cp-5. 繰り返し計算

（C プログラミング入門）

URL: https://www.kkaneko.jp/pro/adp/index.html

金子邦彦

BY NC SA

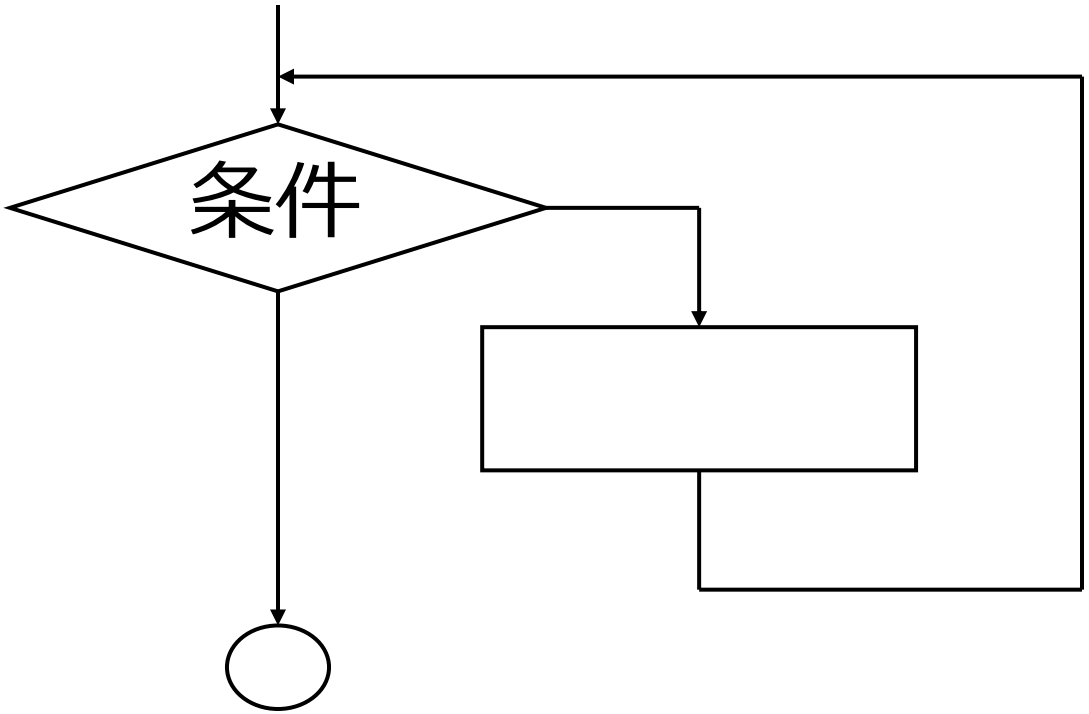
while文とfor文による繰り返し計算の実装、ループカウンタを用いた制御方法の習得

【学習内容の構成】

1. **while文**：条件が成り立つ限り処理を繰り返す構文
2. **for文**：初期式・条件式・再設定式による繰り返し制御
3. **繰り返しの入れ子**：for文を二重に使用した表形式データの処理
4. **応用例題**：ユークリッドの互除法、自然数の和、フィボナッチ数列の計算

- 前提：C言語の基本文法、変数宣言、条件式の理解
- 意義：アルゴリズムを繰り返し構造で表現する能力の獲得

繰り返しとは



- 繰り返しとは，ある条件が満たされるまで，同じことを繰り返すこと.
- 繰り返しを行うための文としてwhile文, for 文などがある.

繰り返しの例

- ユークリッドの互助法
  - m と n の最大公約数を求めるために，「割った余りを求めること」を，余りが 0 になるまで繰り返す.
- 九九の表
  - 九九の表を求めるために，掛け算を８１回繰り返す
- フィボナッチ数
  - フィボナッチ数を求めるために， f(n)=f(n-1)+f(n-2)を，n に達するまで繰り返す

など

例題１．最大公約数の計算

- ２つの整数データを読み込んで，最大公約数を求めるプログラムを作る.
  - ユークリッドの互助法を用いること
  - ユークリッドの互助法を行うために while 文を書く

例）　20, 12 のとき：　4

ユークリッドの互助法

- 最大公約数を求めるための手続き
- m,nの最大公約数は,
  - m ≧ n とすると,
  - 「m をn で割った余り」 = 0 なら，最大公約数は　n
  - 「m をn で割った余り」 ≠ 0 なら，m と n の最大公約数は，「m をn で割った余り」 と n の最大公約数に等しい （なお，n ＞ 「m をn で割った余り」 が成り立つ）

