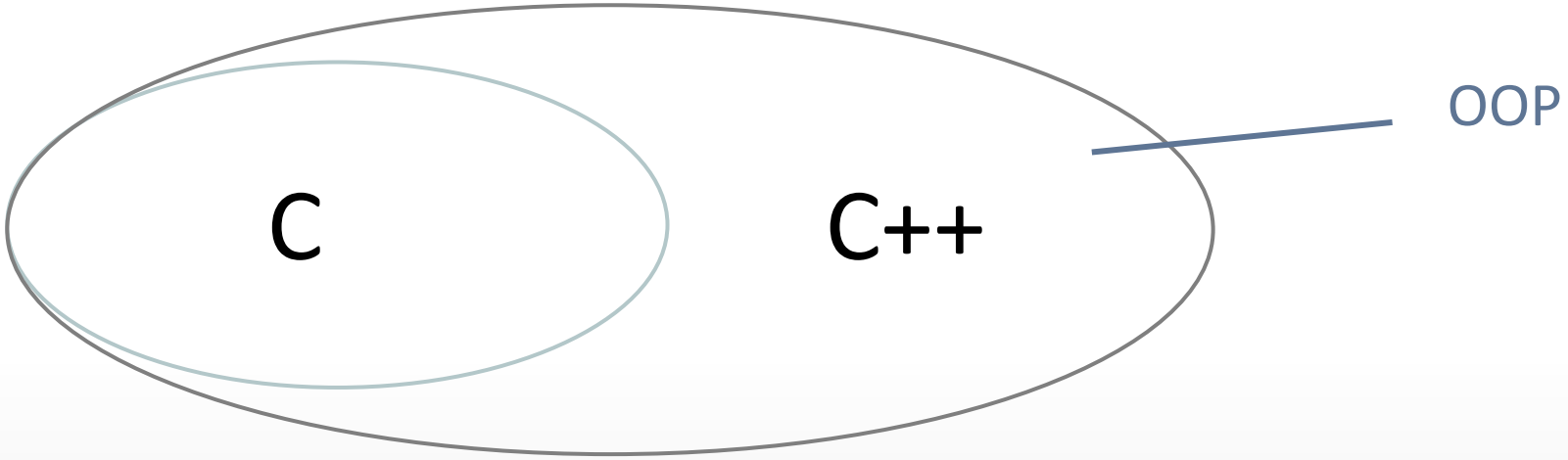
Introduction

/ C++是什麼？

C++

The relationship between C and C++ is like milk tea and milk tea with tapioca balls



Advantage

- lay the groundwork
- have idea of programming
- Object Oriented Programming (OOP)
- efficacy

