資訊之芽 0328
review, homework, and while loop

by 王姵心&林致民

複習-break

break會跳出一層迴圈

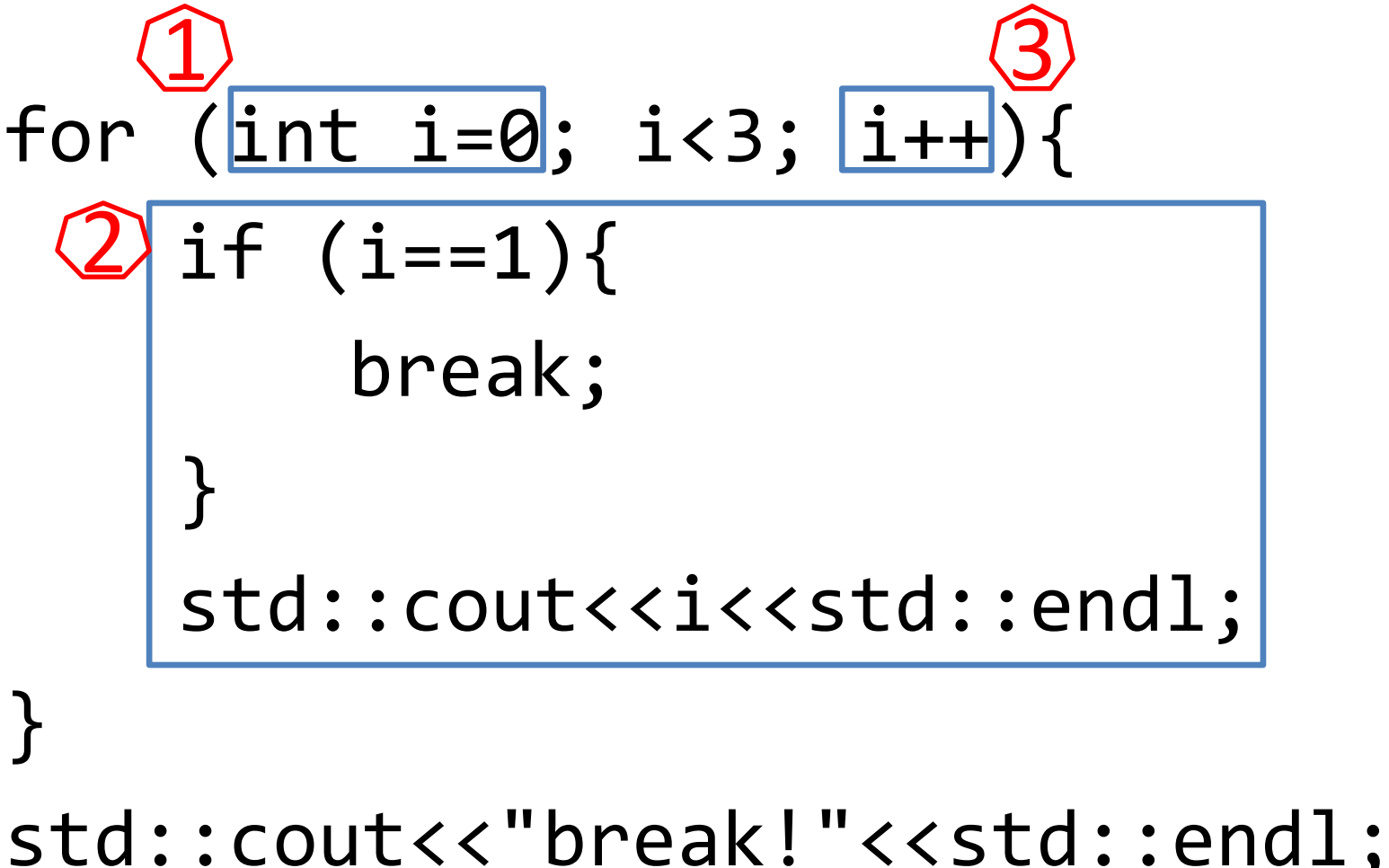
只有一層喔!!!

而且是跳"迴圈"!!!

