for文の文法

復習

for文の文法のまとめ

これ自体も一つの文！

```
for （初期値設定 ； 反復条件 ； ステップ値設定）
   文
```

文 → 単文 or 複文

復習

単文

```
printf("負の数です¥n");
```

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

複文

複数の文を
ひとまとめ

```
{
  b = 10;          単文
  a = 2*b;         単文
  if (a > 5)                    if文
  {
    printf("aは%dである¥n",a);
  }
}
```

if文も一つの文

問題:このプログラムは誤り？

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

ループで実行する文が
一つならこれでもOK

よくある間違い

復習

問題:この実行結果は？

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

こうなるはずでは？

```
3
4
5
6
7
続行するには . . .
```

違います！

実際にはこうなる！

```
8
続行するには . . .
```

この空文をi=3～7まで繰り返し実行！

わかりやすくソースを書き直すと

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

i=8の時，ループを終了してこの文を一度だけ実行！

単文

```
printf("負の数です¥n");
a = 2*a;
;
```

セミコロンだけでも一つの文！（空文）

復習

ループ変数の利用(1)

1から5の整数値の和を求める　１＋２＋３＋４＋５＝１５

```
int a,sum = 0;
for (a = 1; a <= 5; a++)
{
    sum = sum + a;
    printf("a=%d  sum=%d¥n", a, sum);
}
printf("和は%d¥n", sum);
```

この例では，ループを始める前にsumが0であることが重要

大原則
初期化(代入)されていない変数の値は不定である.

- ゼロとは限らない
- どんな値が入っているか決まっていない
- 実行するたびに異なった値になることも

代入前のsumの値

aの値

足し算の結果

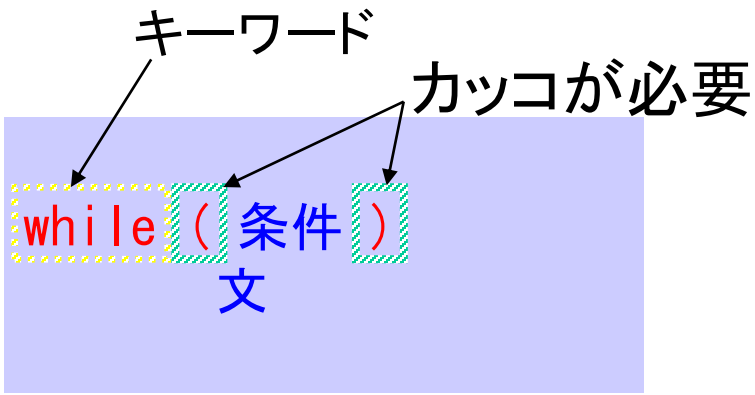
```
a=1   sum=1
a=2   sum=3
a=3   sum=6
a=4   sum=10
a=5   sum=15
和は15
```

```
sum ← 0                  + 1
sum ← 1 (=0+1)           + 2
sum ← 3 (=0+1+2)         + 3
sum ← 6 (=0+1+2+3)       + 4
sum ← 10 (=0+1+2+3+4)    + 5
```

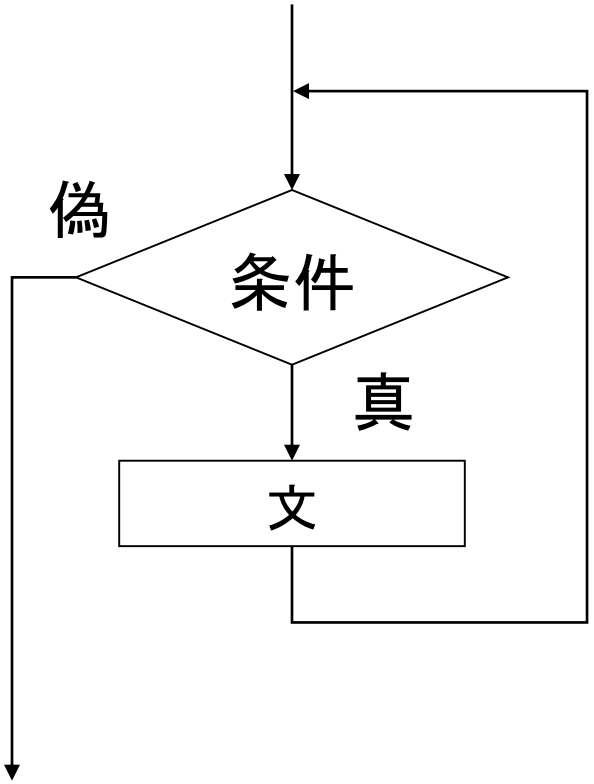
sum=(0+1+2+3+4)+5

復習

while文(1)