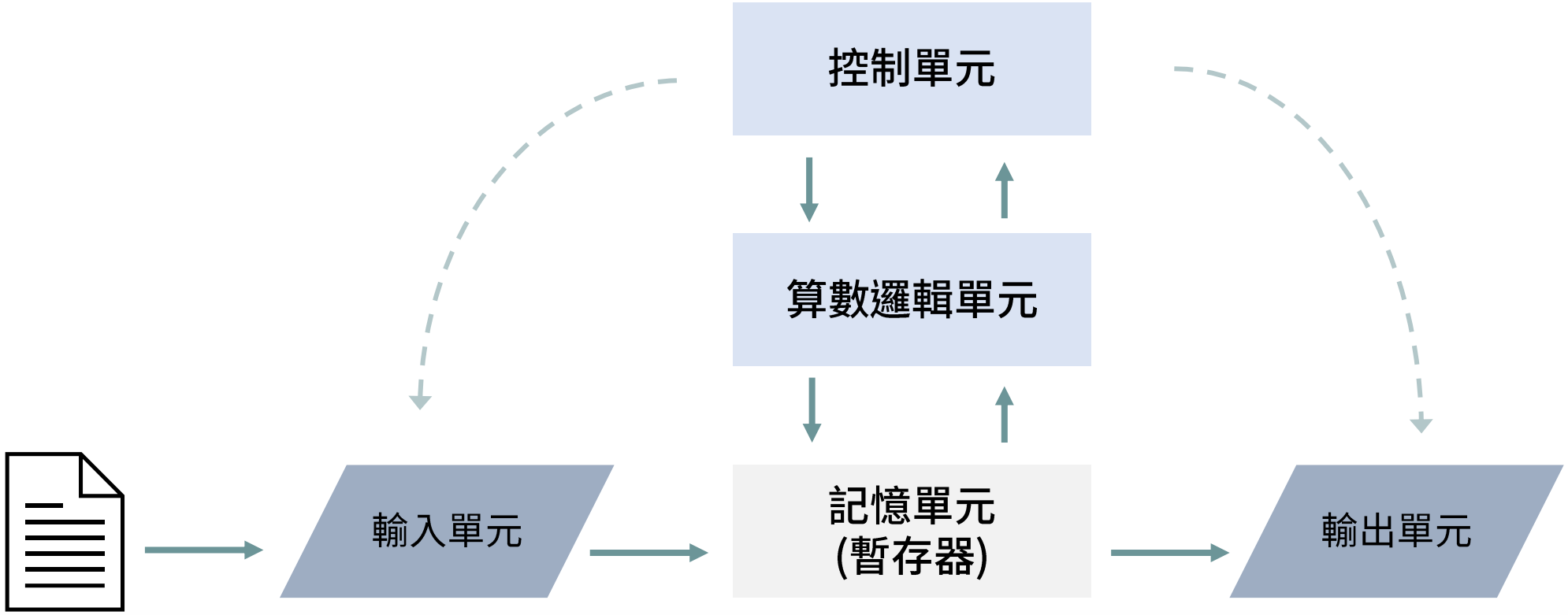
Disadvantage

- harder than others
- Garbage Collection

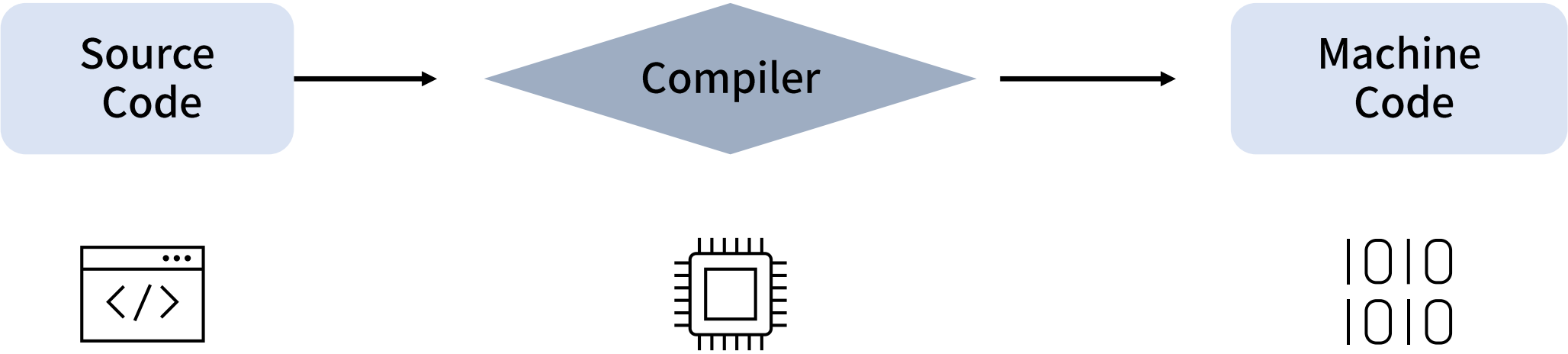
Process

/ input -> calculate -> output

Unit



Compiler



Base

/ 為什麼第一個程式都是Hello World？

Base

```
#include<iostream>
using namespace std;

int main(){

    statement...

}
```

Output

```
#include<iostream>
using namespace std;

int main(){

    cout<<"hello world";

}
```

