プログラミング入門
第8回

KITAMI
Institute of Technology

繰返し処理(1)

◆ 1から1000までの和を求めるプログラムの例

```
int sum = 0;
sum += 1;
sum += 2;
sum += 3;
sum += 4;
sum += 5;
sum += 6;
sum += 7;
sum += 8;
sum += 9;
sum += 10;

. . .

sum += 1000;
```

1000行書くのは大変

for文による繰り返し処理

```
int i;
int sum = 0;
for (i=1; i<=1000; i++) {
        sum += i;
}
```

for文ブロック内の処理を1000回繰り返す

◆ 繰返し処理を利用するとプログラムを簡潔に記述できる

繰返し処理(2)

◆繰返し処理を行うには，次のいずれかの文を利用する.

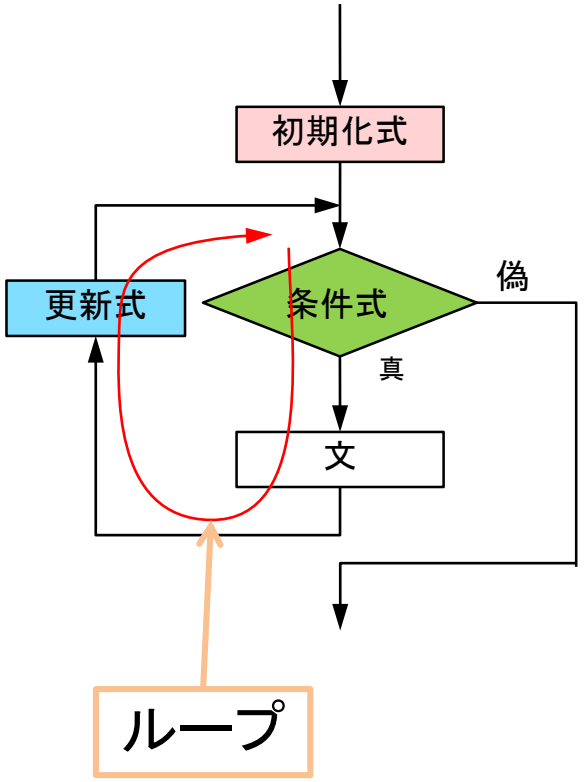
- for 文
- while 文
- do-while 文

for 文(p.150-151)

```
for (初期化式; 条件式; 更新式)
        文;
```

for 文では，繰返しを制御するために**ループ変数**と呼ばれる変数を利用する．

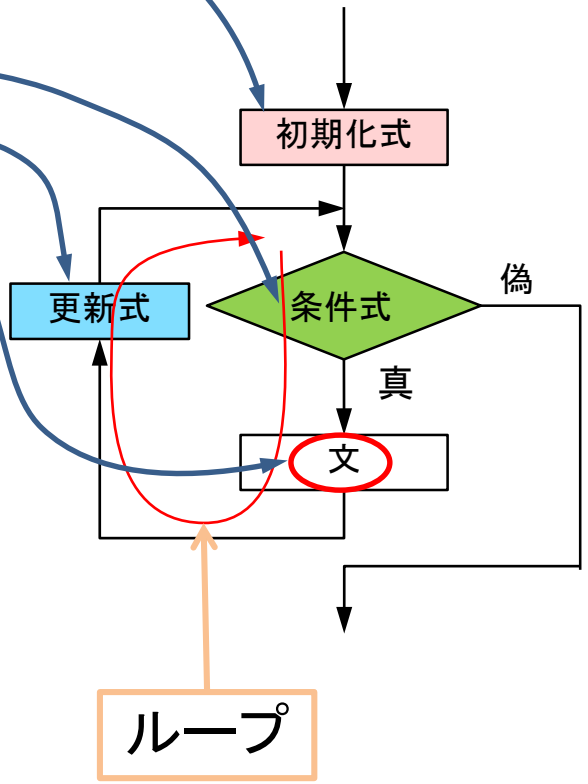
- **初期化式**で，ループ変数を初期化する．
- **条件式**で，繰返しの継続を判断する．
- **更新式**で，ループ変数を更新する．