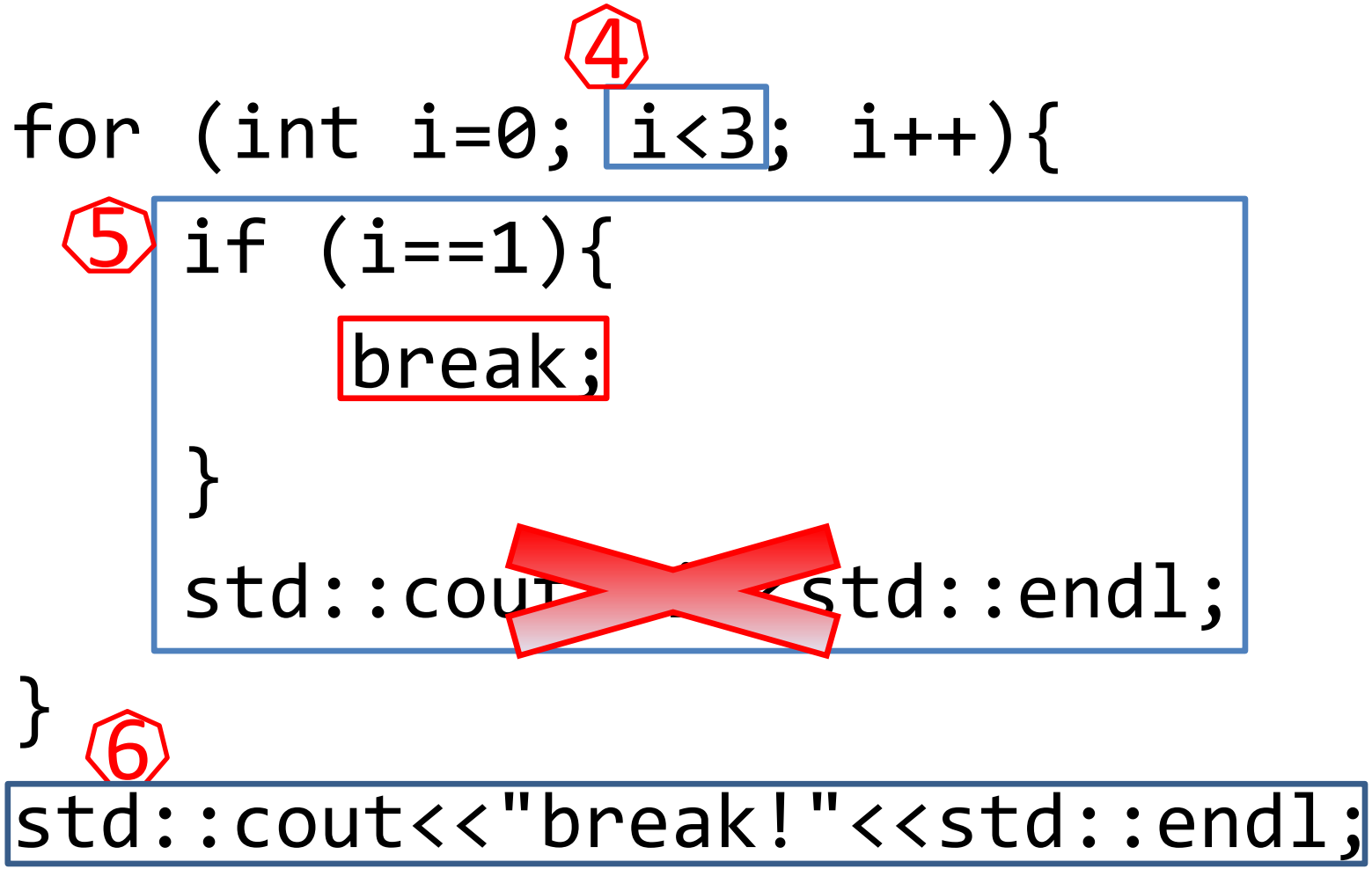
break之後的敘述都不會執行

(包含更新值的部分)

複習-break


①
③
②
for (int i=0; i<3; i++){
if (i==1){
break;
}
std::cout<<i<<std::endl;
}
std::cout<<"break!"<<std::endl;

複習-break


④
for (int i=0; i<3; i++){
⑤
if (i==1){
break;
}
std::cout<<std::endl;
}
⑥
std::cout<<"break!"<<std::endl;

break兩層迴圈

設一個變數flag,當flag==1時表示要break二層

```
int flag=0;
for (int i=0; i<3; i++){
     for (int j=0; j<3; j++){
          if (j==2){
               flag = 1;
               break;
          }
     }
     if (flag == 1){
          break;
     }
}
```

複習-continue

continue之後的敘述不會執行

會更新變數值

並跳至迴圈最前面繼續判斷和執行迴圈

複習-continue

①　　③

```
for (int i=0; i<3; i++){
  ②  if (i==1){
          continue;
      }
      std::cout<<i<<std::endl;
}
std::cout<<"break!"<<std::endl;
```

