

プログラミング入門 第9回

for文をネストする (p.164-165)

- ◆ 構造化プログラミングにおける**ネスティング**, **ネスト**, **入れ子**とは, プログラムの構造が再帰的に繰り返されて記述されることである. このような構造を**ネスト構造**, **入れ子構造**と呼ぶ.
- ◆ ロシアの民芸品マトリョーシカは, 人形の中に人形が入っており, 入れ子構造になっている.
- ◆ for 文の中に, 再度 for 文を書くと, ネスト構造になる.



for文をネストする (p.164-165)

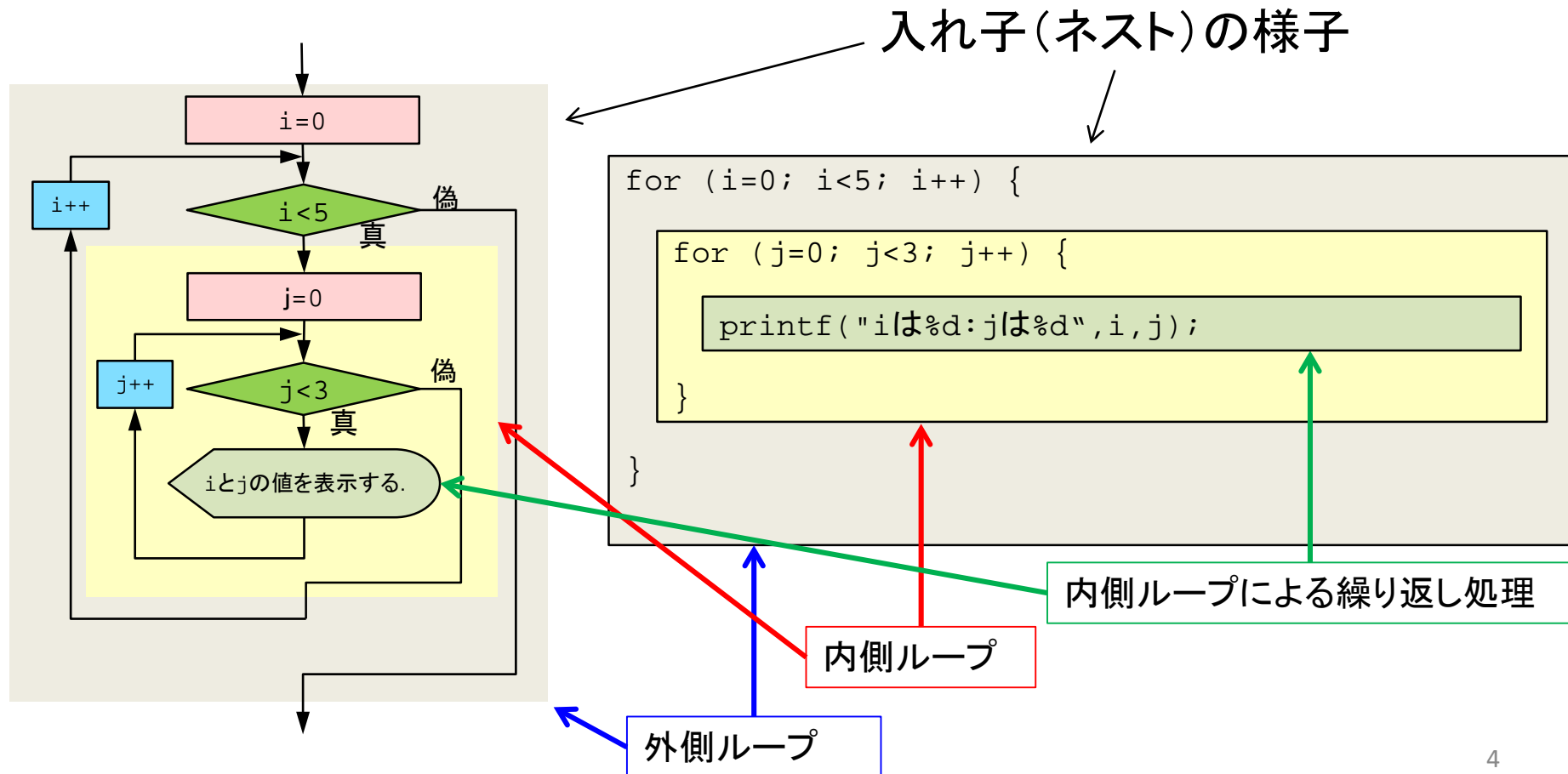
- ◆ for 文をネストする（入れ子にする）.
- ◆ for 文の中で for 文を使う.

```
for (初期化式1; 条件式1; 更新式1) {  
    ...  
    for (初期化式2; 条件式2; 更新式2) {  
        ...  
    }  
}
```