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int r;
    int m;
    int n;
    printf("m=");
    scanf("%d", &m);
    printf("n=");
    scanf("%d", &n);
    r = m % n;
    while( r != 0 ){
        m = n;
        n = r;
        r = m % n;
    }
    printf("GCD=%d¥n",  n);
    return 0;
}
```

条件式

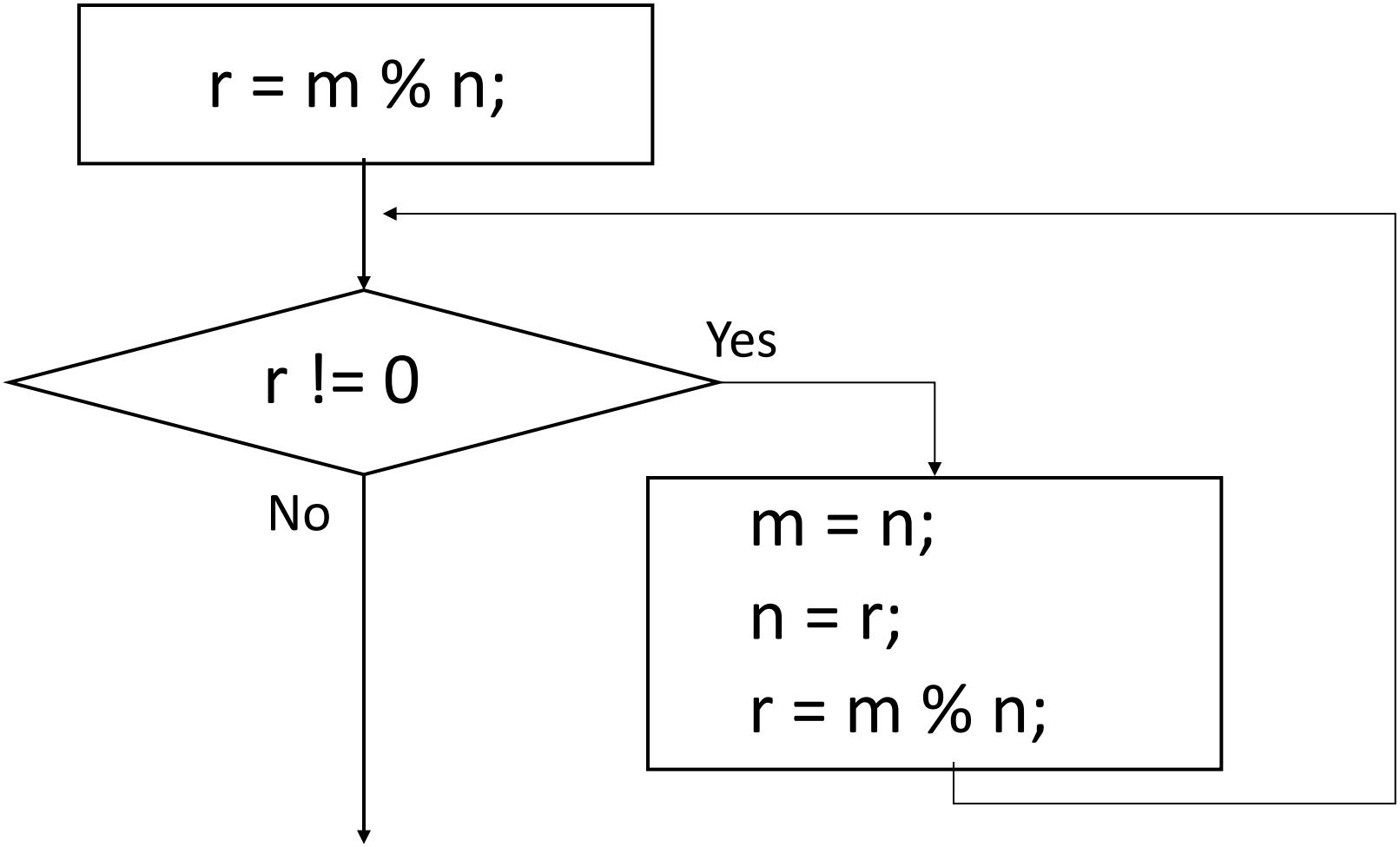
条件が成り立つ限り，
実行されつづける部分

ユークリッドの互助法

実行結果の例

```
m=12

n=8

GCD=4
```

プログラム実行順



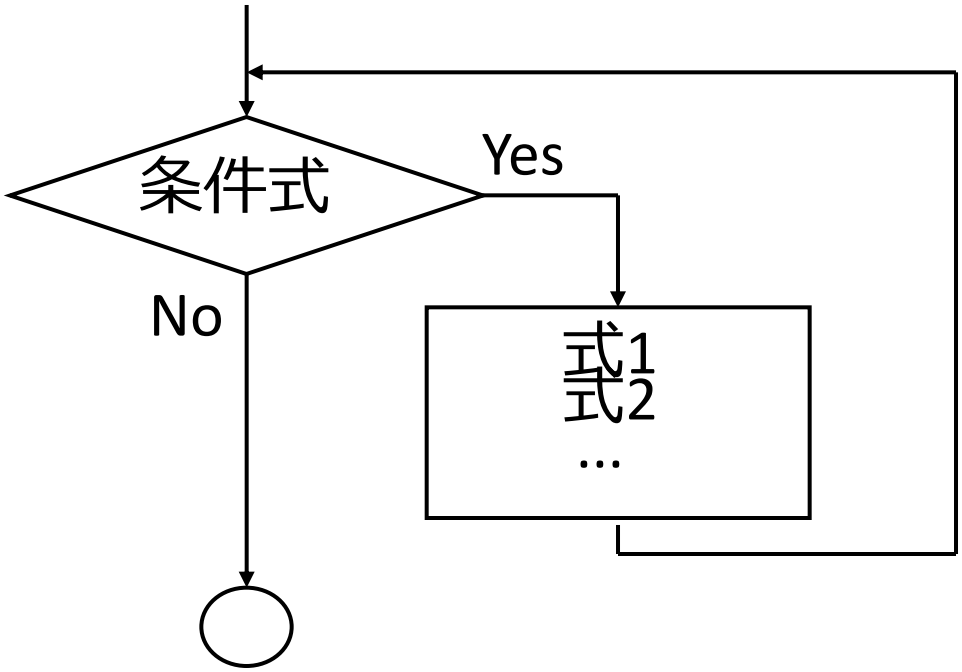
ユークリッドの互助法

最初の「 r = m % n; 」で,
m = 80, n = 35 とすると, r = 10 になる

		m の値	n の値	r の値
繰り返し 1 回目	r != 0 が成立する	m = 35	n = 10	r = 5
繰り返し 2 回目	r != 0 が成立する	m = 10	n = 5	r = 0
繰り返し 3 回目	r != 0 が成立しない			

while 文

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



- 何かの処理の繰り返し
- 繰り返しのたびに while 文で書かれた条件式の真偽が判定され，真である限り，while のあとに続く文が実行され続ける.

例題２．自然数の和

- 整数データ（Ｎとする）を読み込んで，１からＮまでの和を求めるプログラムを作る
- ここでは，練習のため，自然数の和の公式は使わずに，while文を用いる

例）　100 → 5050

自然数の和

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int i;
    int n;
    int sum;
    printf("n=");
    scanf("%d", &n);
    sum = 0;
    i = 1;
    while( i<=n ) {
        sum = sum + i;
        i = i + 1;
    }
    printf("n=%d, sum=%d¥n", n, sum);
    return 0;
}
```