複習-continue

④ ⑥

```
for (int i=0; i<3; i++){
  ⑤ if (i==1){
        continue;
    }
    std::cout<<i<<std::endl;
}
std::cout<<"break!"<<std::endl;
```

複習-continue

⑦ ⑨

```
for (int i=0; i<3; i++){
  if (i==1){
      continue;
  }
  std::cout<<i<<std::endl;
}
std::cout<<"break!"<<std::endl;
```

⑧

複習-continue

10

```
for (int i=0; i<3; i++){
    if (i==1){
        continue;
    }
    std::cout<<i<<std::endl;
}
```

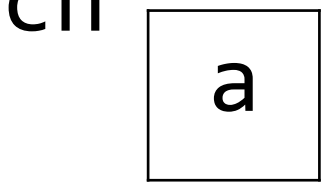
11

```
std::cout<<"break!"<<std::endl;
```

複習-string

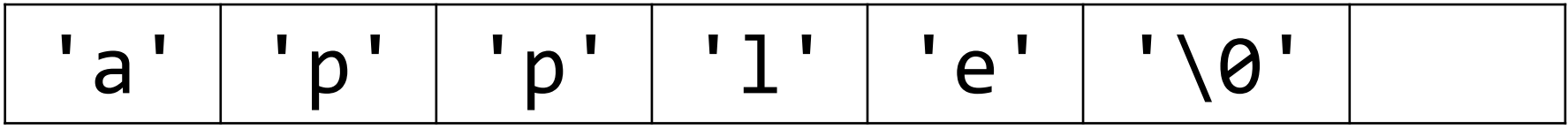
char: 字元

```
char ch = 'a';
```



複習-string

str




```
str[0] == 'a';
str[0] != a ;
str[0] != "a";
```

複習-string

```
char str[2][7] = {"apple","girl"};
```

str

0	'a'	'p'	'p'	'l'	'e'	'\0'	
1	'g'	'i'	'r'	'l'	'\0'		
	0	1	2	3	4	5	6

複習-string

```
#include <iostream>
int main(){
    char name[20] = {0};
    std::cin>>name;
    std::cout<<"Hello, "
             <<name<<"!";
    return 0;
}
```

複習(cont)

判斷兩陣列是否相等

```
if (array1==array2) (X)

for (int i=0; i< len; i++){
    if (array1[i]==array2[i])
}                          (O)
```

複習(cont)

複製陣列

```
array1=array2          (X)

for (int i=0; i< len; i++){
    array1[i]=array2[i]
}                      (O)
```

複習-strlen

strlen

一個函式

在cstring裡

有一個回傳值(長度)

複習-strcmp

strcmp

一個函式

在cstring裡

有回傳值(0或>0或<0)

踩地雷(214)

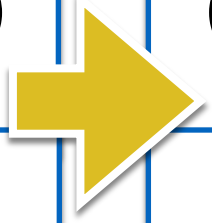
先來觀察：

(i-1, j-1)	(i-1, j)	(i-1, j+1)
(i , j-1)	(i , j)	(i , j+1)
(i+1, j-1)	(i+1, j)	(i+1, j+1)

踩地雷(214)

歸納：

(i-1, j-1)	(i-1, j)	(i-1, j+1)
(i , j-1)	(i , j)	(i , j+1)
(i+1, j-1)	(i+1, j)	(i+1, j+1)



(-1, -1)	(-1, 0)	(-1, +1)
(0, -1)	(0 , 0)	(0, +1)
(+1, -1)	(+1, 0)	(-1, -1)

踩地雷(214)

建表：

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

踩地雷(214)

設當前點為 (i , j)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

踩地雷(214)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

踩地雷(214)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

踩地雷(214)

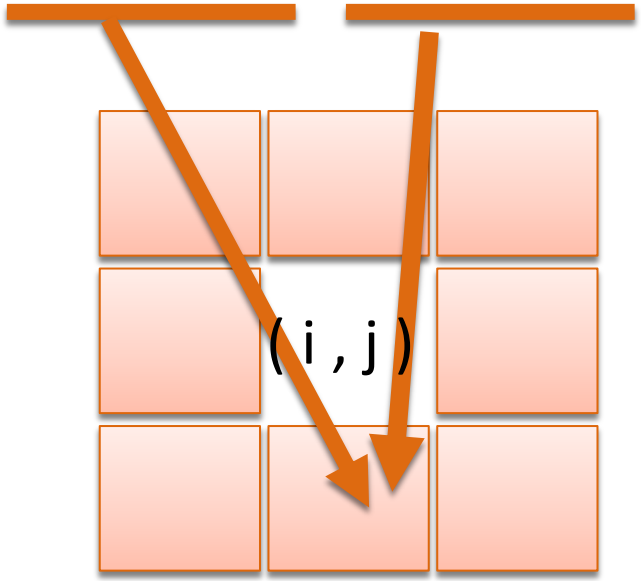
設當前點為 (i , j)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 0



