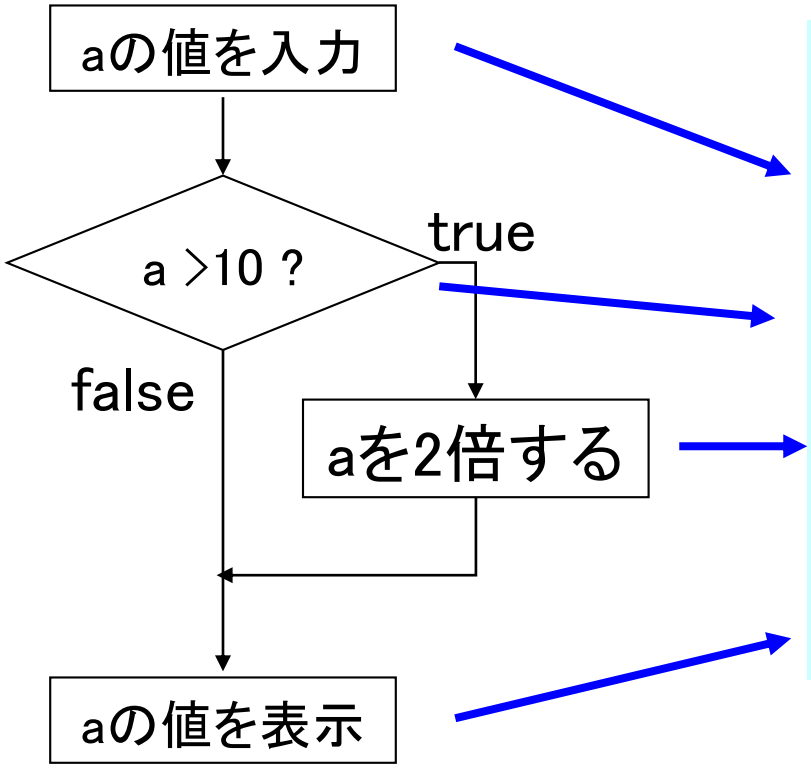
復習

if ～ 選択制御文(条件分岐，if文)




```
int a;

scanf("%d", &a);

if ( a > 10 )
{
    a = 2*a;
}

printf("aの値は%d¥n", a);
```

