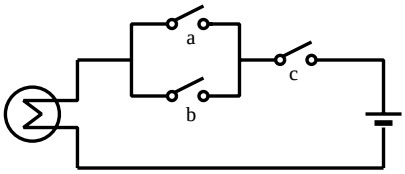
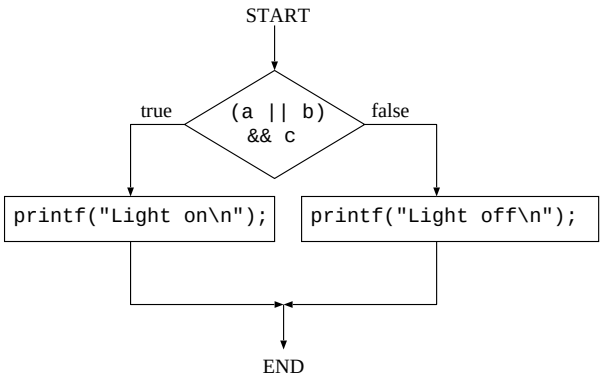
# 第３章 プログラムの流れの制御

## 3.1 if 文




```
void light(int a, int b, int c)
{
    if ((a || b) && c)
        printf("Light on\n");
    else
        printf("Light off\n");
}
```

流れ図（flowchart）



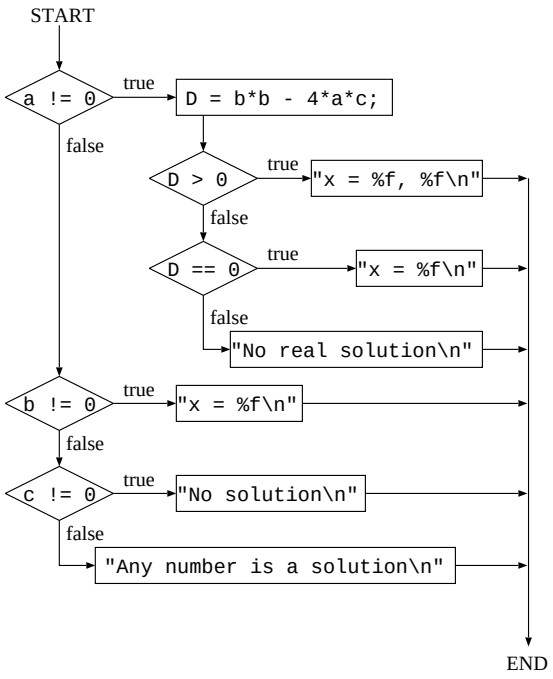



2 次方程式 $ax^2 + bx + c = 0$ の解を求める．

1. $a \neq 0$ のとき
   (a) 判別式 $D = b^2 - 4ac$ が正（$D > 0$）ならば $x = (-b \pm \sqrt{D})/2a$ （解の公式）
   (b) $D = 0$ ならば $x = -b/2a$
   (c) $D < 0$ ならば実数解なし
2. $a = 0, b \neq 0$ のとき $x = -c/b$
3. $a = 0, b = 0, c \neq 0$ のとき解なし
4. $a = 0, b = 0, c = 0$ のとき解は不定

```
if (a != 0) {
    double D = b*b - 4*a*c;
    if (D > 0)
        printf("x = %f, %f\n",
               (-b-sqrt(D)) / (2*a),
               (-b+sqrt(D)) / (2*a));
    else if (D == 0)
        printf("x = %f\n", -b / (2*a));
    else
        printf("No real solution\n");
} else if (b != 0)
    printf("x = %f\n", -c/b);
else if (c != 0)
    printf("No solution\n");
else
    printf("Any number is a solution\n");
```








## 3.2 while 文

```
while (《式》) 《文》
```

1 から $n$ $(n \geq 0)$ までの整数の和を求める

```
int sum(int n)
{
    int s = 0, i = 1;

    while (i <= n) {
        s += i;
        i++;
    }
    return s;
}
```