- ◆ for文による多重ループが形成される.

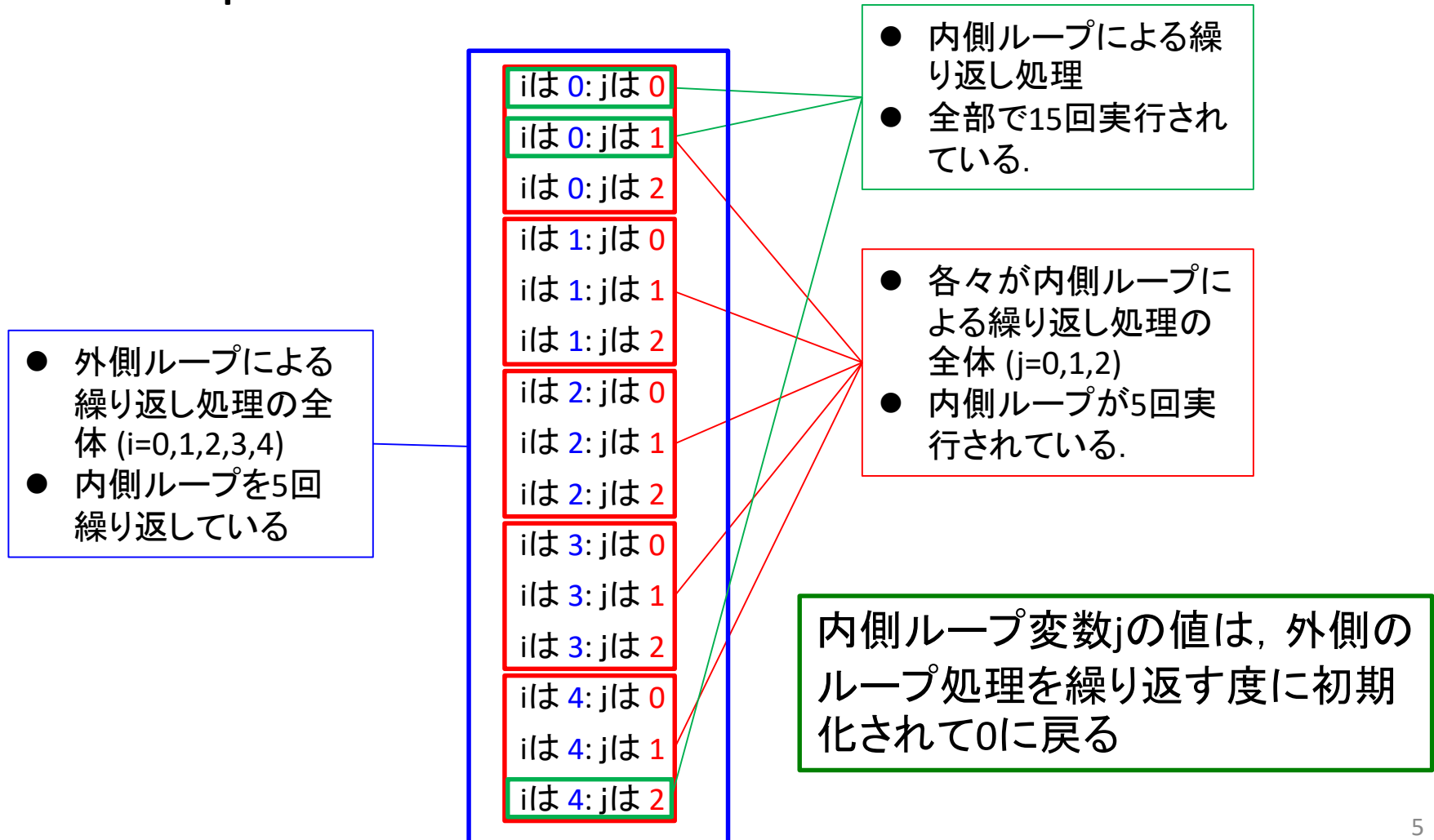
for文をネストする (p.164-165)