条件式

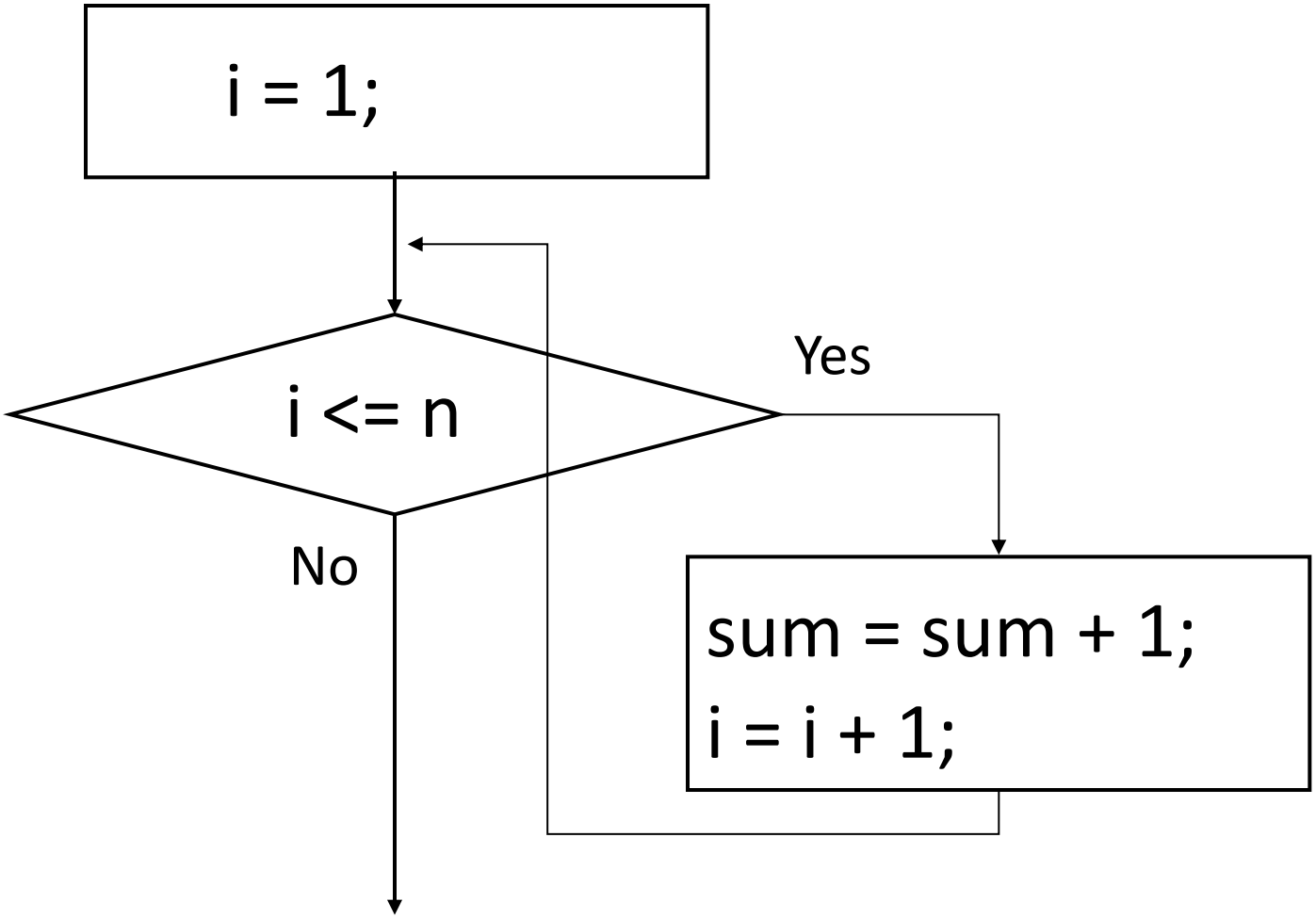
条件が成り立つ限り，
実行されつづける部分

自然数の和

実行結果の例

```
n=100

n=100, sum=5050
```

プログラム実行順



自然数の和

n = 7 とすると

		sum の値	i の値
繰り返し 1 回目	i <= 7 が成立する	sum = 0 + 1	i = 2
繰り返し 2 回目	i <= 7 が成立する	sum = 1 + 2	i = 3
繰り返し 3 回目	i <= 7 が成立する	sum = 3 + 3	i = 4
繰り返し 4 回目	i <= 7 が成立する	sum = 6 + 4	i = 5
繰り返し 5 回目	i <= 7 が成立する	sum = 10 + 5	i = 6
繰り返し 6 回目	i <= 7 が成立する	sum = 15 + 6	i = 7
繰り返し 7 回目	i <= 7 が成立する	sum = 21 + 7	i = 8
繰り返し 8 回目	i <= 7 が成立しない		

1 から 「繰り返し数」の和 + 「繰り返し数」+ 1

例題３．フィボナッチ数列

- 整数（Ｎとする）を読み込んで，１からＮまでのフィボナッチ数列を求めるプログラムを作る．
  - フィボナッチ数列f(n) とは

    $f(0)=1,\ f(1)=1,\ f(n)=f(n-1)+f(n-2)$
  - フィボナッチ数列を求めるために for 文を使う

例）　1,1,2,3,5,8,13,21,34,55,89,144,....

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int i;
    int n;
    int fn;
    int fn1;
    int fn2;
    printf("n=");
    scanf("%d", &n);
    fn1 = 1;
    fn2 = 1;
    for (i=2; i<=n; i++) {
        fn=fn1+fn2;
        fn2=fn1;
        fn1=fn;
        printf("f(%d) = %d¥n", i, fn);
    }
    return 0;
}
```

条件式

条件が成り立つ限り，
実行されつづける部分

順番に意味がある．