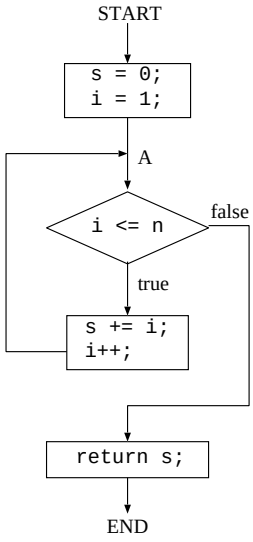
$n = 3$ のとき：

1. 最初は s $= 0$, i $= 1$.
2. s $= 0 + 1 = 1$, i $= 1 + 1 = 2$ となる.
3. s $= 1 + 2 = 3$, i $= 2 + 1 = 3$ となる.
4. s $= 3 + 3 = 6$, i $= 3 + 1 = 4$ となる.
5. i $\leq 3$ は偽なので while 文の実行を終了する.

$n = 0$ のとき：

1. 最初は s $= 0$, i $= 1$.
2. i $\leq 0$ は偽なので while 文の実行を終了する.

loop invariant




Aの位置で，次の関係が成り立つ．

$$s = \sum_{k=1}^{i-1} k, \qquad i \leq n+1$$

これが「証明」できれば， `i <= n` が偽のとき，

$$s = \sum_{k=1}^{i-1} k = \sum_{k=1}^{(n+1)-1} k = \sum_{k=1}^{n} k$$

だから，プログラムは正しく動作する．



関係

$$s = \sum_{k=1}^{i-1} k, \qquad i \leq n+1$$

を数学的帰納法で証明する．

最初にA点に到達したときは， $s = 0, i = 1$ なので

$$0 = \sum_{k=1}^{0} k, \qquad 1 \leq n+1$$

次に，Aの位置で成り立っていると仮定して，
ループを回ってきたときにも成り立つことを示す．
回る前の `s` と `i` の値を $s_0$， $i_0$ とすると

$$s_0 = \sum_{k=1}^{i_0-1} k, \qquad i_0 \leq n$$

回った後の `s` と `i` の値を $s_1$， $i_1$ とすると

$$s_1 = s_0 + i_0 \qquad i_1 = i_0 + 1$$

従って，

$$s_1 = s_0 + i_0 = \left(\sum_{k=1}^{i_0-1} k\right) + i_0 = \sum_{k=1}^{i_0} k = \sum_{k=1}^{i_1-1} k$$

$$i_1 = i_0 + 1 \leq n+1$$

となり，ループを回った後でも関係は成り立つ．

証明終り．



3.3 do 文

```
do 《文》 while (《式》);
```

$e^x$ の値をテーラー展開

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

を使って計算する

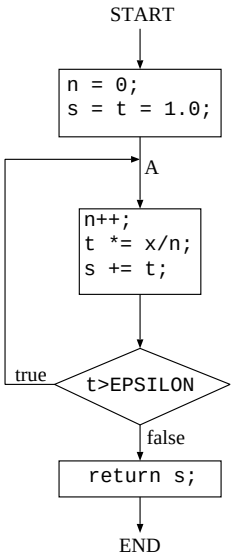
```
#define EPSILON 1e-8

double exponent(double x)
{
    double s = 1.0, t = 1.0;
    int n = 0;

    do {
        n++;
        t *= x/n;
        s += t;
    } while (t > EPSILON);

    return s;
}
```






図のA点において，次の関係が成り立っている．

$$t = \frac{x^n}{n!}, \qquad s = \sum_{k=0}^{n} \frac{x^k}{k!}$$

各自で証明してみよ．

3.4 for 文

for (《式$_1$》; 《式$_2$》; 《式$_3$》) 《文》

おおざっぱにいえば

《式$_1$》;
while (《式$_2$》) {
    《文》
    《式$_3$》;
}

のように動作する.