・ 条件が真の間ずっと文(複文も可)を反復する
・ ループ変数は必要ない



復習

while文(2)

ユーザーが0を入力するまで繰り返す

【注意】初めからaが0だと
一度も実行されない

```
int a = 1;
while ( a != 0 )
{
    printf("正の整数を入力してください: ");
    scanf("%d", &a);
}
printf("終了します");
```

条件

```
int a = 0;
```

➡ 一度も実行されない

```
int a;
```

➡ 実行するたびに異なった結果！

条件が真(true)なら
この中が繰り返し
実行される

```
正の整数を入力してください:5
正の整数を入力してください:6
正の整数を入力してください:10
正の整数を入力してください:15
正の整数を入力してください:0
終了します
```

復習

デバッガの利用(1)

プログラムのミス
→ バグ

虫のこと

バグを見つけて修正すること
→ デバッグ

バグを発見する道具
→ デバッガ

どんな優秀なプログラマーでも、初めからバグのない完全なプログラムは作れない！
⇒ プログラムを完成させることはデバッグを完了すること！

プログラミング ≠ プログラムを書く

プログラミング ≒ デバッグ

自分で考え抜いてデバッグすることが重要

復習

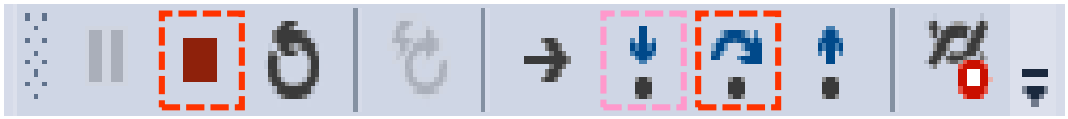
デバッガの利用(2)

大原則
バグがあったら，デバッガを起動すること！

[A] ツールバーでデバッグを始める手順

(1) ツールボタンで右クリックして，デバッグを選ぶ

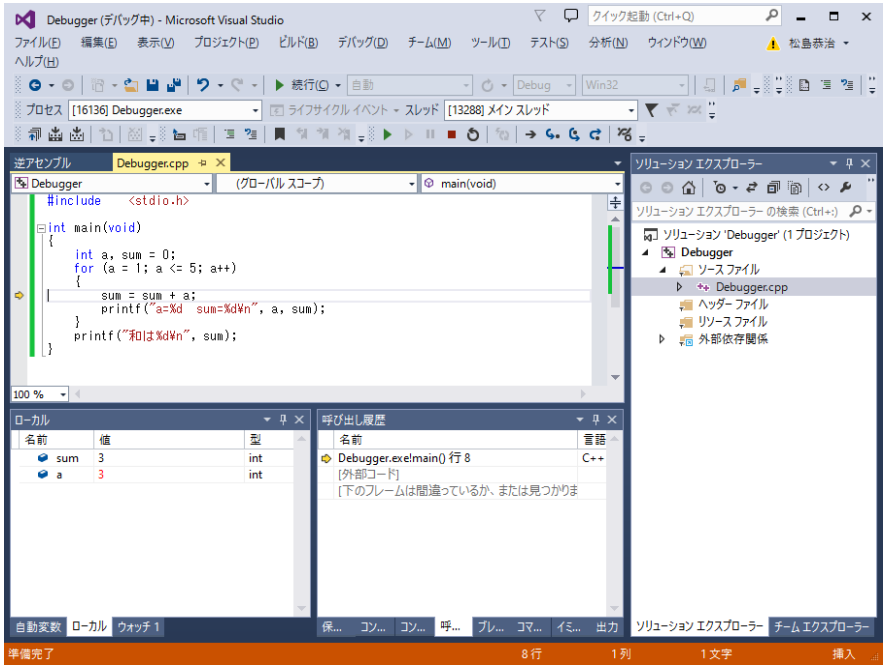
(2) デバッグツールバーからボタンを選ぶ



(3)　(2)　(1)