◆Sample8.cのフローチャート



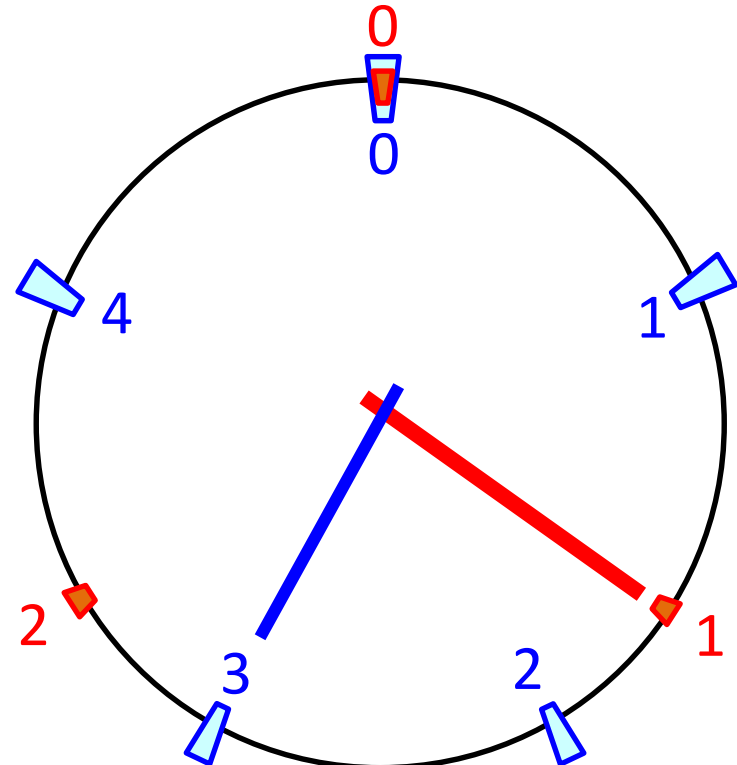
for文をネストする (p.164-165)

◆Sample8.cの実行結果



for文をネストする (p.164-165)

◆ for文の**変数j**(内側ループ)、**変数i**(外側ループ)の値が変化する様子を時計の**長針**と**短針**に例えてみる



長針が一周すると**短針**が1つ分動く

if文などと組み合わせる(p.166-167)

◆ Sample9.c の構造

```
int i, j, ch;
```

```
ch = 0;
```

chの値を0で初期化しておく

```
for (i=0; i<5; i++) {
```

```
    for (j=0; j<5; j++) {
```

```
        if (ch == 0) {
```

```
            printf("*");
```

```
            ch = 1;
```

```
        } else {
```

```
            printf("-");
```

```
            ch = 0;
```

```
        }
```

```
    printf("\n");
```

```
}
```

```
}
```