$n$ 以下の 3 つの自然数 $a$, $b$, $c$ $(a \le b \le c)$ で,
$a^2 + b^2 = c^2$ を満たすものを探す

```
main(int argc, char *argv[])
{
  int a, b, c, n;

  sscanf(argv[1], "%d", &n);
  for (a = 1; a <= n; a++)
    for (b = a; b <= n; b++)
      for (c = b; c <= n; c++)
        if (a*a + b*b == c*c)
          printf("a = %d, b = %d, c = %d\n",
                 a, b, c);
  printf("end.\n");
}
```



| | | | | |
|---|---|---|---|---|
| $(1,1,1)$ | $(1,1,2)$ | $(1,1,3)$ | $(1,1,4)$ | $(1,1,5)$ |
| | $(1,2,2)$ | $(1,2,3)$ | $(1,2,4)$ | $(1,2,5)$ |
| | | $(1,3,3)$ | $(1,3,4)$ | $(1,3,5)$ |
| | | | $(1,4,4)$ | $(1,4,5)$ |
| | | | | $(1,5,5)$ |
| | $(2,2,2)$ | $(2,2,3)$ | $(2,2,4)$ | $(2,2,5)$ |
| | | $(2,3,3)$ | $(2,3,4)$ | $(2,3,5)$ |
| | | | $(2,4,4)$ | $(2,4,5)$ |
| | | | | $(2,5,5)$ |
| | | $(3,3,3)$ | $(3,3,4)$ | $(3,3,5)$ |
| | | | $(3,4,4)$ | $(3,4,5)$ |
| | | | | $(3,5,5)$ |
| | | | $(4,4,4)$ | $(4,4,5)$ |
| | | | | $(4,5,5)$ |
| | | | | $(5,5,5)$ |

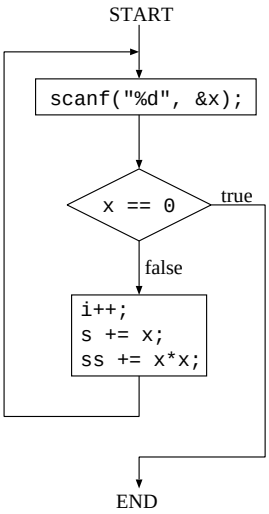
for 文

for (《式$_1$》; 《式$_2$》; 《式$_3$》) 《文》

における各《式》は省略できる.
《式$_1$》を省略すると初期化しない.
《式$_3$》を省略すると繰り返しごとの設定をしない.
《式$_2$》を省略すると止まらない（無限ループ）.



break 文




```
for (;;) {
    scanf("%d", &x);
    if (x == 0) break;
    i++;
    s += x;
    ss += x*x;
}
```