キーワード

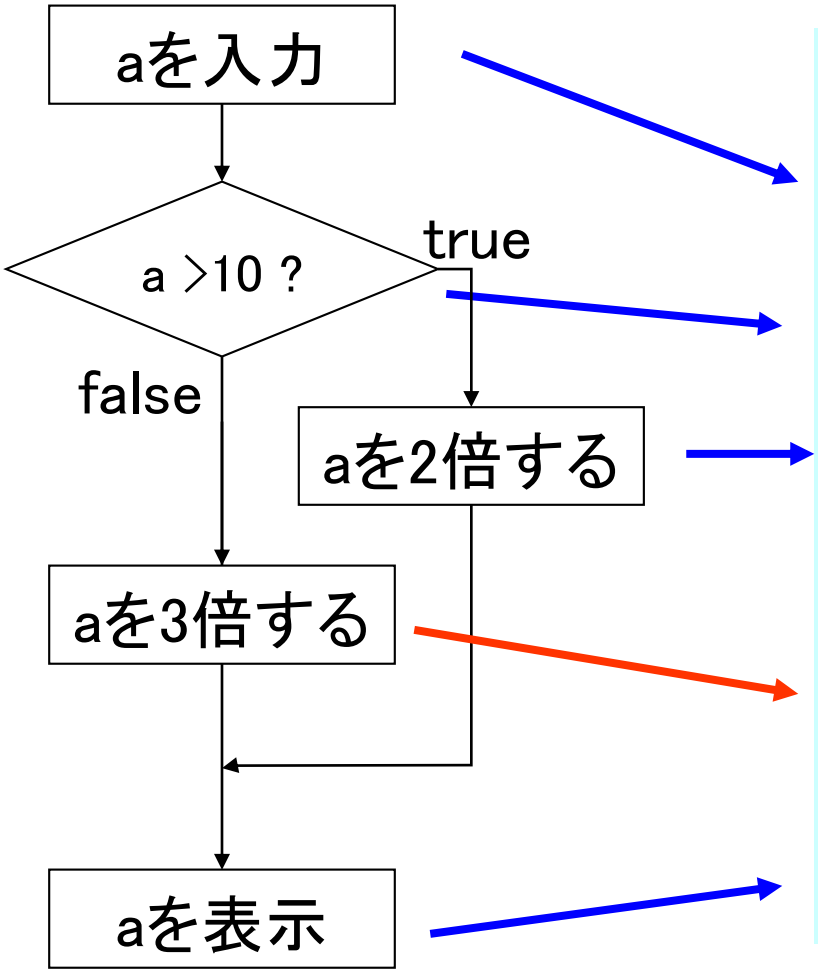
カッコが必要

条件

真(true)なら
この中が
実行される

if ～ else～選択制御文(条件分岐，if～else～文)

復習




```
int a;

scanf("%d", &a);

if ( a > 10 )
{
    a = 2*a;
}
else
{
    a = 3*a;
}

printf("aの値は%d¥n", a);
```

真(true)なら
この中が
実行される

偽(false)なら
この中が
実行される

復習

いろいろな条件式と関係演算子

	C のソース
a ＞ b	`if (a > b)`
a ≧ b	`if (a >= b)`
a ＜ b	`if (a < b)`
a ≦ b	`if (a <= b)`
a ＝ b	`if (a == b)`
a ≠ b	`if (a != b)`
5 ＜ a かつ a ＜ 10	`if (5<a && a<10)`
10 ＞ a または 15 ＜ a	`if (10>a || 15<a)`
10 ＞ a ではない	`if ( !(10 > a) )`

大原則
「=」記号は常に右側に付く

半角文字

！注意！
等号が二つ

これはダメ
5 ＜ a ＜ 10

かつ

または

否定

単文と複文

復習

文は命令(関数)や式

C言語の文は単文か複文のどちらか

単文

```
printf("負の数です¥n");
```

```
a = 2*a;
```

単文はセミコロンで終わる
(単文以外はセミコロン不要)

複文

波カッコで複数の単文をひとまとめ

```
{
    b = 10;
    a = 2*b;
    printf("aは%dである¥n",a);
}
```

複文

b = 10; 単文

a = 2*b; 単文

printf("aは%dである¥n",a); 単文

```
{
    a = 2*a;
}
```

a = 2*a; と単文で書いても同じ

if～else～文の文法

```
if (条件)
   文1
else
   文2
```

文1と文2はどちらも，単文でも複文でもOK

復習

if～else～文の書き方

```
if (条件)
   文1
else
   文2
```

単文で書いた場合

```
#include <stdio.h>
int main(void)
{
  int data;

  scanf("%d", &data);
  if ( data < 0 )
      printf("負の数です¥n");   文1
  else
      printf("正の数です¥n");   文2
}
```

複文で書いた場合

```
#include <stdio.h>
int main(void)
{
  int data;

  scanf("%d", &data);
  if ( data < 0 )
  {                              文1
      printf("負の数です¥n");
  }
  else
  {                              文2
      printf("正の数です¥n");
  }
}
```

この例では単文で書いても複文で書いても意味の違いはないが，ミスが減るのでこちらがおススメ

複雑なif～else～文

復習

```
if (条件)
   文1
else
   文2
```

文

```
if (条件1)
    文1
else
    if (条件2)
        文2
    else
        文3
```

文2

if文も全体で一つの文である.

```
int main(void)
{
    int x;
    printf("整数値xを入力してください：");
    scanf("%d", &x);
    if (x < 5)
    {
        printf("xの値は5未満です¥n");
    }
    else if (x >= 5 && x < 10)
    {
        printf("xの値は5以上10未満です¥n");
    }
    else
    {
        printf("xの値は10以上です¥n");
    }
}
```