踩地雷(214)

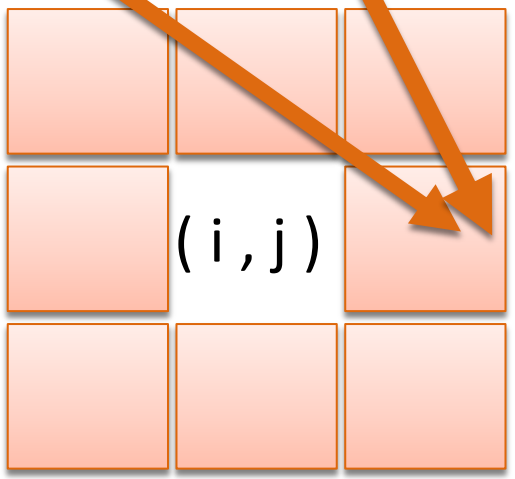
設當前點為 (i , j)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設 k = 0 ~ 7

k = 1

周圍的點為 (i + d[k][0] , j + d[k][1])



踩地雷(214)

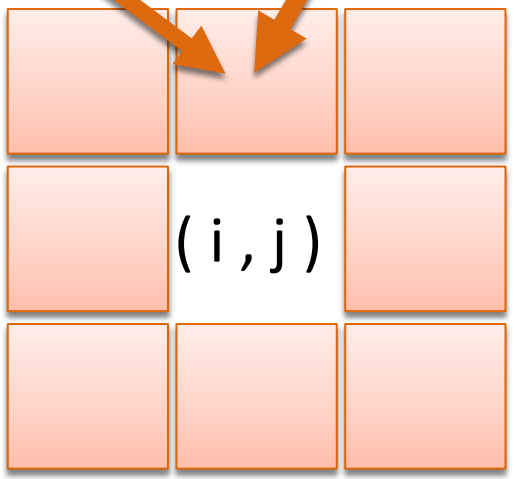
設當前點為 (i , j)

設 k = 0 ~ 7

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 2



踩地雷(214)

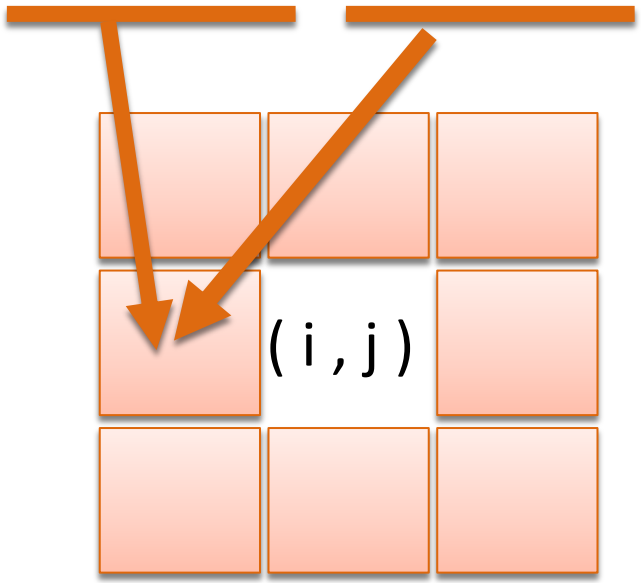
設當前點為 (i , j)

設 k = 0 ~ 7

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

k = 3

周圍的點為 (i + d[k][0] , j + d[k][1])



踩地雷(214)