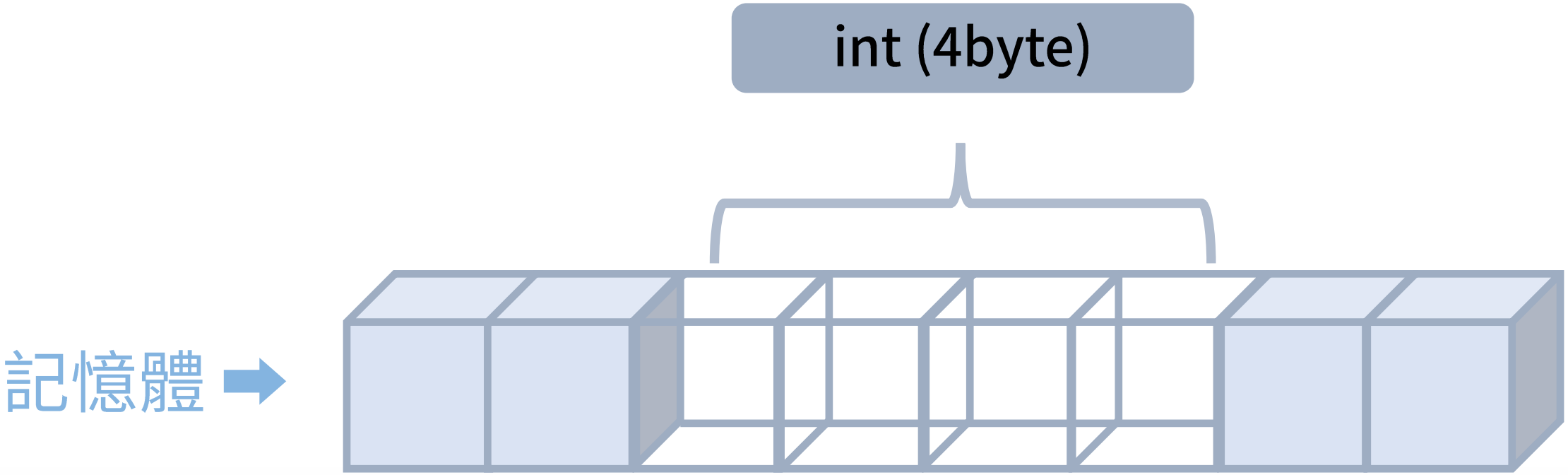
End of line

```
#include<iostream>
using namespace std;

int main(){

    cout<<"hello"<<endl<<"My name is Shark";

}
```