文1

文2

文2

文3

整数値xを入力してください：3
xの値は5未満です

整数値xを入力してください：8
xの値は5以上10未満です

整数値xを入力してください：12
xの値は10以上です

反復処理(反復制御文)

これがコンピュータの仕事だ

プログラムにおける反復処理→
(a) ある処理を決まった回数繰り返す処理
(b) 一定の条件を満たす間繰り返す処理

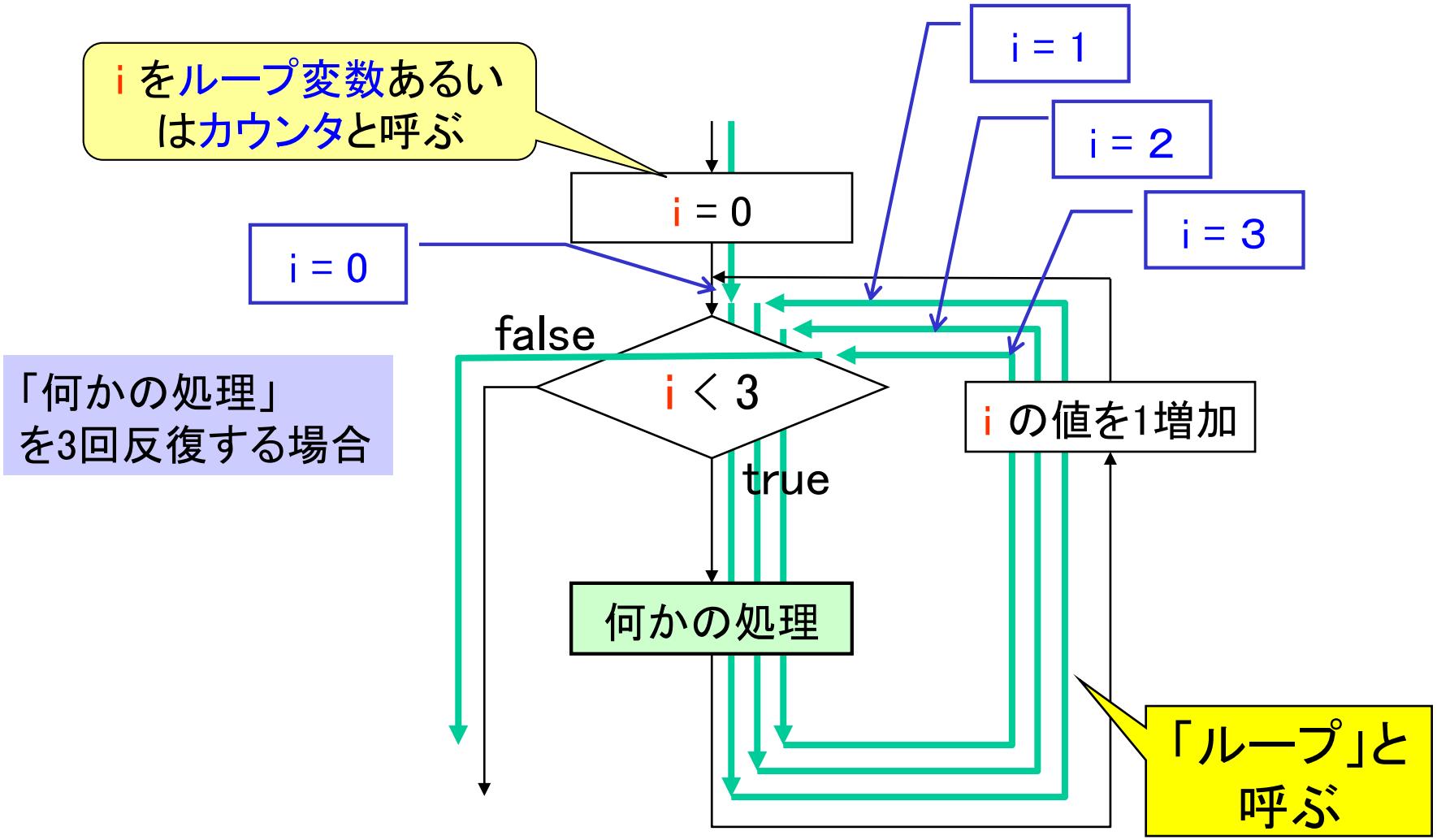
繰り返す回数は決まっていない

C言語における反復処理 → 3種類

(a) 決まった回数反復処理を行う　for 文

(b) 条件が成り立つ間反復処理を行う
i) 反復処理の初めに条件を調べる場合　while 文
ii) 反復処理の終わりに条件を調べる場合　do while 文