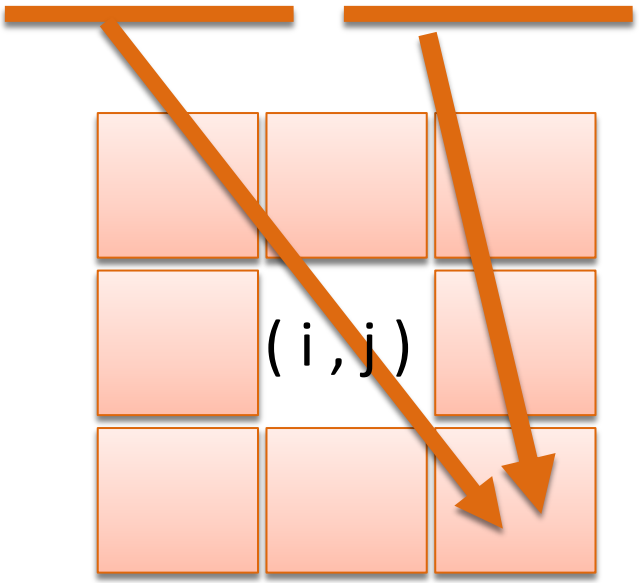
設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

k = 4



踩地雷(214)

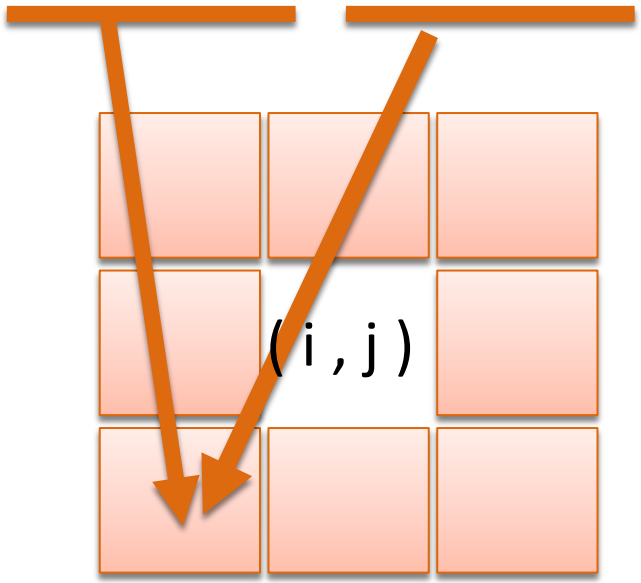
設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

k = 5



踩地雷(214)