ループ変数は事前に変数の宣言をしておく

for 文(p.150-151)

```
for (初期化式; 条件式; 更新式)
        文;
```



for 文では，繰返しを制御するために**ループ変数**と呼ばれる変数を利用する．

- **初期化式**で，ループ変数を初期化する．
- **条件式**で，繰返しの継続を判断する．
- **更新式**で，ループ変数を更新する．

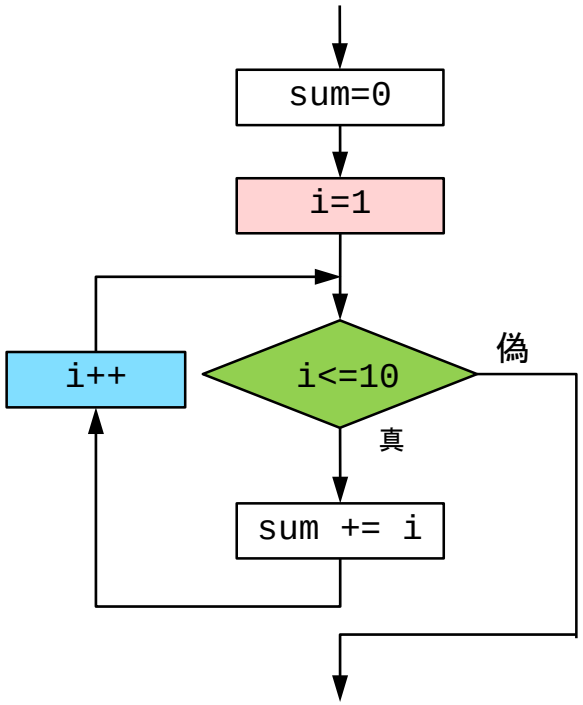
ループ変数は事前に変数の宣言をしておく

for 文

次の計算が，3行のプログラムで書ける．

```
int sum = 0;
sum += 1;
sum += 2;
sum += 3;
sum += 4;
sum += 5;
sum += 6;
sum += 7;
sum += 8;
sum += 9;
sum += 10;
```

```
int i;
int sum = 0;
for (i=1 ; i<=10 ; i++ ) sum += i ;
```



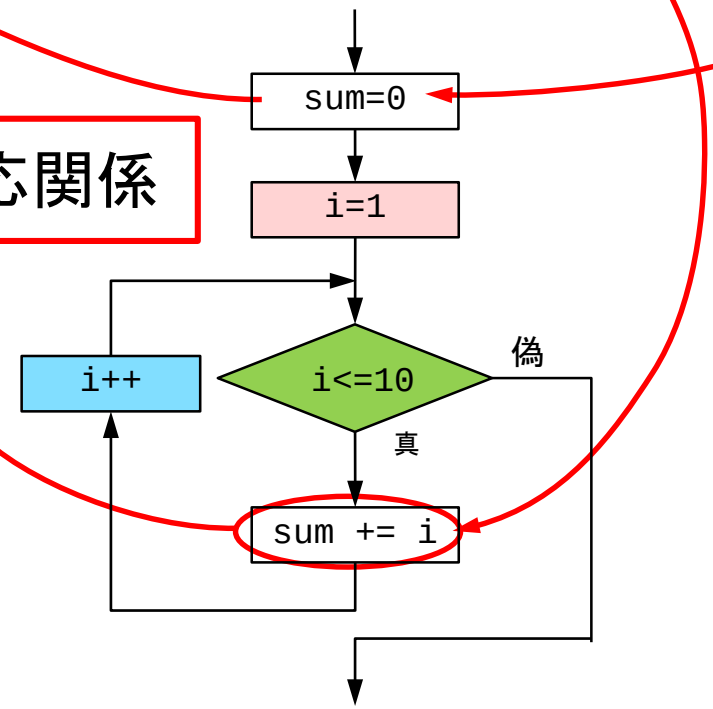
for 文

次の計算が，3行のプログラムで書ける．

```
int sum = 0;
sum += 1;
sum += 2;
sum += 3;
sum += 4;
sum += 5;
sum += 6;
sum += 7;
sum += 8;
sum += 9;
sum += 10;
```

```
int i;
int sum = 0;
for (i=1 ; i<=10 ; i++ ) sum += i;
```