```
fn1=fn;
fn2=fn1;
```

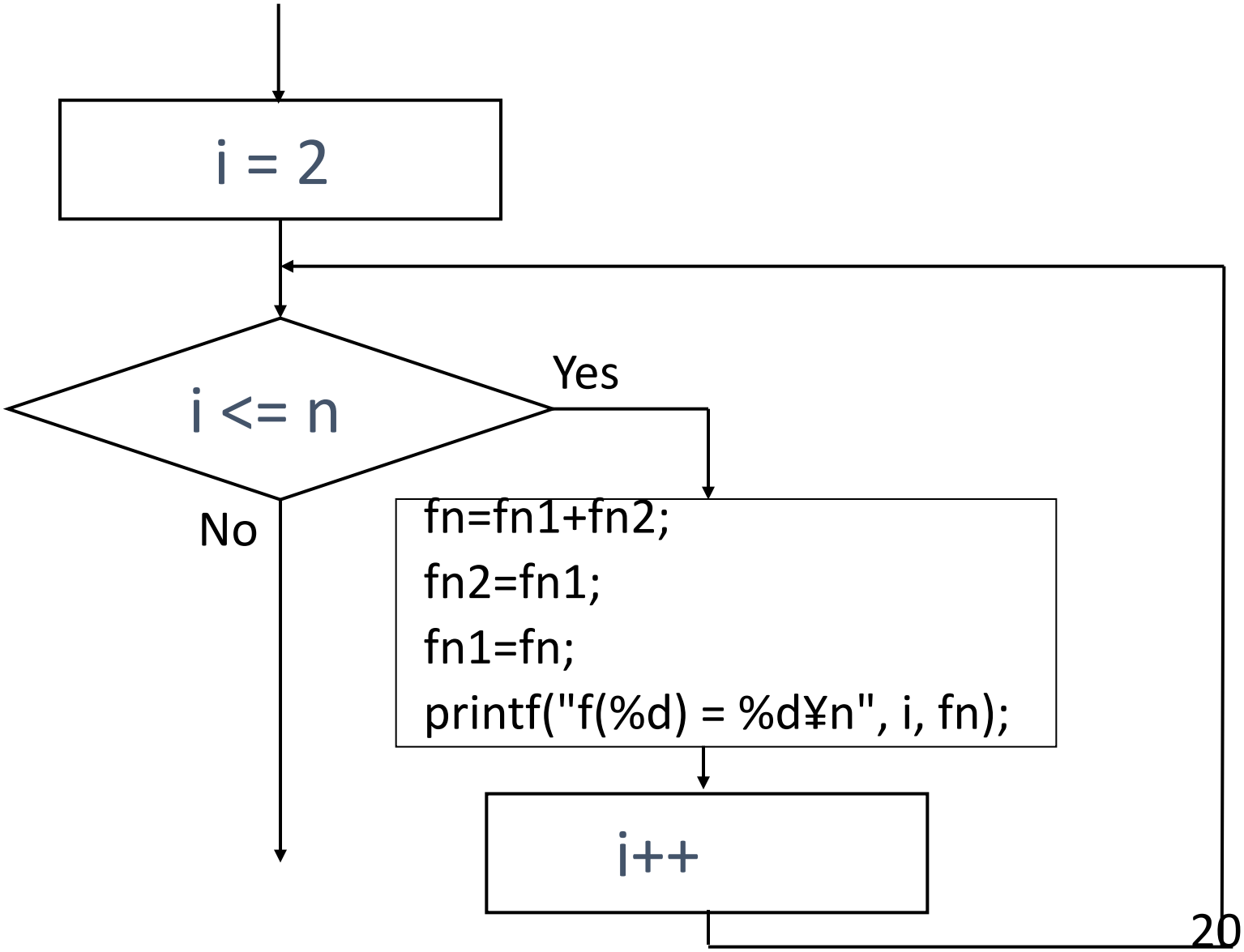
とはしないこと

フィボナッチ数列

実行結果の例

```
n=6

f(2) = 2

f(3) = 3

f(4) = 5

f(5) = 8

f(6) = 13
```

フィボナッチ数列



フィボナッチ数列

n = 5 とすると

	fn の値	fn2 の値	fn1 の値
i = 2	i <= 5 が成立する	fn = 1 + 1;	fn2 = 1; fn1 = 2
i = 3	i <= 5 が成立する	fn = 2 + 1;	fn2 = 2; fn1 = 3
i = 4	i <= 5 が成立する	fn = 3 + 2;	fn2 = 3; fn1 = 5
i = 5	i <= 5 が成立する	fn = 5 + 3;	fn2 = 5; fn1 = 8
i = 6	i <= 5 が成立しない		
	f i が入る	f i-1 が入る	f i が入る

++, -- の意味

＋＋

- インクリメントを行う演算子.
- インクリメントとは１足すこと.　オペランドに１を足して，オペランドに格納する.

- -

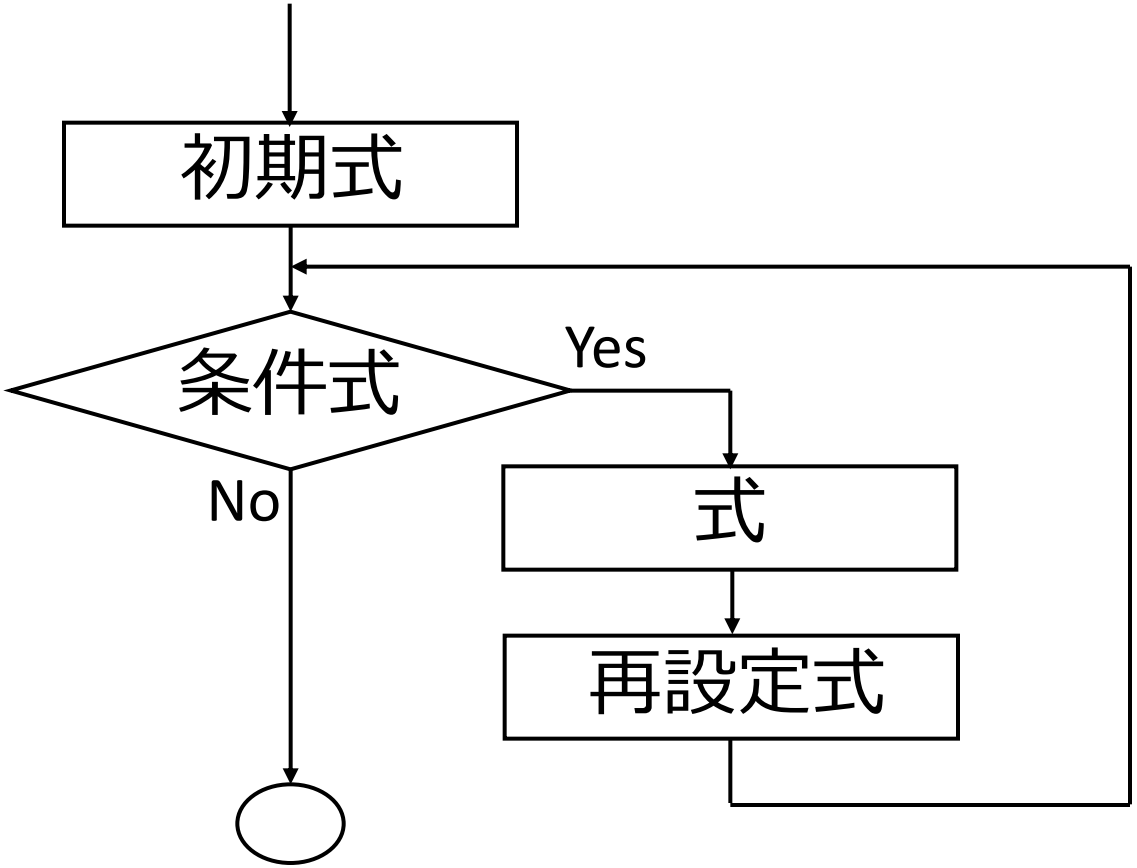
- デクリメント演算子.
- デクリメントとは１引くこと.　オペランドから１を引いて，オペランドに格納する.

for 文