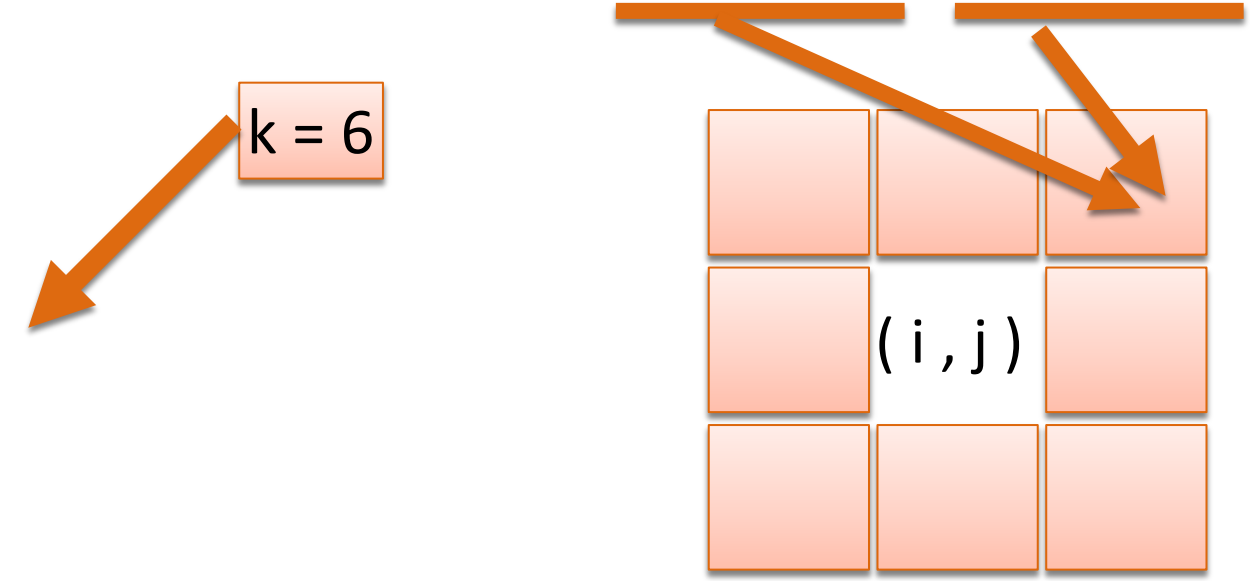
```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 6



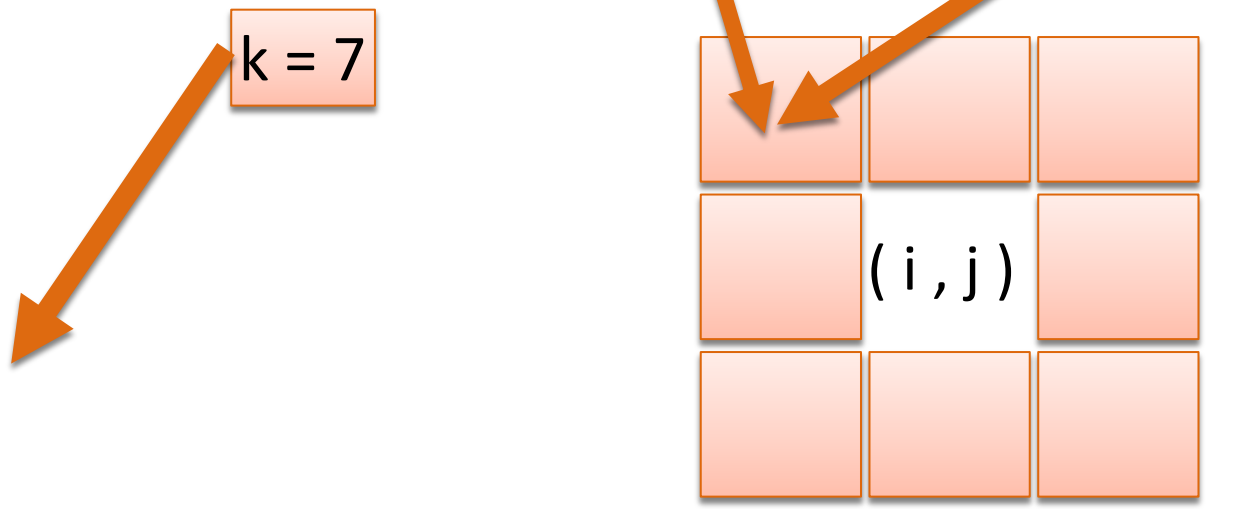
踩地雷(214)

設當前點為 (i , j)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

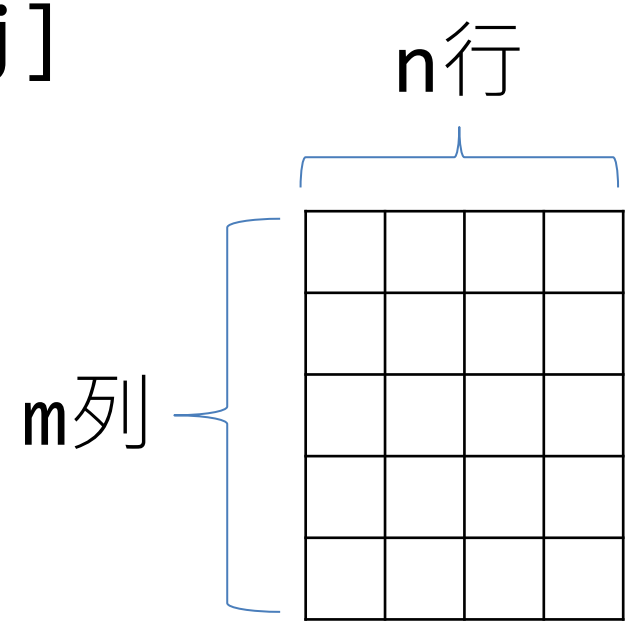
設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])



Homework#214踩地雷

```
for (...){                  //第0列~第m-1列
    for (...){              //第0行~第n-1行
        //map[i][j]
    }
}
```



Homework#214踩地雷

```
for (...){                  //第0列~第m-1列
    for (...){              //第0行~第n-1行
        for (...){  //第[i][j]格周圍8格
            if (...)//確認沒有超出邊界
            /*算出第[i][j]格周圍地雷數*/
        }
    }
}
```

Homework#215全都是零

Sample Input

3

0 1 1

1 0 1

0 0 0

Sample Output

2

Homework#215全都是零

```
for (...){
    for (...){
        /*檢查第i列是否皆為0*/
    }
}
```

Homework#215全都是零

```
for (...){
    check = 0;
    for (...){
        /*檢查第i列是否皆為0*/
        /*若input[i][j]==0, check++*/
    }
    /*第i列檢查完畢,判斷0的個數*/
    /*如果個數正確表示找到答案*/
}
```

Homework#215全都是零

