理と組み合わせることが多いようです。

4.2. continue 文

一方、continue 文は、while 文の中カッコ {} の先頭に処理を戻します。
構文は以下のとおりです。



こちらも break 文と同様に if 文などの分岐処理と組み合わせることが多いよう
です。

5. 演習問題

- 6-1. 1~100 までの整数の総和を求めるプログラム「ensyu6-1.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

```
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25+26+27
... (中略) ...
84+85+86+87+88+89+90+91+92+93+94+95+96+97+98+99+100=5050 です。
```

ヒント :

「while6.cpp」を改造すればすぐできますね。

- 6-2. 「ensyu6-1.cpp」を改造して 20~50 までの整数の総和を求めるプログラム「ensyu6-2.cpp」を作りなさい。
- 6-3. 「ensyu6-2.cpp」を改造して 80~100 までの整数の総和を求めるプログラム「ensyu6-3.cpp」を作りなさい。
- 6-4. 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求めるプログラム「ensyu6-4.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

```
開始値 : 12
終了値 : 23
12+13+14+15+16+17+18+19+20+21+22+23=210 です。
```

- 6-5. 「while7.cpp」を改造して 1~10 までの偶数の総和を求めるプログラム「ensyu6-5.cpp」を作りなさい。
- 6-6. 「ensyu6-5.cpp」を改造して 1~100 までの 3 の倍数の総和を求めるプログラム「ensyu6-6.cpp」を作りなさい。