```
for ( 初期式; 条件式; 再設定式 ) {
    式1;
    式2;
    …
}
```

- 初期式：繰り返しの最初に１回だけ実行される.
- 条件式：繰り返しのたびに，真偽が判定される（偽ならば繰り返しが終わる）.
- 再設定式： 繰り返しのたびに実行される.

for 文による繰り返し



1．まず，「初期式」を実行
2．次に，「条件式」を実行．条件式が成立すれば，
　　式と「再設定式」を実行し，「条件式」に戻る

例題４．自然数の和

- 数（Ｎとする）を読み込んで，１からＮまでの和を求めるプログラムを作る
- ここでは，練習のため，自然数の和の公式は使わずに，for文を用いる

例）　100 → 5050

自然数の和

実行結果の例

```
n=100

n=100, sum=5050
```

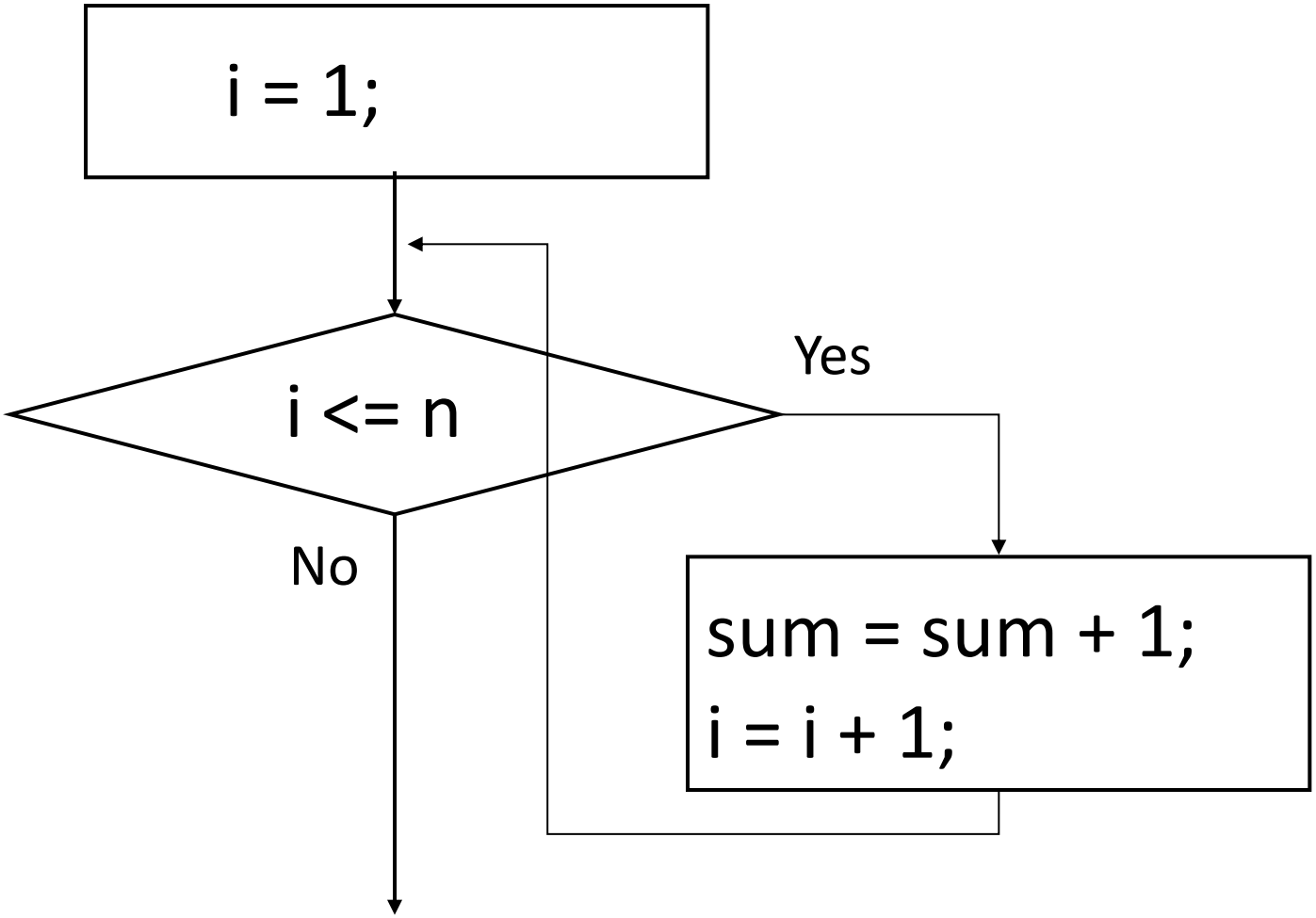
自然数の和

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int i;
    int n;
    int sum;
    printf("n=");
    scanf("%d", &n);
    sum = 0;
    for (i=1; i<=n; ++) {
        sum = sum + i;
    }
    printf("n=%d, sum=%d¥n", n, sum);
    return 0;
}
```