```
for (...){
    check = 0;
    for (...){
        /*檢查第i列是否皆為0*/
        /*若input[i][j]==0, check++*/
    }
    /*第i列檢查完畢,判斷0的個數*/
    /*如果個數正確表示找到答案*/
    /*如果都沒有,則輸出-1*/
}
```

Homework#215全都是零

Sample Input

3

0 1 1

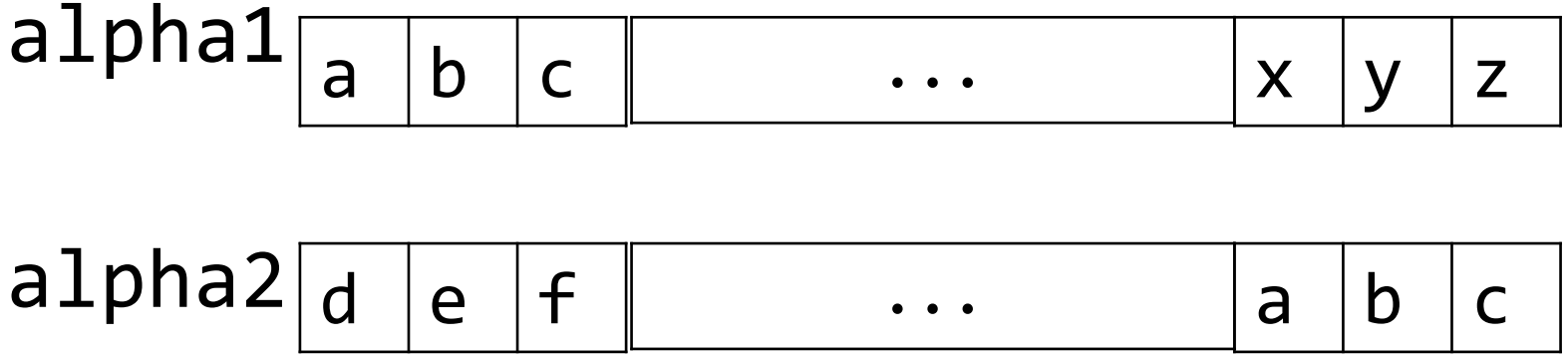
1 0 1

0 0 0

Sample Output

2

Homework#217凱薩密碼




```
char alpha1[26]=
{'a','b','c', ..., 'x','y','z'};
char alpha2[26]=
{'d','e','f', ..., 'a','b','c'};
```

Homework#217凱薩密碼

```
for (int i...){       //輸入字串的第i格
    for (int j...){ //比對字串的第j格
        /*如果在alpha1裡找到符合的*/
        /*就對應到alpha2,複製到output*/
    }
}
```

Homework#217凱薩密碼

每個字元都有一個對應的ASCII碼,

例如' '對應的ASCII碼是20,

'A'對應的ASCII碼是65

Example:

```
int i='A';
std::cout<<A<<endl;
```

輸出結果就會是65

Homework#217凱薩密碼

目前ASCII碼的範圍是0~127,至於每個ASCII碼所對應的字元大家可以自行上網搜尋,在維基百科裡就有囉~!

常用的字元及其ASCII碼:

A~Z:65~90　a~z:97~122

Homework#217凱薩密碼

```
for (int i...){ //輸入字串的第i格
    if (input[i] == 'x')
        input[i] = 'a';
    else if (input[i] == 'y')
        input[i] = 'b';
    else if (input[i] == 'z')
        input[i] = 'c';
    else
        input[i] = input[i] + 3;
}
```

Homework#218字串種數

Sample Input

5

aaaa

abc

aaaa

abc

bb

Sample Output

3

Homework#218字串種數