デバッグの中止

ステップオーバー



デバッガを試さずにTAさんに相談することは禁止！

意図的な無限ループ

プログラムミスによる無限ループの例

```
while ( 3 >= 2 )
{
        printf("A¥n");
}
```

```
while (1)
{
        printf("A¥n");
}
```

Cで意図的に無限ループを作る書き方

```
for( ; ; )
{
        printf("A¥n");
}
```

初期値も条件も何も書かない
この書き方がOK！

```
while (1)
{
        printf("A¥n");
}
```

条件が常に真(1)

C言語では0以外の定数値は真を意味する

無限ループの利用とbreak文による脱出

0が入力されるまで2乗を計算するプログラム

```
float x = 1;
while(x != 0)
{
  printf("x = ?");
  scanf("%f", &x);
  printf("%fの２乗は%f¥n", x, x*x);
}
```

```
x = ? 3
3.000000の２乗は9.000000
x = ? 2.5
2.500000の２乗は6.250000
x = ? 0
0.000000の２乗は0.000000
続行するには何かキーを押して . . .
```

この行を出さないためには？

無限ループ

```
float x;
while(1)
{
  printf("x = ?");
  scanf("%f", &x);
  if (x == 0.0)
    break;
  printf("%fの２乗は%f¥n", x, x*x);
}
```

if文を挿入

```
x = ? 3
3.000000の２乗は9.000000
x = ? 2.5
2.500000の２乗は6.250000
x = ? 0
続行するには何かキーを押して . . .
```