条件式

条件が成り立つ限り，
実行されつづける部分

プログラム実行順



自然数の和

n = 7 とすると　　sum の値

i = 1	i <= 7 が成立する	sum = 0 + 1
i = 2	i <= 7 が成立する	sum = 1 + 2
i = 3	i <= 7 が成立する	sum = 3 + 3
i = 4	i <= 7 が成立する	sum = 6 + 4
i = 5	i <= 7 が成立する	sum = 10 + 5
i = 6	i <= 7 が成立する	sum = 15 + 6
i = 7	i <= 7 が成立する	sum = 21 + 7
i = 8	i <= 7 が成立しない	

sum には 1 から i までの和が入る

for 文と while 文

```
for ( 初期式; 条件式; 再設定式 ) {
    式1;
    式2;
    …
}
```

```
初期式
while ( 条件式 ) {
    式1;
    式2;
    …
    再設定式
}
```

for 文と while 文の能力は同じ．自分にとって分かりやすい方と使うべし

例題５．九九の表

- 九九の表を表示するプログラムを作成する
  - 九九の表を表示するために，繰り返しの入れ子を使う

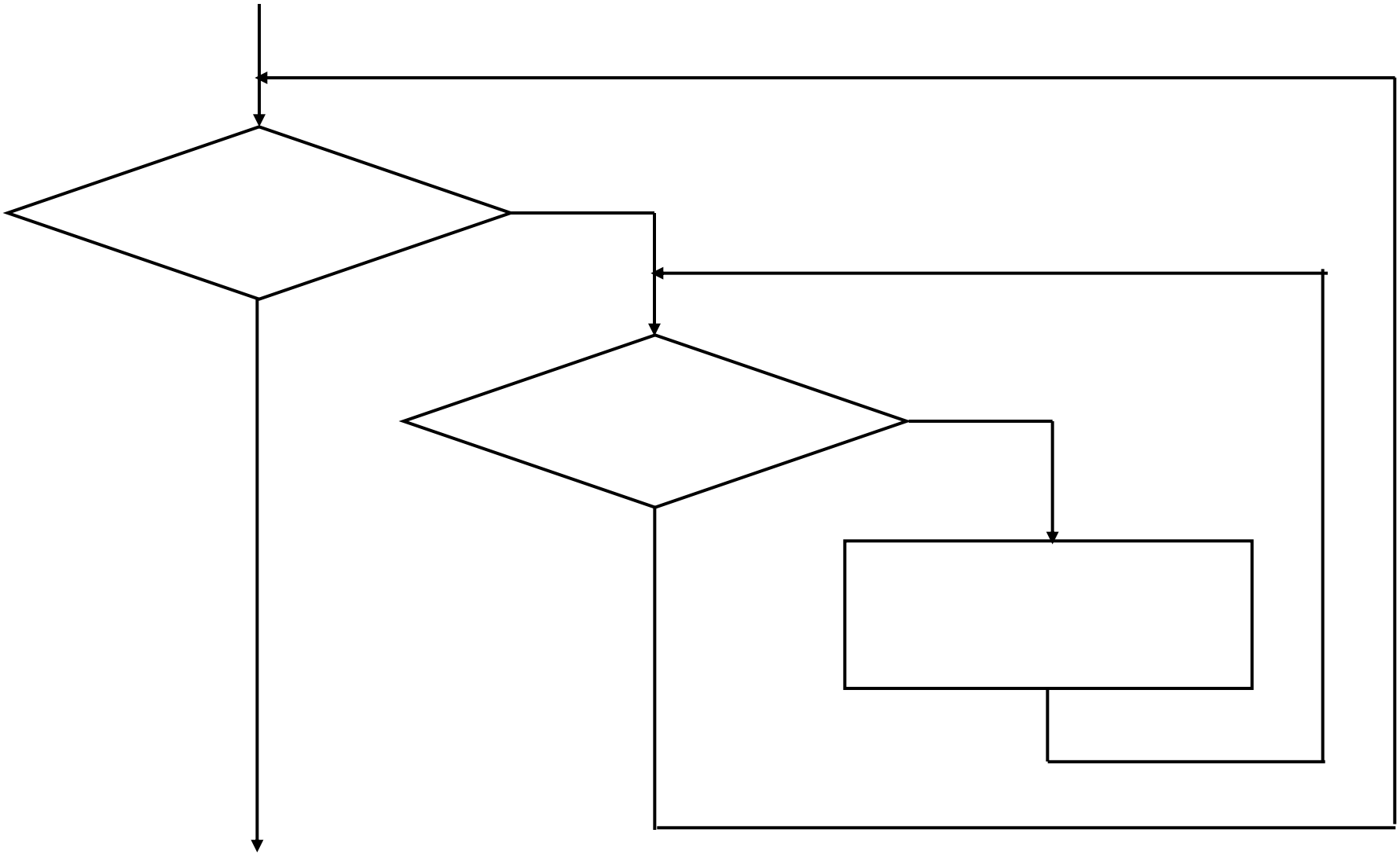
九九の表

```
#include <stdio.h>
int main()
{
    int i;
    int j;
    for (i=1; i<=9; i++) {
        for(j=1; j<=9; j++) {
            printf("%d, ", i*j);
        }
        printf("¥n");
    }
    return 0;
}
```