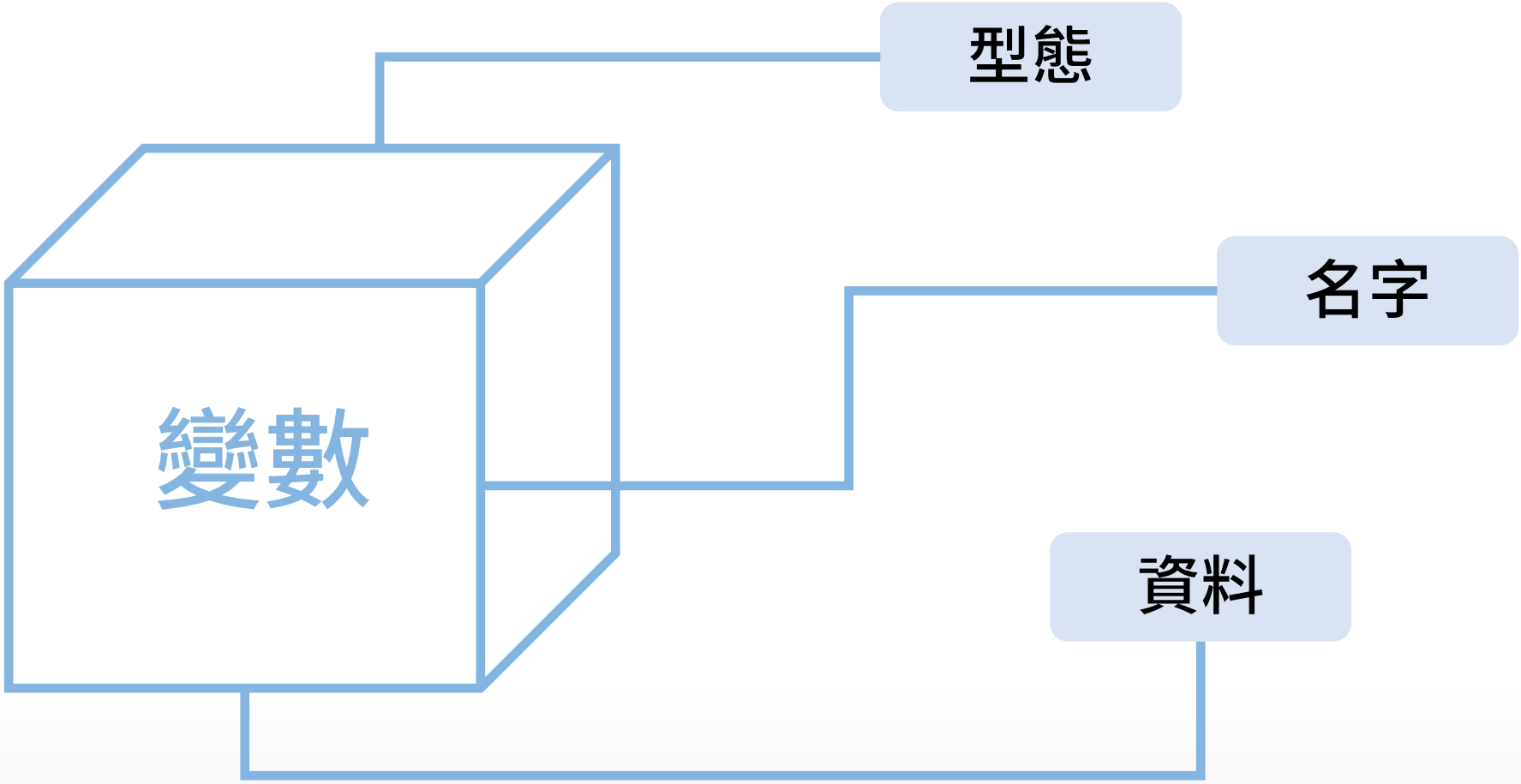
Variable

/ 變數和記憶體有關喔~

Introduction



Introduction



Declare

```
int number = 2654
```

型態　名字　資料

Try

```
#include<iostream>
using namespace std;

int main(){

    int number = 0;
    cin>>number;
    cout<<number;

}
```