break文を実行すると強制的にループを終了する

Cにおけるループからの脱出と制御

これらは，while文でもfor文でも，反復処理であれば常に使用可

break

ループを強制終了してループから脱出する．if文と組み合わせて利用するのが一般的．

continue

現在のループの残りの処理をスキップし，そのループの次の周回を始める．if文と組み合わせて利用するのが一般的．

goto

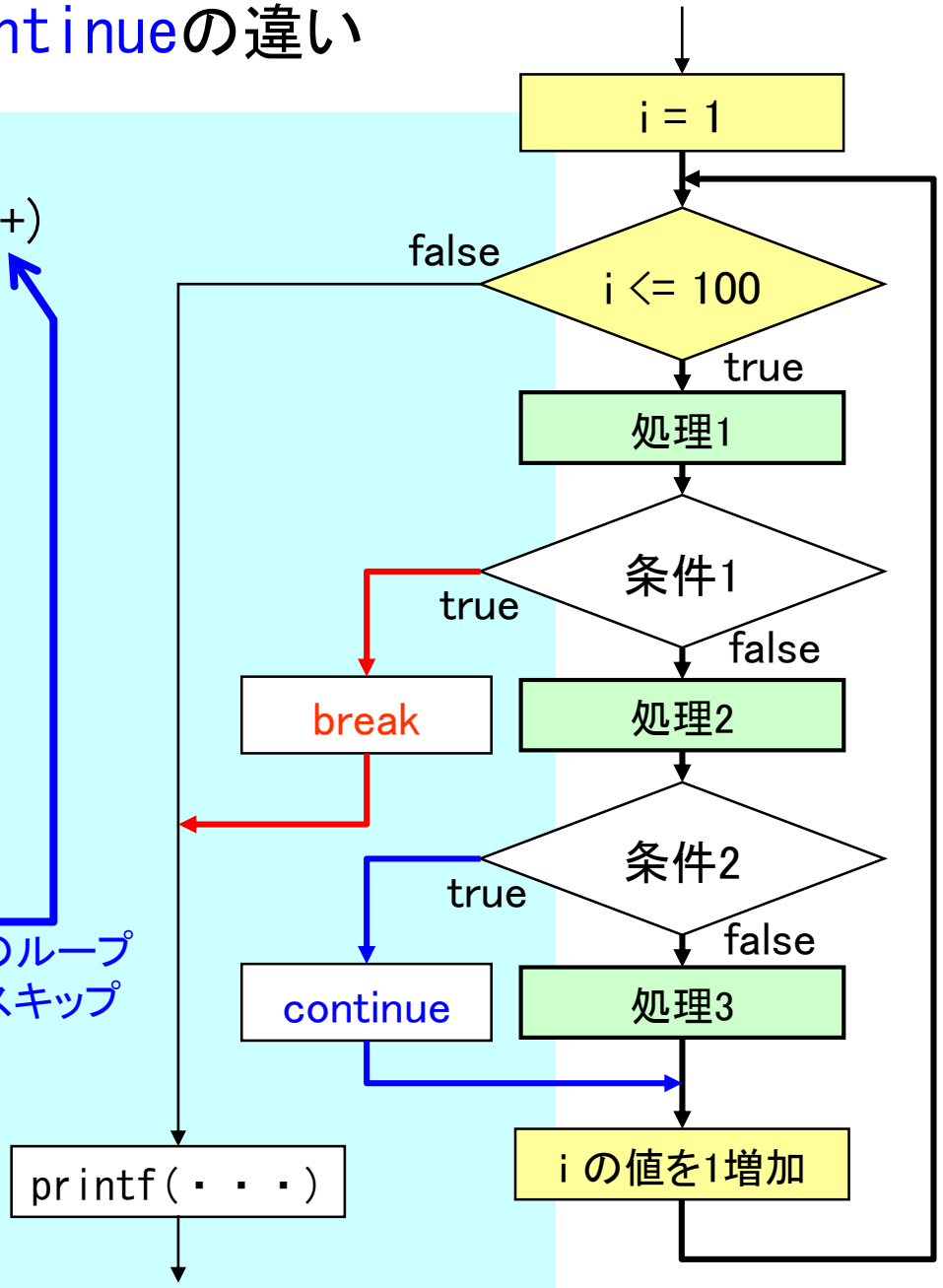
任意の場所へジャンプし，そこから文を実行する．C言語においては使用しないことが望ましい(従って説明略)．