対応関係



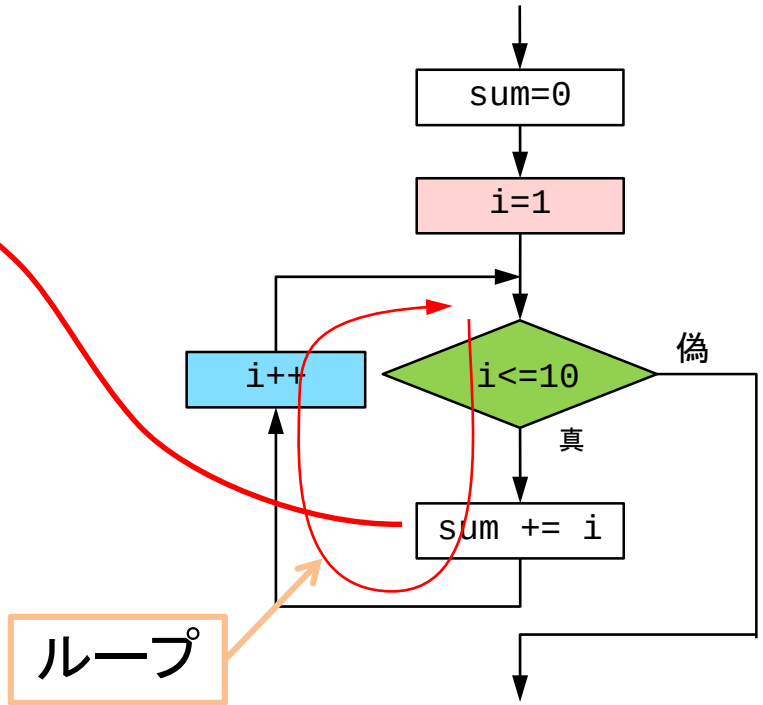
for 文

次の計算が，3行のプログラムで書ける．

```
int sum = 0;
sum += 1;
sum += 2;
sum += 3;
sum += 4;
sum += 5;
sum += 6;
sum += 7;
sum += 8;
sum += 9;
sum += 10;
```

ループ変数

```
int i;
int sum = 0;
for (i=1 ; i<=10 ; i++ ) sum += i ;
```



for 文

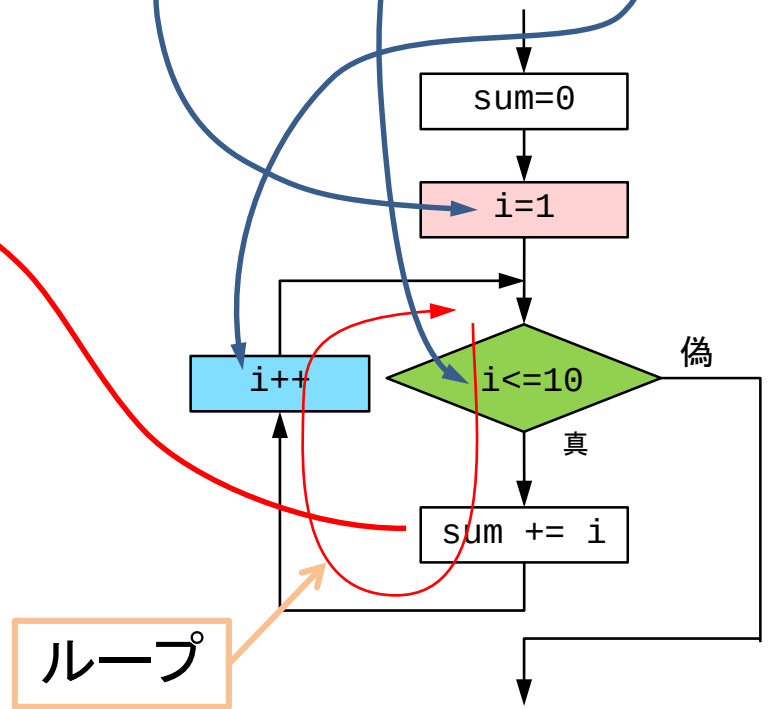
次の計算が，3行のプログラムで書ける．

```
int sum = 0;
sum += 1;
sum += 2;
sum += 3;
sum += 4;
sum += 5;
sum += 6;
sum += 7;
sum += 8;
sum += 9;
sum += 10;
```