Logic Operation

/ 你是否同意廢除電業法第95條第1項，即廢除「核能發電設備應於中華民國一百十四年以前，全部停止運轉」之條文？

簡單介紹

AND	A	B	Y
	0	0	0
	1	0	0
	0	1	0
	1	1	1

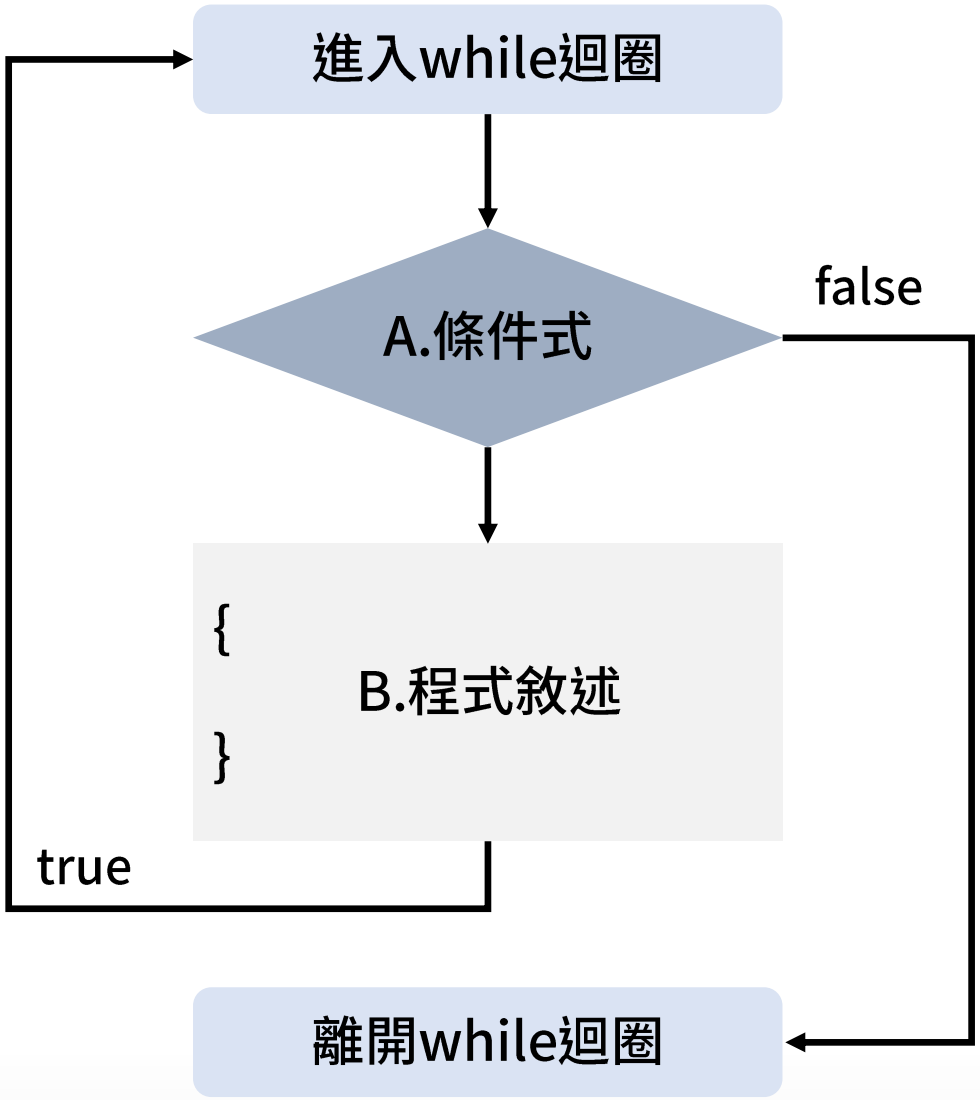