breakとcontinueの違い

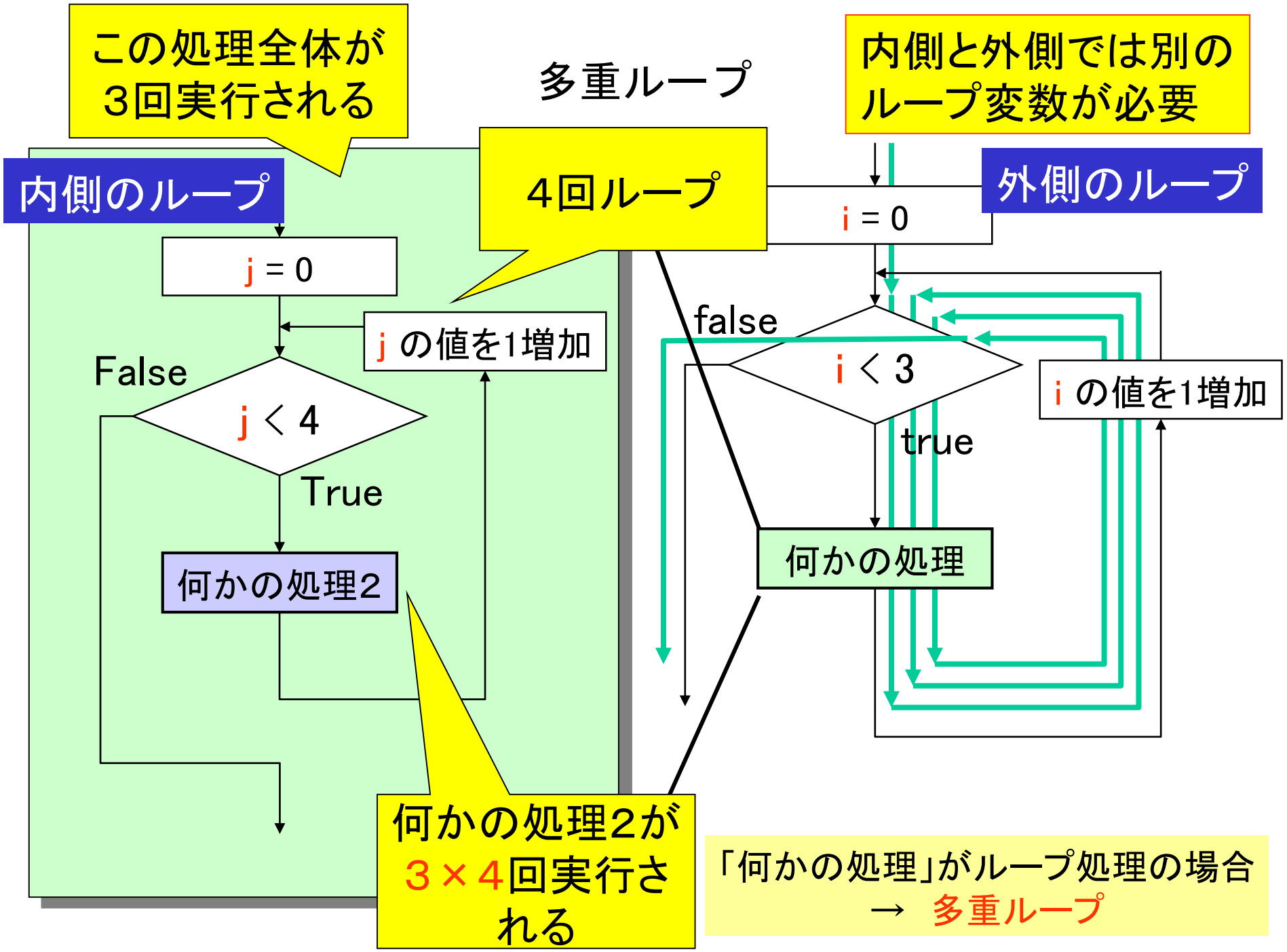
```
int i;
for (i = 1; i <= 100; i++)
{
    ・・・処理１・・・・

    if (条件1)
        break;

    ・・・処理２・・・・

    if (条件2)
        continue;

    ・・・
    ・・・処理３・・・・
}
printf(・・・);
```

ループから飛び出す

これ以下のループ内処理をスキップ



多重ループ


この処理全体が
3回実行される
内側と外側では別の
ループ変数が必要
内側のループ
外側のループ
4回ループ
i = 0
j = 0
false
j の値を1増加
i ＜ 3
i の値を1増加
False
j ＜ 4
true
True
何かの処理
何かの処理2
何かの処理2が
3×4回実行さ
れる
「何かの処理」がループ処理の場合
→ 多重ループ

2重のループ(その1)

外側のループ
ここがi=0からi=4まで
5回反復される

```
int i, j;
for (i = 0; i < 5; i++)
{
    printf("%d: ", i);
    for (j = 0; j <= 9; j++)
    {
        printf("%d,", j);
    }
    printf("¥n");
}
```

内側のループ
ここがj=0からj=9まで
反復される

外側のループ

i = 0
i = 1
i = 2
i = 3
i = 4

j = 0, 1, 2, ・・・

内側のループ

```
0: 0,1,2,3,4,5,6,7,8,9,
1: 0,1,2,3,4,5,6,7,8,9,
2: 0,1,2,3,4,5,6,7,8,9,
3: 0,1,2,3,4,5,6,7,8,9,
4: 0,1,2,3,4,5,6,7,8,9,
```

ループのネスト

2重のループ(その2)

＊印で三角形を描く

```
int i, j;
for (i = 1; i <= 5; i++)
{
    for (j = 1; j <= i; j++)
    {
        printf("*");
    }
    printf("¥n");
}
```

外側のループ
ここがi=1からi=5まで
5回反復される

内側のループ
ここがj=1からj=iまで
i回反復される

外側のループ

i = 1
i = 2
i = 3
i = 4
i = 5

```
*       j = 1
**      j = 1, 2
***     j = 1, 2, 3
****    j = 1, ～, 4
*****   j = 1, ～, 5
```

内側のループ