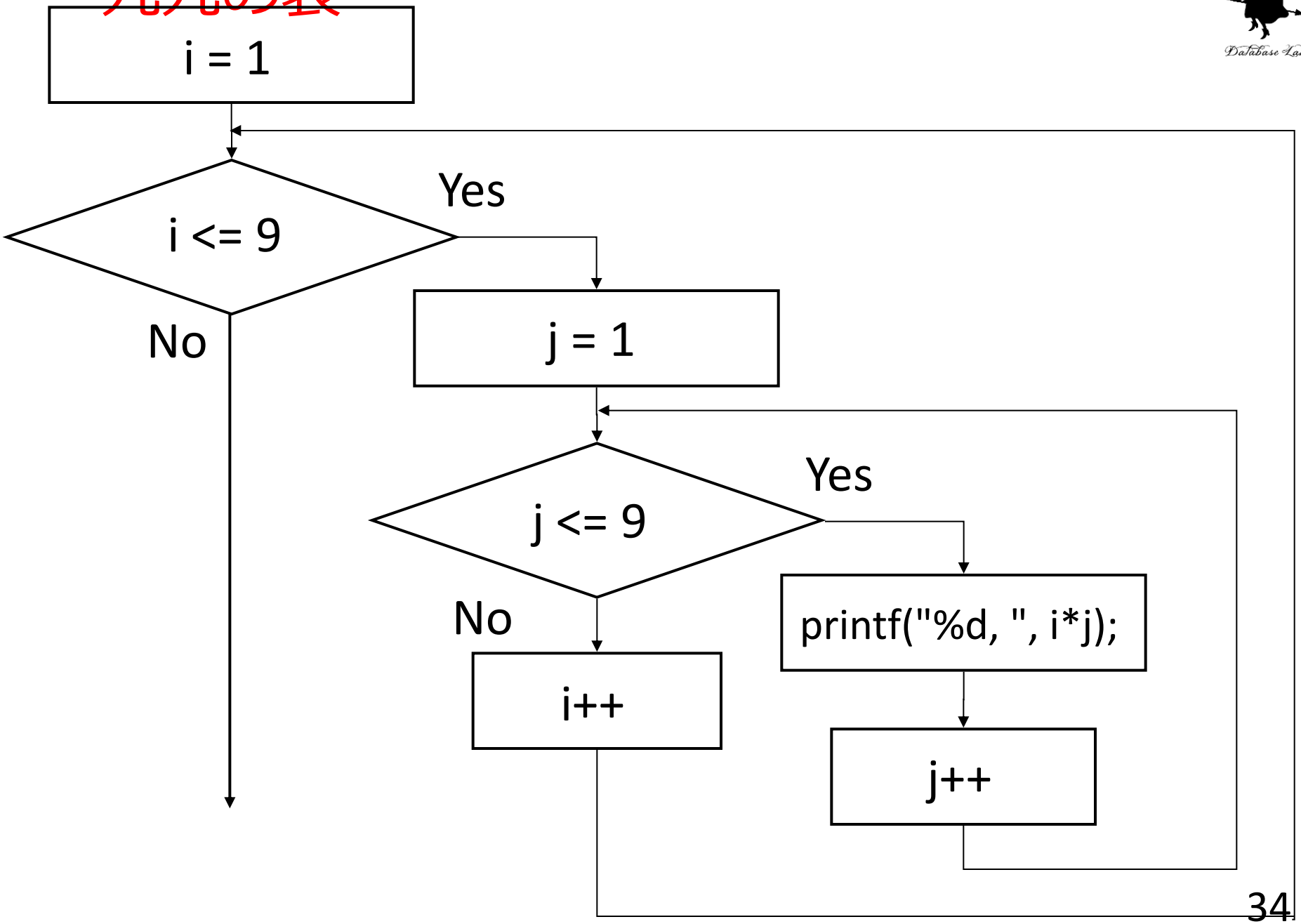
内側

外側

繰り返しの入れ子



九九の表



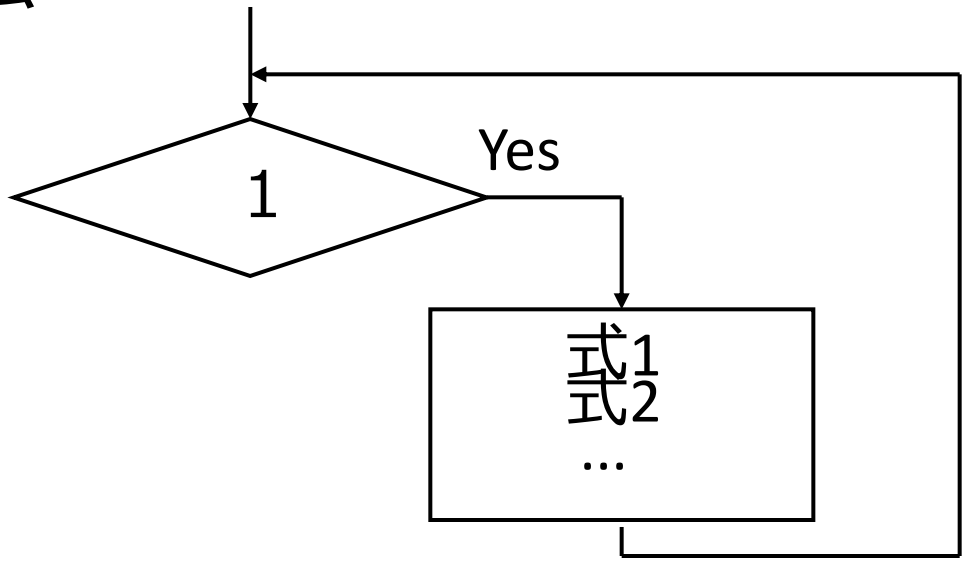
課題１

- 「例題１．最大公約数の計算」のプログラムを書き換えて，m,nを何度も入力して計算できるようにせよ

課題１のヒント
「m,nを何度も入力して計算できる」とは

繰り返し続けるプログラム

```
while (1) {
    式１;
    式２;
    …
}
```



この「１」は，「常に真である」
ことを示す特別な数

＊後述する do while 文を使っても「繰り返し続ける」ことは可

条件式での「１」の意味

値	意味
1	真(true)
0	偽(false)