OR	A	B	Y
	0	0	0
	1	0	1
	0	1	1
	1	1	1

Loop

/ 機器對於人的用途是什麼？

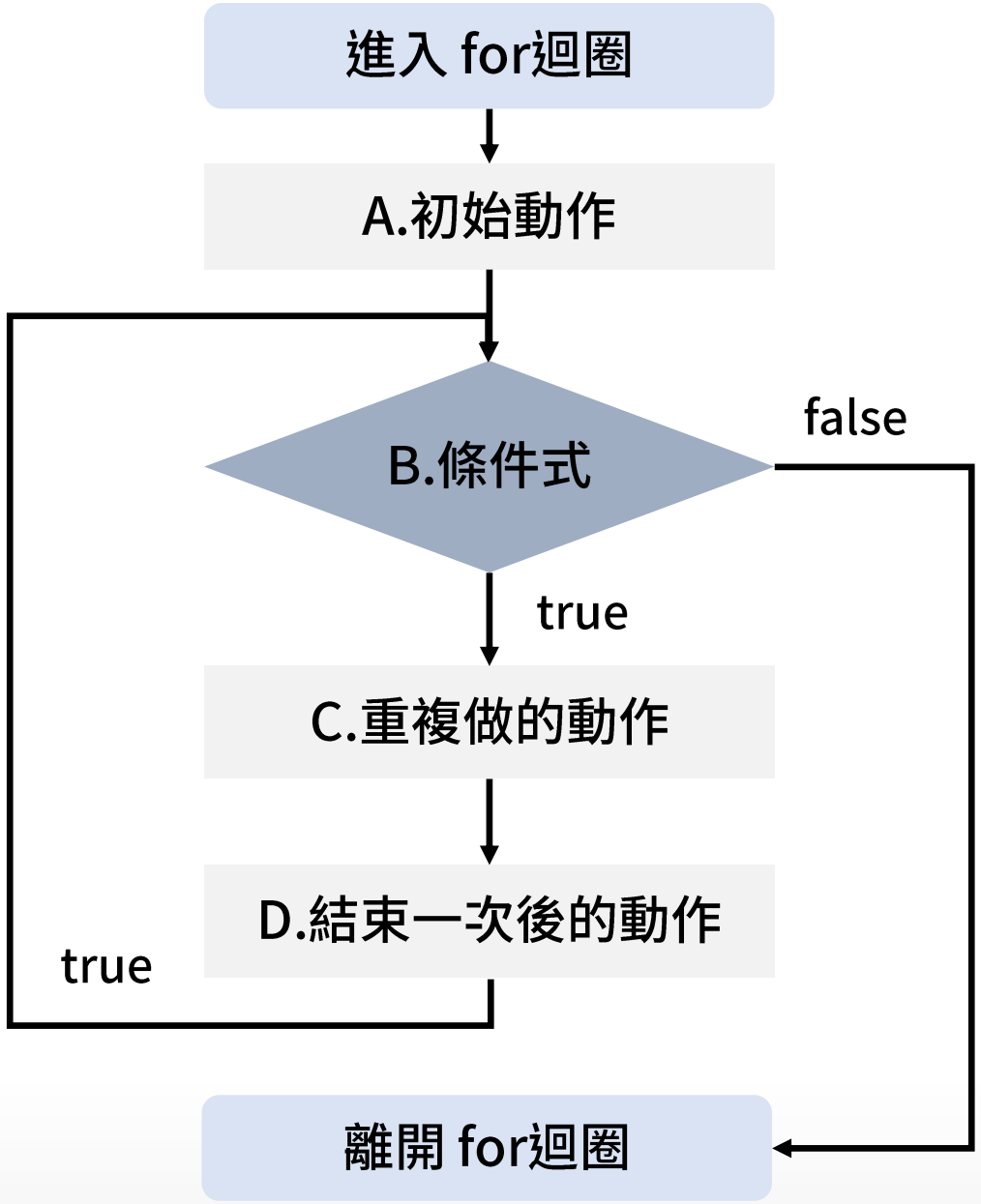
程式語法

```
while( A.條件式 )
{
    B. 重覆做的事...
}
```