chの値が0のとき, chの値を1に変更する

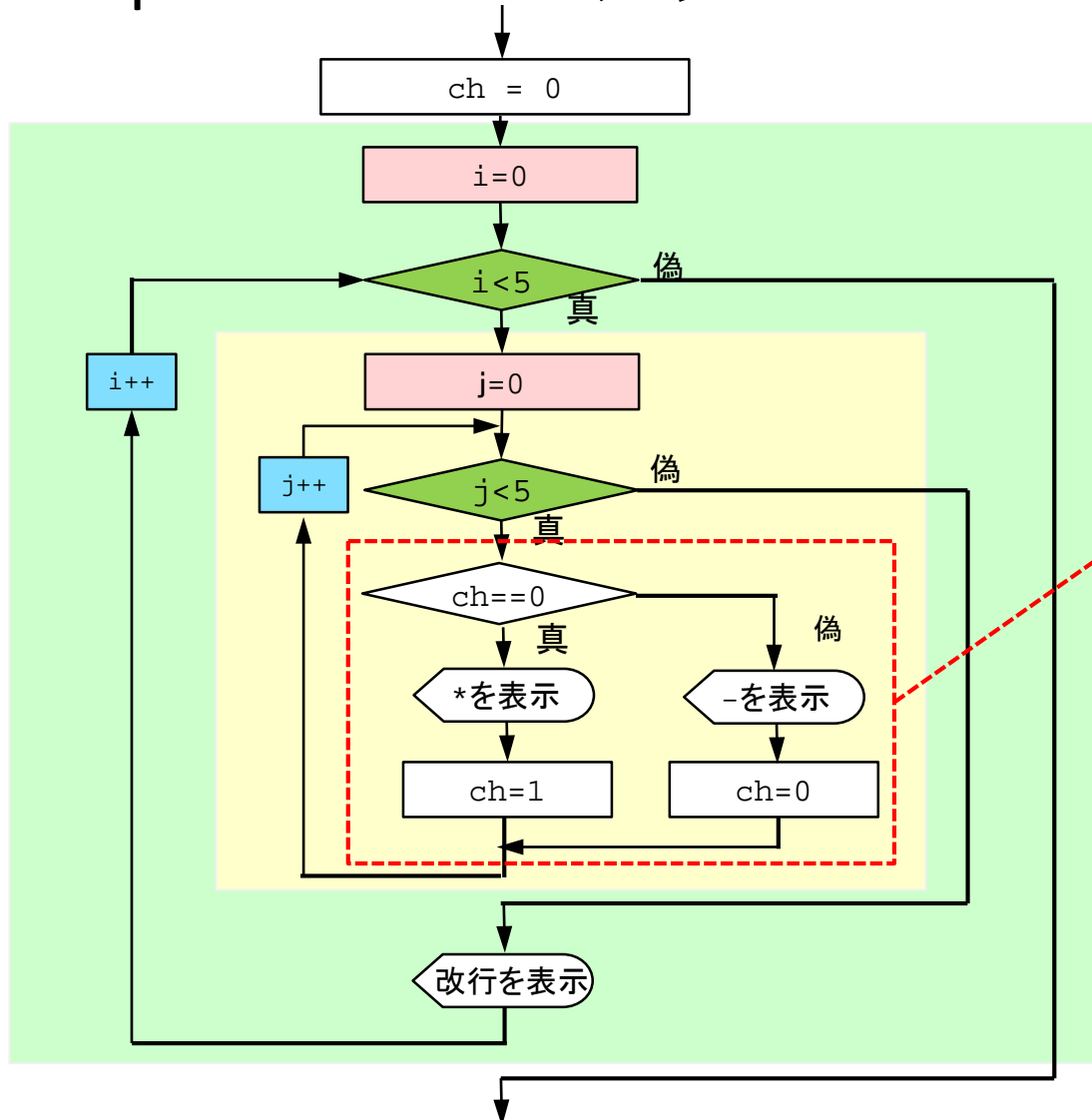
chの値が1のとき, chの値を0に変更する



内側のループを回るときに ch の値は, 0 と 1 の値を交互に入れ替わる。ただし, chの値は 0 から始まる。

if文などと組み合わせる(p.166-167)