```
int count = 0;
for (int i...){       //第i個字串
    for (int j...){  //比較第0~i-1個字串
        /*如果有一樣的就break*/
        /*如果比到最後都不一樣就count++*/
    }
}
```

Homework#218字串種數

Sample Input

5

aaaa

abc

aaaa

abc

bb

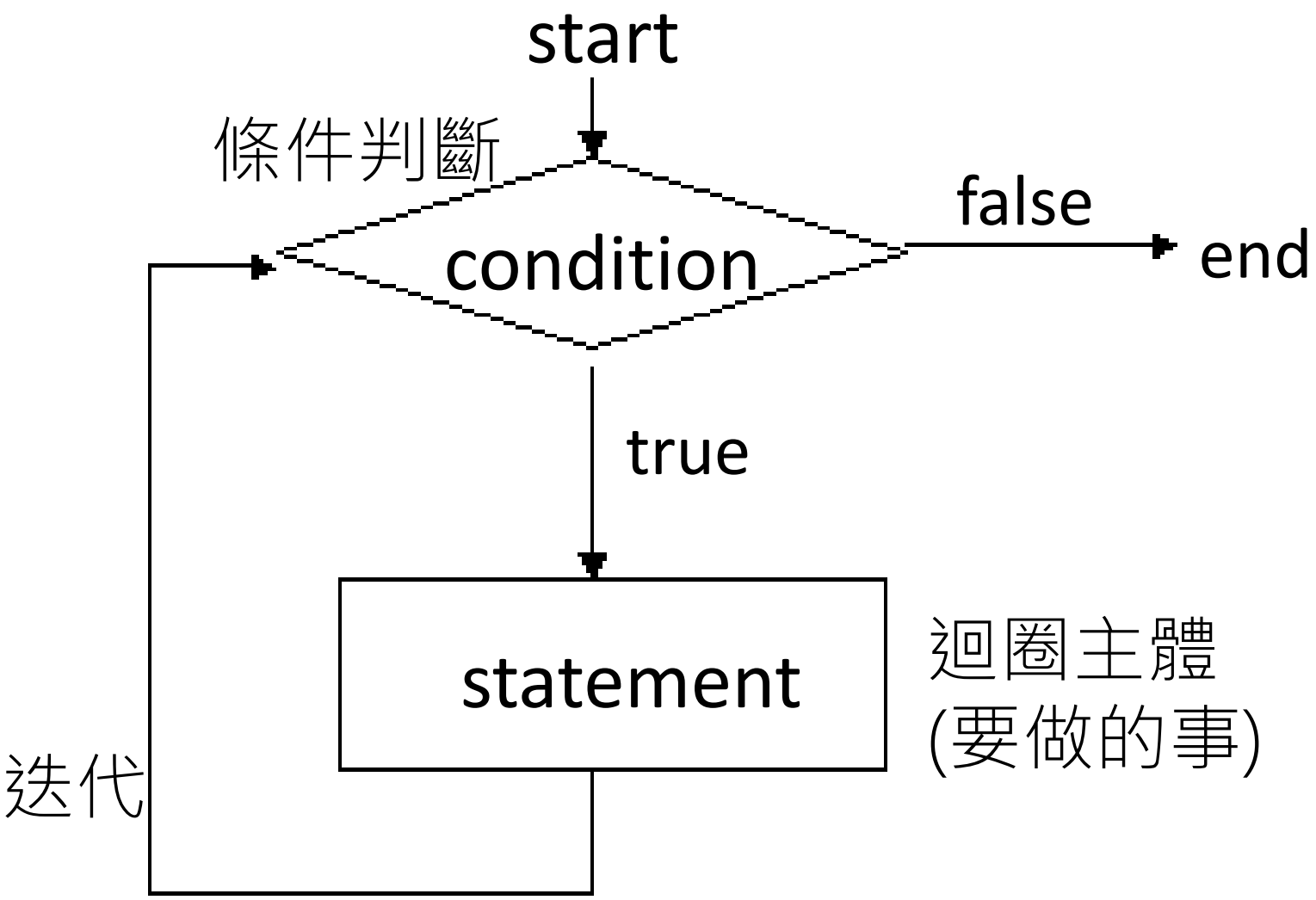
Sample Output

3

while迴圈

```
while (執行條件){
    要做的事(指令);
}
```

while迴圈



do while迴圈

```
do{
    要做的事(指令);
}while(執行條件);
```

跟while的差別: 一定會先做一次!

do while迴圈

例:猜密碼,要先輸入一次才能判斷正確與否

```
int i;
do{
    std::cin>>i;
}while(i!=25);
```

Example

題目說明:不斷輸入整數並將其加總,直到輸入的不是整數,最後印出總和,結尾記得換行

```
Sample Input
99
-100
q
Sample Output
-1
```

Example

```
int i, sum = 0;
while (std::cin>>i){
    sum = sum + i;
}
std::cout<<sum<<std::endl;
```

Practice#220

假如n是偶數,那就n/2,

假如n是奇數,那就3n+1,

問對於任意正整數,n操作到1所需要的次數是多少?

Ex: 5->16->8->4->2->1,需操作5次

Sample Input

5

Sample Output

5

Practice#220

```
int n, i=0;
std::cin>>n;
while (n!=1){
    if (n%2 == 0){
        n = n/2;
        i++;
    } else{
        n = 3 * n + 1;
        i++;
    }
}
std::cout<<i<<std::endl;
```