一定回数を繰り返す反復処理の考え方



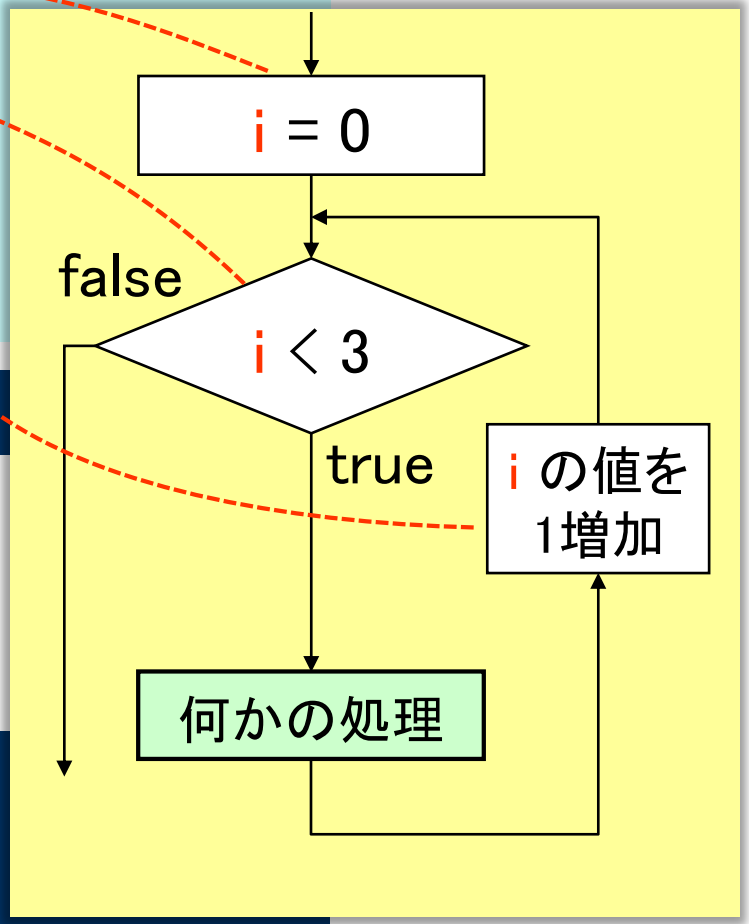
for文のパターン（ある回数だけ文を実行する場合）

for文のセミコロンは3パートの区切り
（単文の意味ではない）

キーワード　セミコロン　カッコが必要

```
int i;
for (i = 0; i < 3; i++)
{
    printf("A");
}
```

この中が3回実行される



AAA

ループ変数 i = 0 でスタートして
i < 3 の間は反復 ⇒ i = 0, 1, 2

⇨ 何かの処理は3回実行される

```
int a;
for (a = 0; a < 5; a++)
{
    printf("B¥n");
}
```

```
B
B
B
B
B
```