◆ Sample9.cのフローチャート



for文のネスト内に
if文のブロックがある

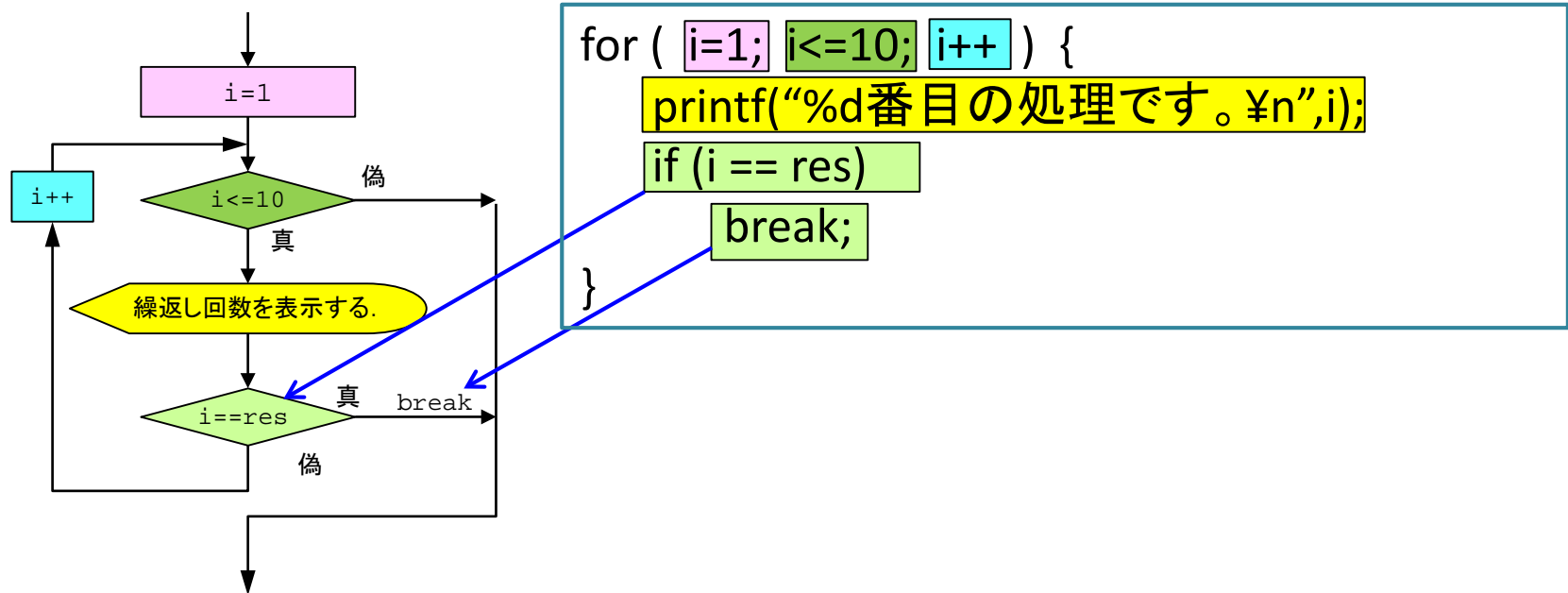
if文の条件判断

break文のしくみを知る(p.168-169)