ループ変数

```
int i;
int sum = 0;
for (i=1 ; i<=10 ; i++ ) sum += i ;
```

初期化式　条件式　更新式



ループ変数と
- 初期化式
- 条件式
- 更新式

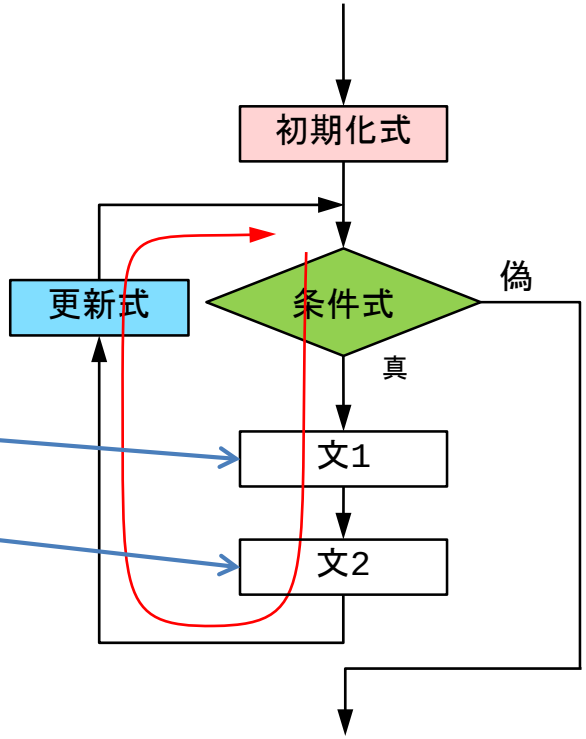
でループを制御

条件式を書きかえれば，繰返し回数を容易に変更できる．

for 文(p.150-151)

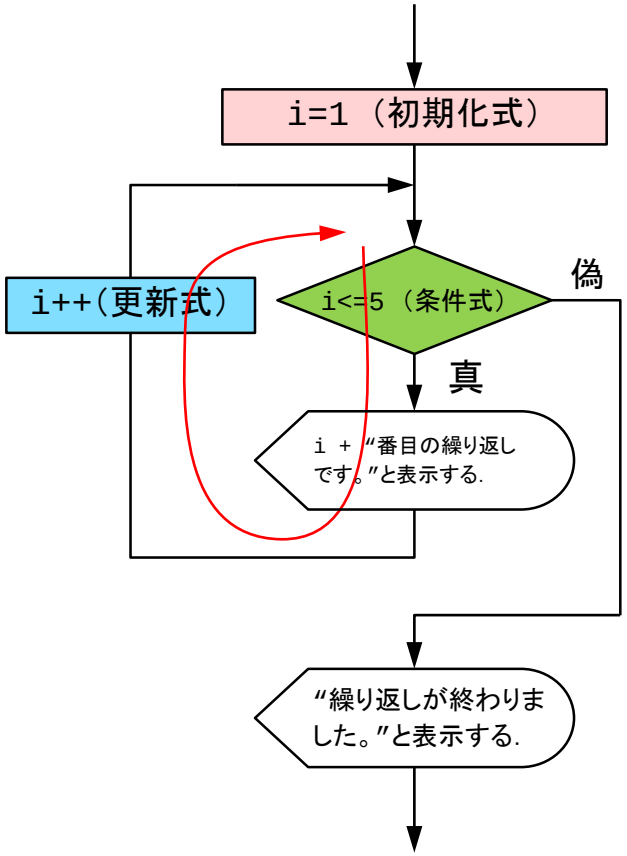
繰返し実行する文が複数ある場合には，ブロックを使う．

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



for文を使う(p.151-152)