3.4 再帰呼出し（recursive call）

数学における再帰的定義（recursive definition）

初項 1，公比 2 の等比数列 $\{a_i\}$ $(i = 1, 2, 3, \ldots)$

$a_1 = 1$
$a_i = 2a_{i-1} \qquad (i \ge 2)$

$n$ の階乗 $n!$

$n! = 1 \times 2 \times \ldots \times n$

正確には

$0! = 1$
$n! = n \times (n-1)! \qquad (n \ge 1)$

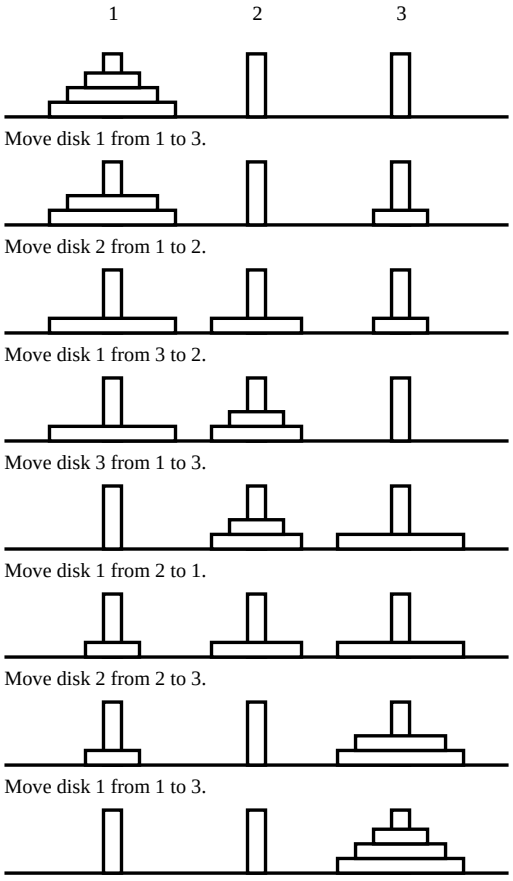
関数の再帰的定義

$n$ の階乗 $n!$ を求める

```
int fact(int n)
{
    if (n == 0)
        return 1;
    else
        return n * fact(n-1);
}
```

```
fact(3) = 3 * fact(2)
        = 3 * (2 * fact(1))
        = 3 * (2 * (1 * fact(0)))
        = 3 * (2 * (1 * 1))
        = 3 * (2 * 1)
        = 3 * 2
        = 6
```

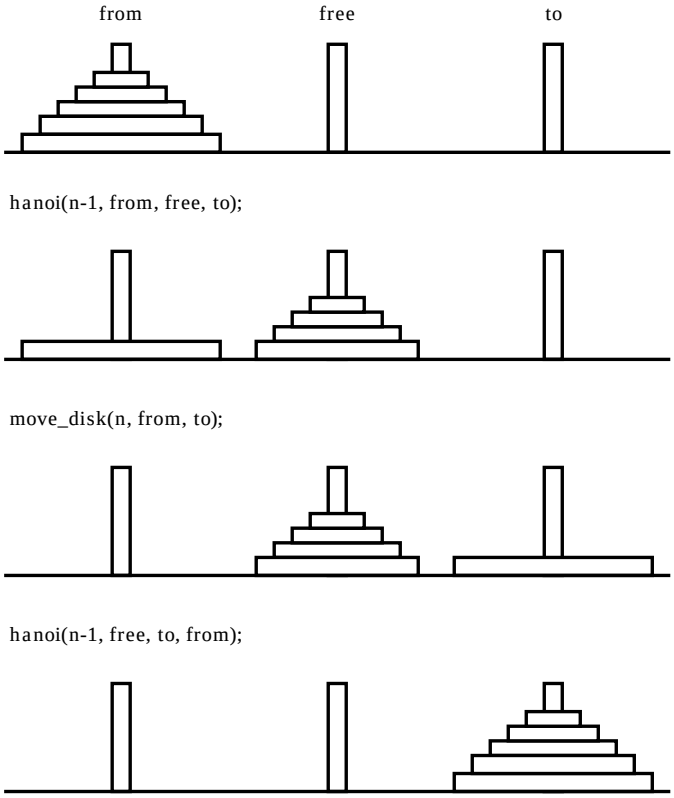
ハノイの塔






基本ステップ

hanoi(n,from,to,free) の実行






```
void move_disk(int n, int from, int to)
{
  printf("Move disk %d from %d to %d.\n",
         n, from, to);
}

void
hanoi(int n, int from, int to, int free)
{
    if (n == 1)
        move_disk(1, from, to);
    else {
        hanoi(n-1, from, free, to);
        move_disk(n, from, to);
        hanoi(n-1, free, to, from);
    }
}
```

hanoi(3,1,3,2) の実行結果

```
Move disk 1 from 1 to 3.
Move disk 2 from 1 to 2.
Move disk 1 from 3 to 2.
Move disk 3 from 1 to 3.
Move disk 1 from 2 to 1.
Move disk 2 from 2 to 3.
Move disk 1 from 1 to 3.
```