◆break文

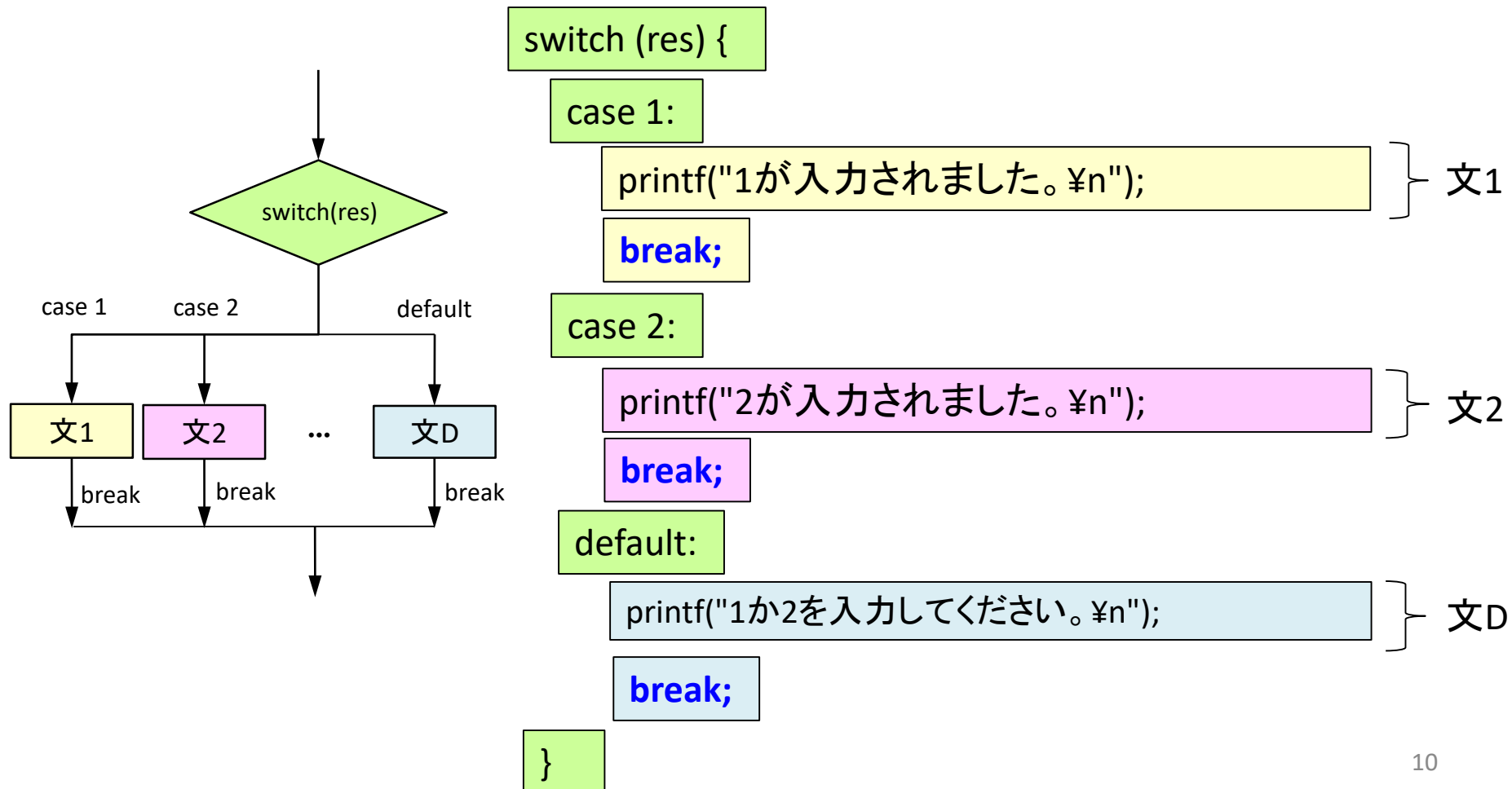
- 処理の流れを強制的に終了し, そのブロックから抜ける.