◆Sample1.cのフローチャート

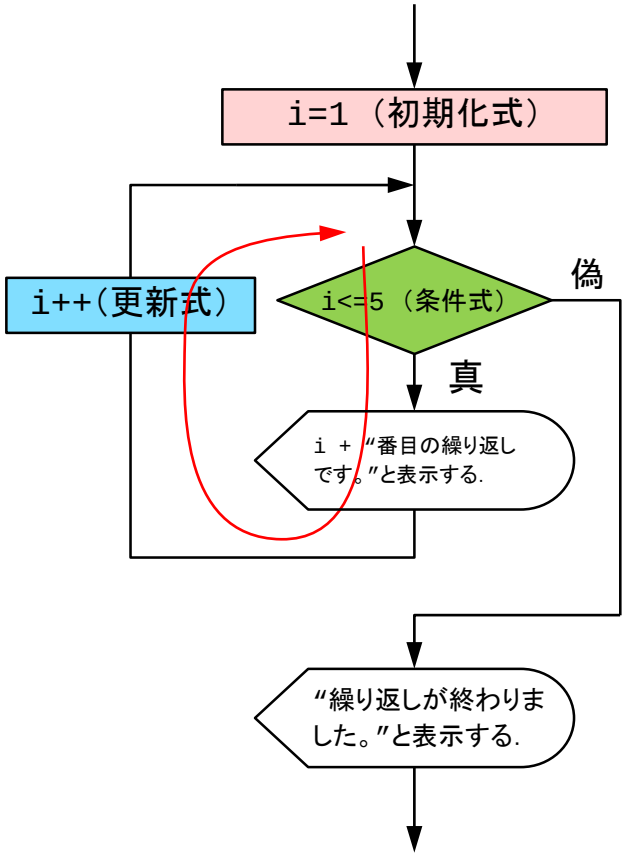



```
#include <stdio.h>

int main(void)

{

    int i;

    for(i=1; i<=5; i++){

        printf(“%繰り返しています。¥n");

    }

    printf("繰り返しが終わりました。¥n");

    return 0;

}
```

ブロック内の処理を
i=1, 2, 3, 4, 5と
5回繰り返す

変数をループ内で使う(p.153-154)

◆Sample2.cのフローチャート

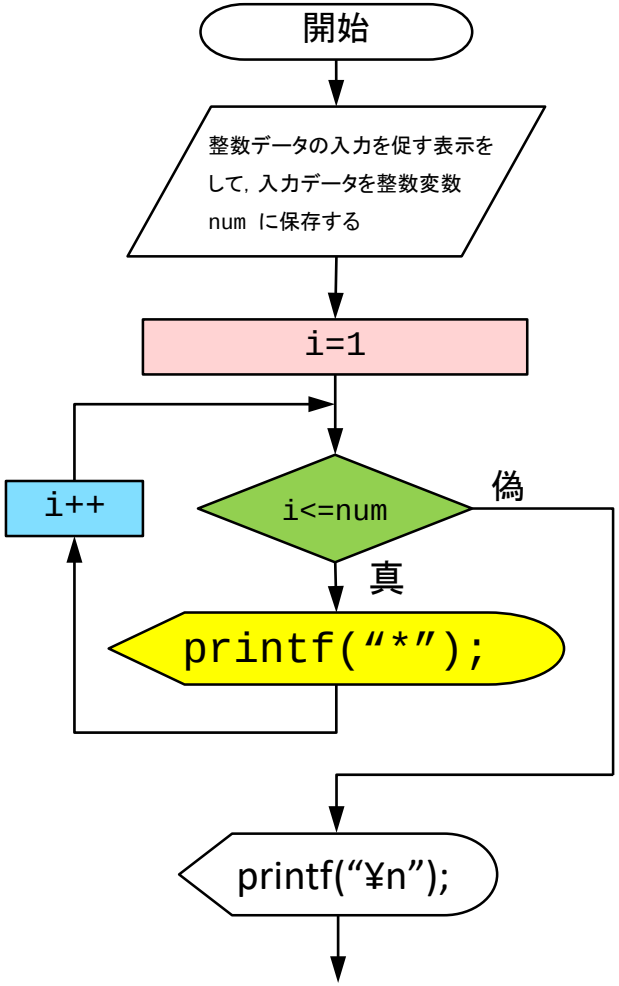



```
#include <stdio.h>

int main(void)

{

    int i;

    for(i=1; i<=5; i++){

        printf(“%d番目の繰り返しです。¥n", i);

    }

    printf("繰り返しが終わりました。¥n");

    return 0;

}
```

ループ変数iの値が
1, 2, 3, 4, 5と変化

for文を応用する(p.154)