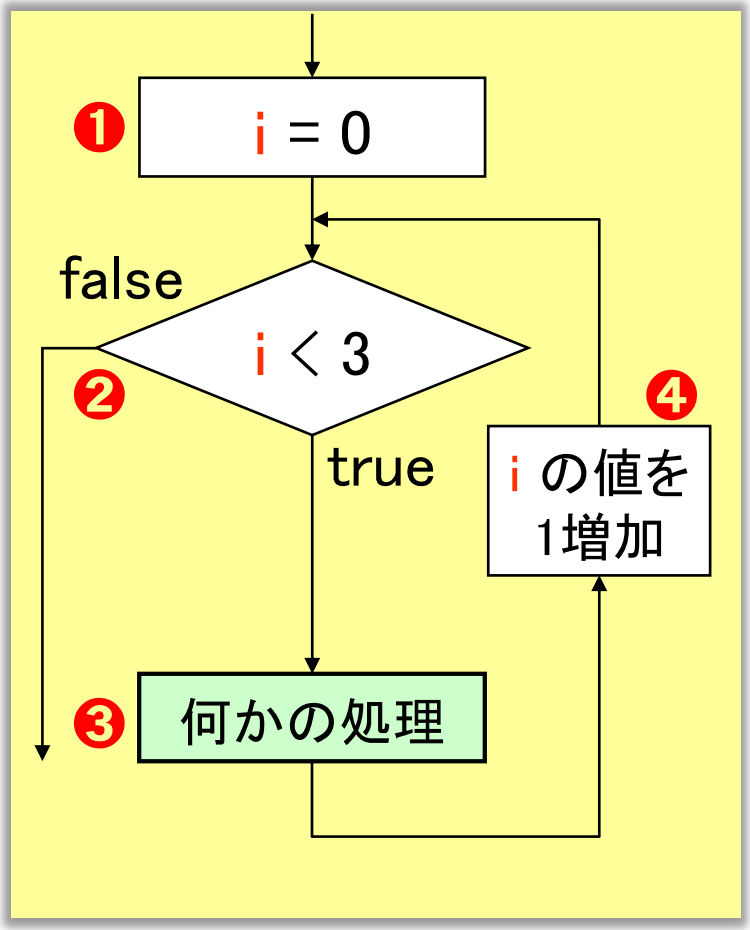
ソース中の実行順のイメージ

```
int i;
for ( i = 0 ; i < 3 ; i++ )
{
    printf("A");
}
```

❶ ❷ ❸ ❹

False

True



AAA

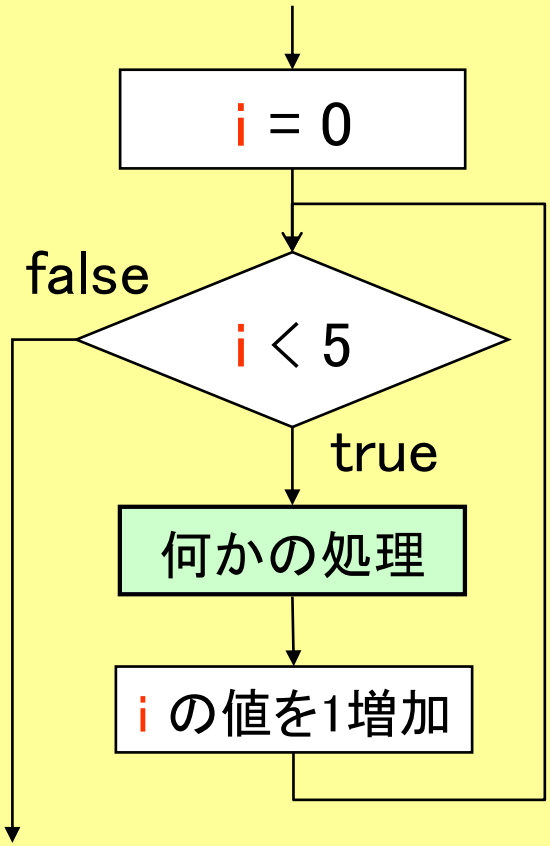
ループ変数(1)

```
int i;
for (i = 0; i < 5; i++)
{
    printf("%d¥n", i);
}
```

```
0
1
2
3
4
```