◆Sample10.cのフローチャート



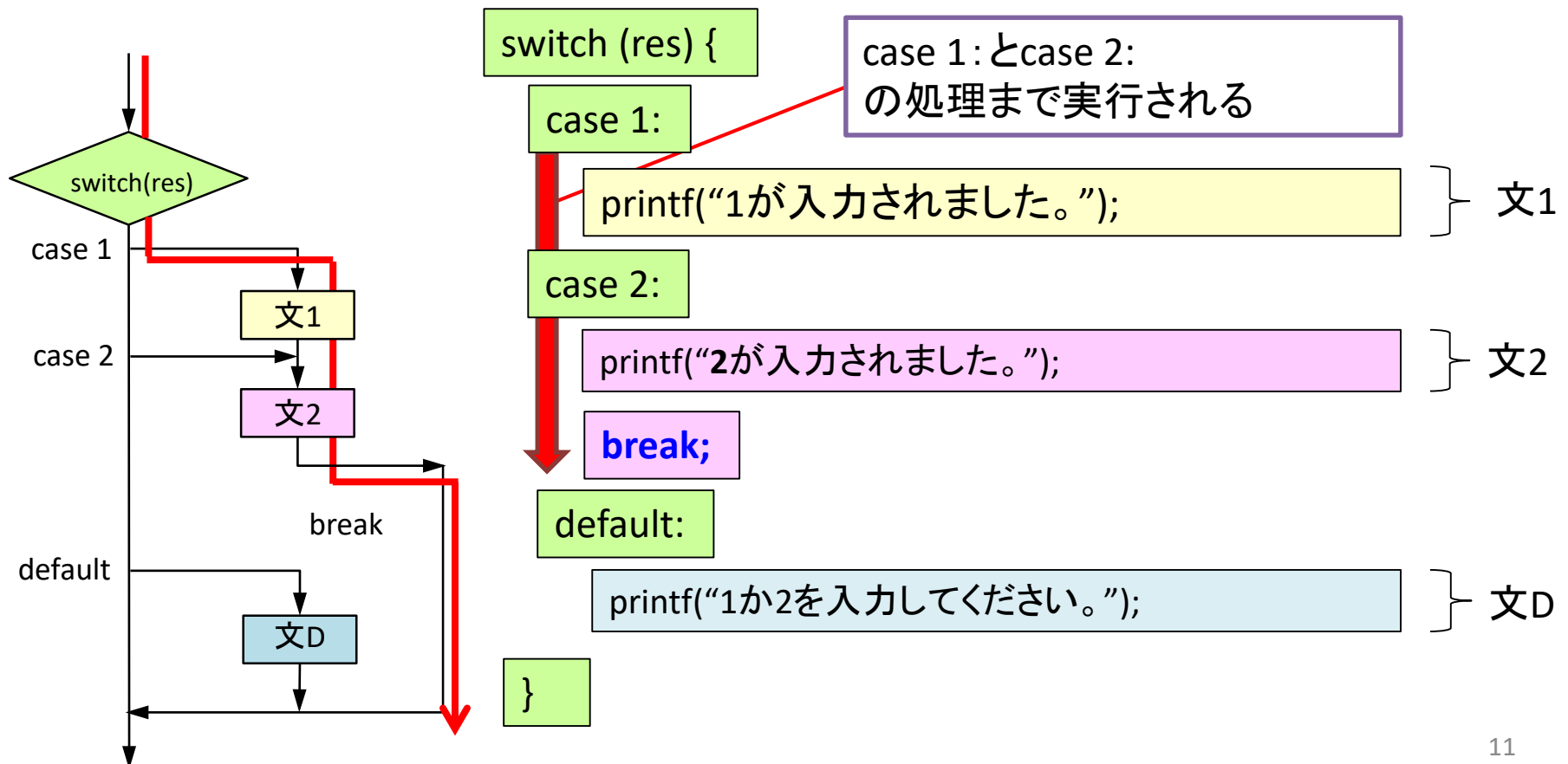
switch文の中でbreak文を使う

◆すべての case の処理に break 文がある場合



switch文の中でbreak文を使う

◆ case の処理に break文が抜けている箇所がある場合



switch文の中でbreak文を使う (p.170-171)