◆Sample3.cのフローチャート



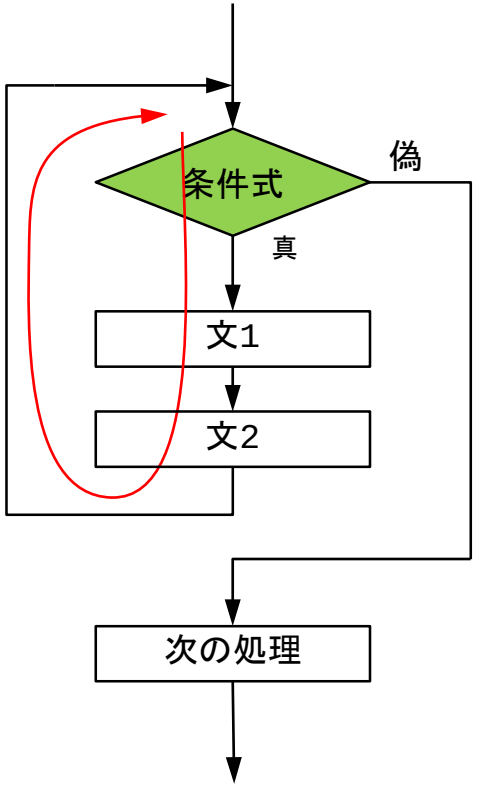
numの値 (整数)の応じた回数分だけ繰り返す

```
for ( i=1; i<=num; i++) {

      printf(“*”);
}
printf(“¥n”);
```

while文(p.157-158)

◆while文

- 条件式が 真 である限り，指定した文を何度でも繰り返す．

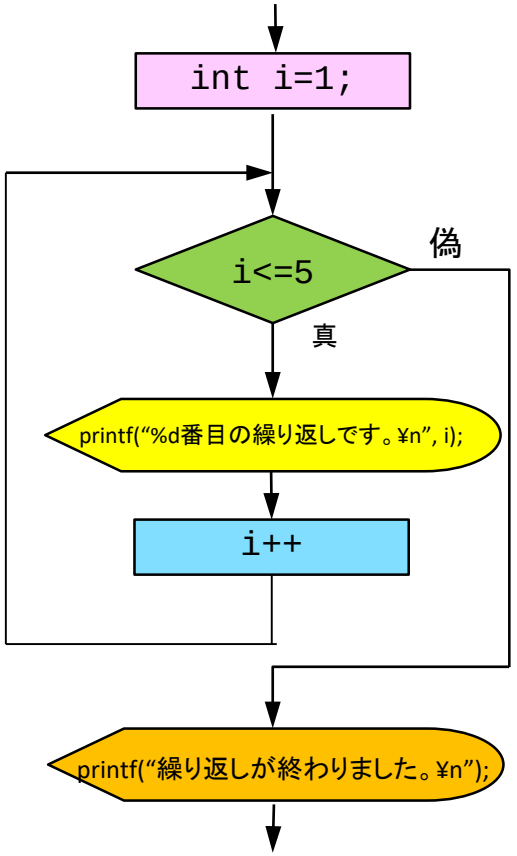



```
while (条件式)
{
        文1;
        文2;
        …
}
```

while文(p.157-158)

◆ Sample5.cのフローチャート

- 「i++;」の文により，条件が真から偽に変化するようにする．
- 条件が 真 のままだとプログラムが永久に終了しない（注）．




```
int i=1;
while ( i <= 5 ) {
    printf("%d番目の繰り返しです。¥n", i);
    i++;
}
printf("繰り返しが終わりました。¥n");
```

（注）break文により繰り返し処理のブロックから抜ける方法もある(pp. 168参照)

条件の記述を省略する(p.160-161)