ループ変数iの値を表示

ループ変数の値は1づつ増加




```
int i;
for (i = 2; i < 5; i++)
{
    printf("%d¥n", i);
}
```

```
2
3
4
```

```
int i;
for (i = 3; i < 7; i++)
{
    printf("%d¥n", i);
}
```

```
3
4
5
6
```

初期値

繰り返しの条件
(反復条件)

ループ変数(2)

```
int i;
for (i = 3; i <= 7; i++)
{
    printf("%d¥n", i);
}
```

```
3
4
5
6
7
```

ループ変数の
初期値は3

ループ変数が7より
小さいか等しい間は
繰り返す！

ちょっと応用問題
次のプログラムの実行結果はどうなる？

```
int k;
for (k = 3; k <= 7; k++)
{
    printf("%d¥n", 2*k);
}
```

```
6
8
10
12
14
```

ループ変数(3)

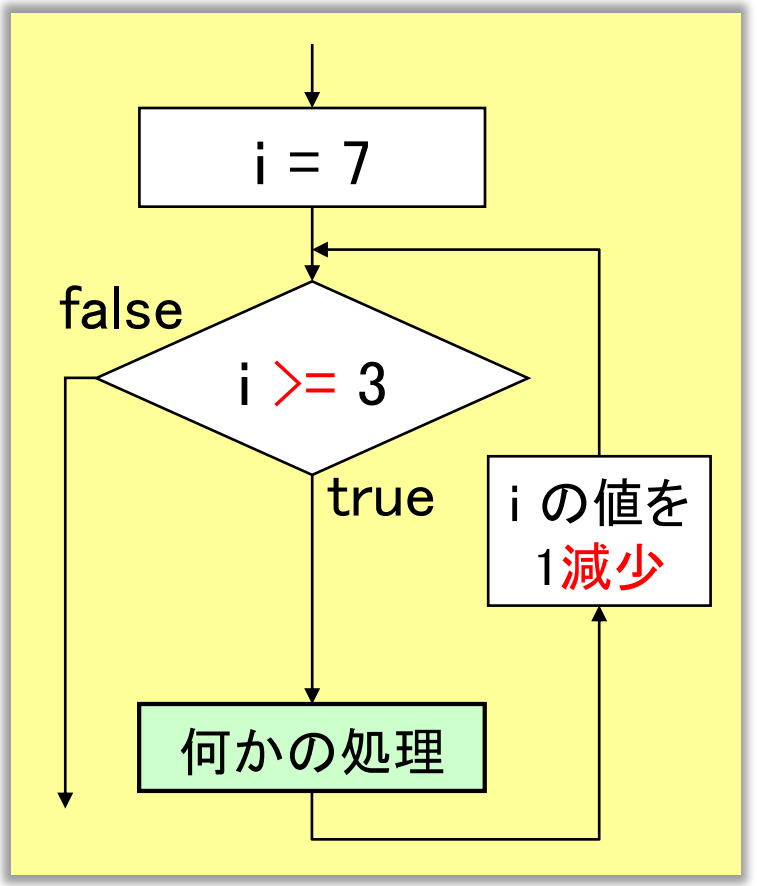
ループ変数の値が減っていく反復はどう書く？

```
int i;
for (i = 7; i >= 3; i--)
{
    printf("%d¥n", i);
}
```

```
7
6
5
4
3
```

ループ変数の
初期値は7

ループ変数が3より大きいか
等しい間は繰り返す！



a++ → 変数aの値を1増やす
a-- → 変数aの値を1減らす

ループ変数(4)

i=i+1は変数iの値を1増やす．i++と同じ．

```
int i;
for (i = 3; i <= 7; i = i + 1)
{
    printf("%d¥n", i);
}
```

```
3
4
5
6
7
```

ちょっと応用問題
次のプログラムの実行結果はどうなる？

```
int i;
for (i = 3; i <= 10; i = i + 2)
{
    printf("%d¥n", i);
}
```

```
3
5
7
9
```

ループ変数のステップ値は2

i=i+2は変数iの値を2増やす