◆ break文の挿入位置によって処理を制御

- 複数のcaseの処理が共通である場合は あえてbreak 文を付けないことでまとめることができる

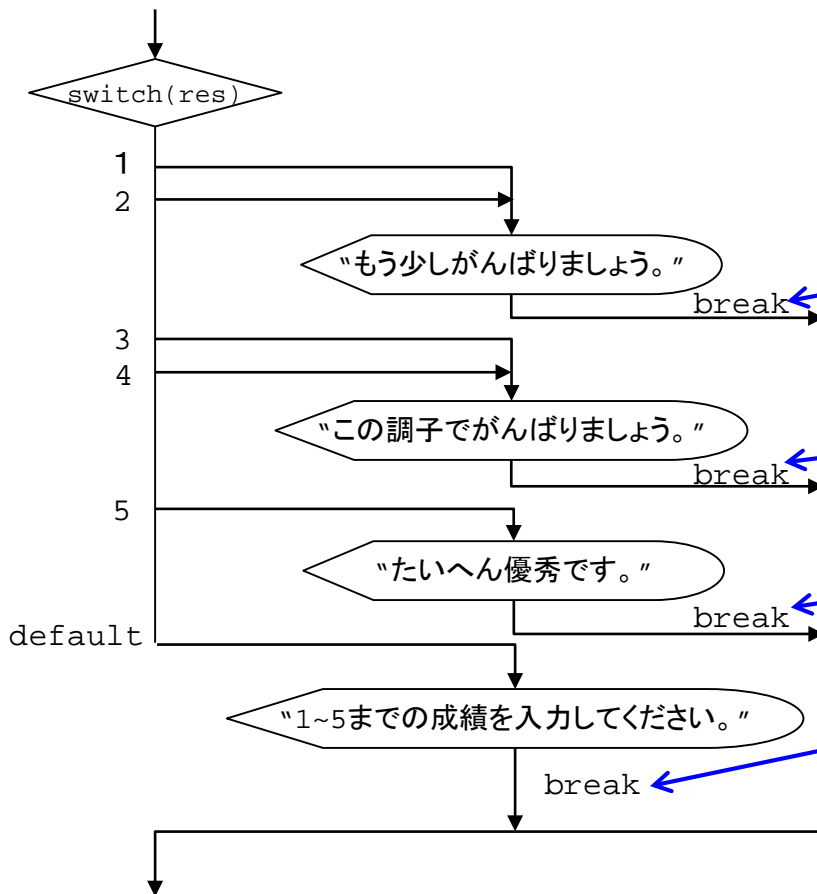
```
switch (res) {  
    case 1:  
    case 4:  
        printf("1か4を入力しました。¥n");  
        break;  
    case 2:  
    case 3:  
    case 5:  
        printf("2か3か5を入力しました。¥n");  
        break;  
    ....  
}
```

resの値が1か4
のときの処理

resの値が2か3か
5のときの処理

switch文の中でbreak文を使う (p.170-171)

◆Sameple11.cのフローチャート



```

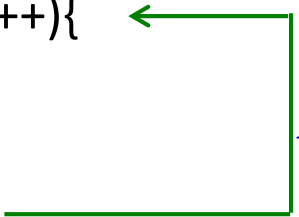
switch(res) {
  case 1:
  case 2:
    printf("もう少しがんばりましょう。¥n");
    break;
  case 3:
  case 4:
    printf("この調子でがんばりましょう。¥n");
    break;
  case 5:
    printf("たいへん優秀です。¥n");
    break;
  default:
    printf("1~5までの成績を入力してください。¥n");
    break;
}
  
```

continue文のしくみを知る(p.172-173)

◆continue文

- 繰り返し文内の処理を飛ばしブロックの先頭位置に戻って処理を続ける

```
for(i=1; i<=10; i++){  
    if(i == res)  
        continue;  
    printf("%d番目の処理です。¥n", i);  
}
```



iの値がresの値と等しいとき
for文のブロック先頭位置に
強制的に戻る

while文による無限ループ

◆ while文の条件を「1」とする常に条件の真であるため無限ループとなる

- if文とbreak文を組わせて無限ループから抜け出さるにする