◆C言語では整数値で条件式で評価することができる

- 0以外の整数値 → 真
- 0 → 偽

他のプログラミング言語では扱わない場合もあるので注意

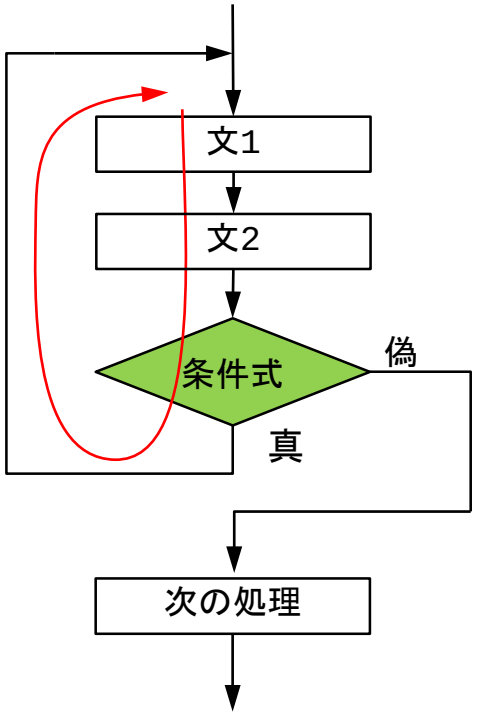
```
int num = 1;

    while(num){

        printf("整数を入力してください。(0で終了)¥n");

        scanf("%d", &num);

        printf("%dが入力されました。¥n", num);

    }

    printf("繰り返しが終わりました。¥n");
```

numの値が0のとき偽となり、
繰り返し処理内のブロックから抜け出す

do～while文(p.162-163)

◆do～while文

- 一度実行してからさらに繰り返すかどうかを判断する.
- 条件が真である限り，指定した文を何度でも繰り返す.

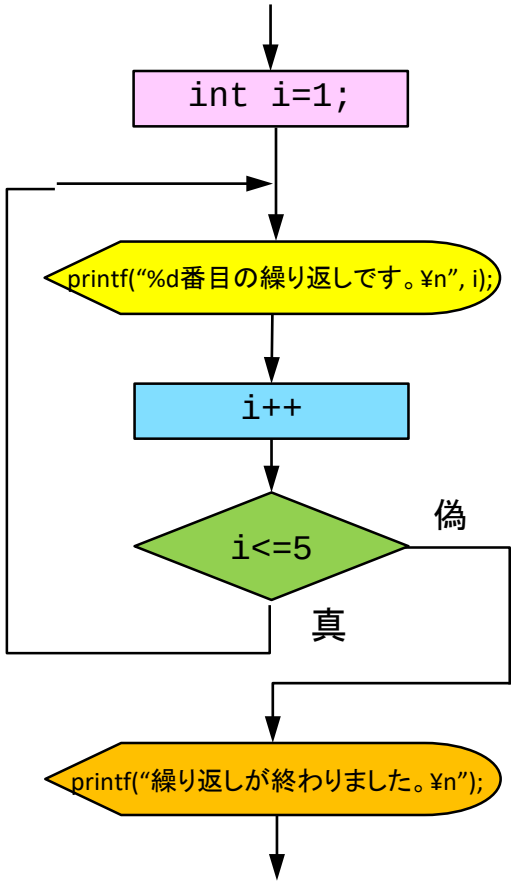



```
do {
        文1;
        文2;
        …
} while (条件式);
```

do～while文(p.162-163)

◆Sample7.cのフローチャート

● 繰り返しブロックの処理は最低でも1回実行される




```
int i=1;
do {
    printf("%d番目の繰り返しです。¥n", i);
    i++;
} while ( i <= 5 );
printf("繰り返しが終わりました。¥n");
```

do～while文(p.162-163)

◆while文とdo～while文の違い

- while文は繰り返し処理をはじめる前に条件を評価する.
- do～while文はブロック内の処理を実行した後に条件を評価する.

```
while (試験に合格していない){
        試験を受ける;
}
```

最初から試験に合格していれば,
試験を受けなくても良い

```
do {
        試験を受ける;
} while (成績判定が不可);
```

最低でも必ず1回は試験を受ける

補足

- while文やdo～while文のプログラム実行時に，繰り返し処理から抜け出させないことがある．これは次のことを疑う必要がある．
  - ソースコードの誤りのために，条件式の値が常に 真 となっている．
  - 「Ctrl」+「C」キーを押してプログラムを強制終了する．
- for文の「更新式」の例
  - 変数を1ずつ増やしていくには「i++」or「i+=1」or「i=i+1」
  - 変数を2ずつ増やしていくには「i+=2」or「i=i+2」
  - 変数を2ずつ減らしていくには「i-=2」or「i=i-2」