課題１のヒント

- x と y を入れ替えるプログラム

```
int x;
int y;
int z;
z = x;
x = y;
y = z;
```

z は入れ替えのためだけに使う変数

課題２

1. ベクトルの値 (x_1, x_2, ... x_k）を１つづつ読み込んで，１つ読み込むごとに，次の計算を行うプログラムを作成しなさい.
   - ベクトル(x_1, x_2, ... x_k）の長さ
2. ２つのベクトルの値(x_1, x_2, ... ），(y_1, y_2, ...)を交互に読み込んで，1組読み込むことに次の計算を行うプログラムを作成しなさい.
   - ２つのベクトルの内積

課題２のヒント

- k 回めの繰り返しにおいて，長さkのベクトル (x1, x2, ... xk) の二乗和を求めるプログラム

```
sum = 0;
while (1) {
    scanf( "%lf", x );
    sum = sum + ( x * x );
    printf( "sum = %f¥n", sum );
}
```

課題３．三角関数

- θを読み込んで，$(\cos\theta + i\sin\theta)^n$ を計算するプログラムを作りなさい
  - なお，i は虚数単位
  - n は 1から100 まで繰り返すこと（つまり，計算は100回行う）

複素数の積

$z_1 = x_1 + i\,y_1$

$z_2 = x_2 + i\,y_2$ のとき

$$
\begin{aligned}
z_1 z_2 &= (x_1 + i\,y_1)(x_2 + i\,y_2) \\
&= x_1x_2 + x_1 i\,y_2 + i\,y_1x_2 + i\,y_1 i\,y_2 \\
&= \underbrace{x_1x_2 - y_1y_2}_{\text{実数部}} + i\,\underbrace{(x_1y_2 + y_1x_2)}_{\text{虚数部}}
\end{aligned}
$$

$(\cos\theta + i\sin\theta)^n = \cos n\theta + i\sin n\theta$

$$(\cos\theta + i\sin\theta)^2 = \cos^2\theta - \sin^2\theta + 2i\cos\theta\sin\theta$$

$$= \cos 2\theta + i\sin 2\theta$$

$$(\cos\theta + i\sin\theta)^3 = (\cos\theta + i\sin\theta)^2(\cos\theta + i\sin\theta)$$

$$= (\cos 2\theta + i\sin 2\theta)(\cos\theta + i\sin\theta)$$

$$= \cos 2\theta\cos\theta - \sin 2\theta\sin\theta$$

$$+ i(\cos 2\theta\sin\theta - \sin 2\theta\cos\theta)$$

$$= \cos(2\theta+\theta) + i\sin(2\theta+\theta)$$

$$= \cos 3\theta + i\sin 3\theta$$

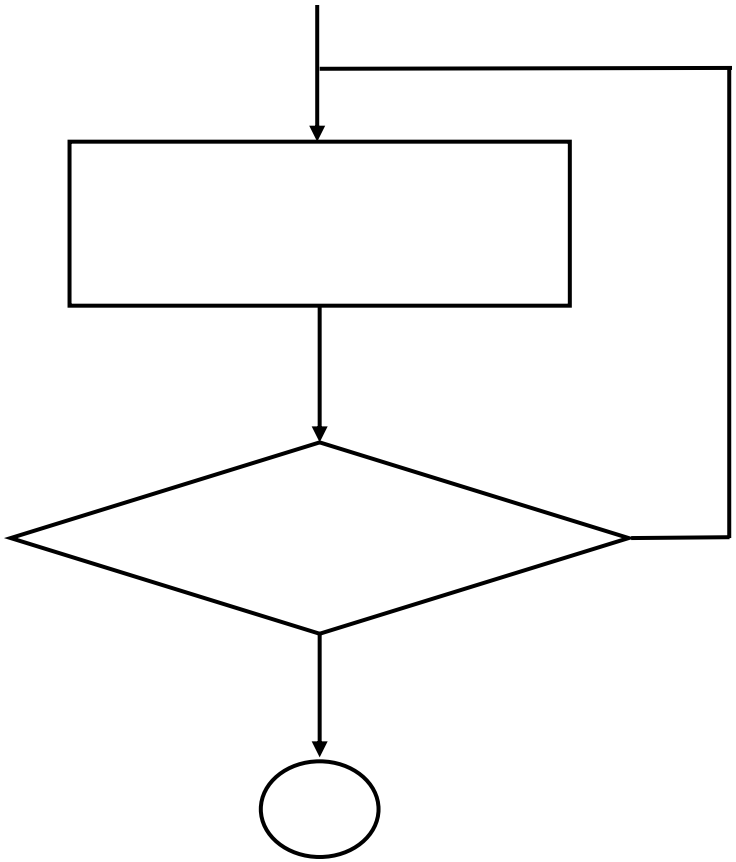
(以下同様に考える．数学的帰納法で証明できる)

課題４

- あるクラスの試験の点数から，その平均値と，標準偏差を計算するプログラムを作成しなさい.

より勉強したい人への付録

do while 文による繰り返し



while 文との違い： 必ず最初に１回は実行される

do while文

```
do {
   式1;
    ...
   式n;
} while (条件式);
```

do while 文が役に立つ場合

- 読み込み値のチェック
  - ０点以上，１００点以下のデータの読み込み
  - 間違いがあったら，読み込みを繰り返す

```
do {
    printf(" tensu=");
    scanf("%d", &tensu);
} while( ( tensu < 0 ) || ( tensu > 100 ) );
```

break文

- 繰り返しを中断
- break を含む最も内側の switch文あるいは繰り返し文（while, for 文など）から抜け出す.

break文

条件１の結果が成り立たない

条件２の結果が成り立つ

```
while (条件1){
    式;
    if (条件2){
        式;
        break; /* whileの外へ*/
    }
    printf("条件２が成り立てば実行されない．");
}
printf("whileの外¥n");
```

continue文

- 次の繰り返し実行を行うための文.
- continue を含む最も内側の繰り返し文（while文, for 文など）について，繰り返しの本体の残りを飛ばして，次の繰り返しを始める.

continue文

```
while (条件1){
    式;
    if (条件2){
        式;
        continue; /* whileの先頭へ*/
    }
    printf(“条件２が成り立たなければ実行される．");
}
printf("whileの外¥n");
```

break 文, continue 文のまとめ

- break 文
  - while 文, for 文での繰り返しから抜け出す

- continue 文
  - 繰り返し途中で，残りの処理を飛ばす