程式語法

```
for( A.初始 ; B.條件 ; D.每次結束 )
{
    C.重覆做的動作...
}
```



Array

/ 陣列每個人都學過，但你真的懂嗎？

一維陣列

Int A[3] = {96 , 45 , 24 }

	A[0]				A[1]				A[2]				
0X68	0X69	0X70	0X7A	0X7B	0X7C	0X7D	0X7E	0X80	0X81	0X82	0X83	0X84	0X85
…	96				45				24				…

4 byte

陣列索引

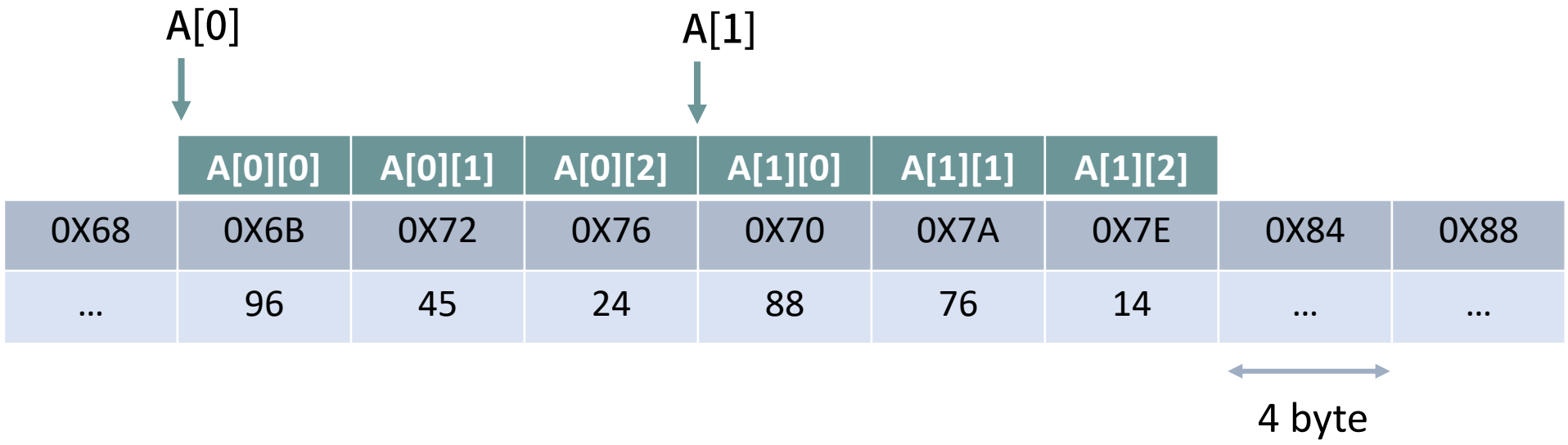
A[3] = {96 , 45 , 24 }

	A[0]				A[1]				A[2]				
	0X69 + 0*4byte				0X7C + 1*4byte				0X81 + 2*4byte				
0X68	0X69	0X70	0X7A	0X7B	0X7C	0X7D	0X7E	0X80	0X81	0X82	0X83	0X84	0X85
…	96				45				24				…

4 byte

二維陣列

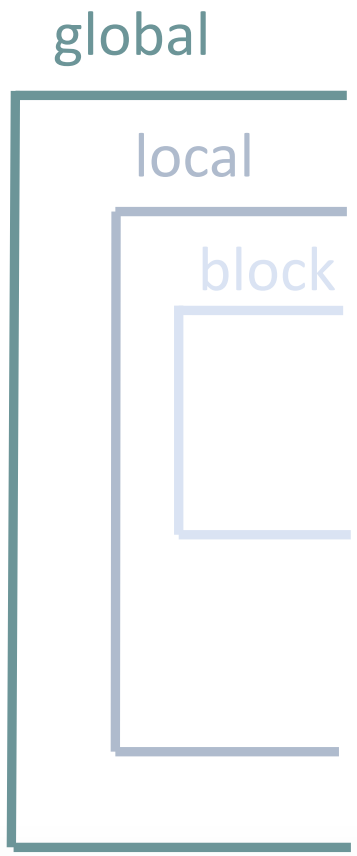
A[2][3] = { { 96 , 45 , 24 } , { 88 , 76 , 14 } } = { 96 , 45 , 24 , 88 , 76 , 14 }



Scope

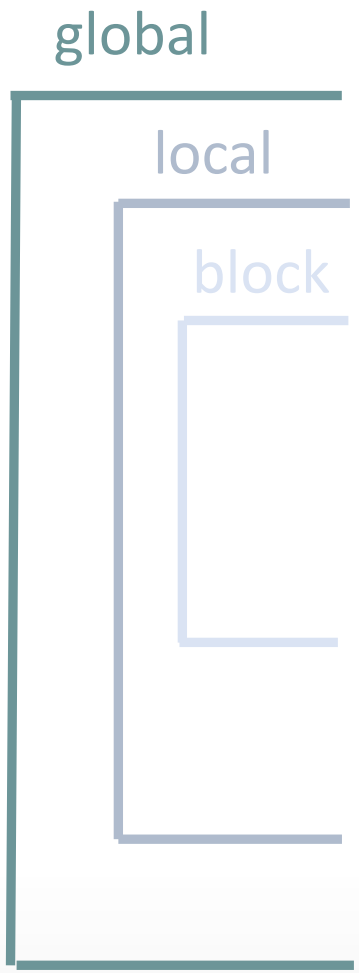
/ 人的生命轉瞬即逝，要好好把握

Basic Scope




```
int a = 0 ;
void  matchZero ( int  b ) {
    if ( b == 0 ) {
        int c = 0
}
    cout << c;
}
//
```

Test




```
int a = 0 ;
void  matchZero ( int  a ) {
    if ( a == 0 ) {
        int  a = 1
        cout << a;
    }
    cout << a;
}
cout << a;
```

Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
  cout << a;
}
```

int main () a=0

Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
    cout << a;
}
```

If..	i=0
int main ()	a=0

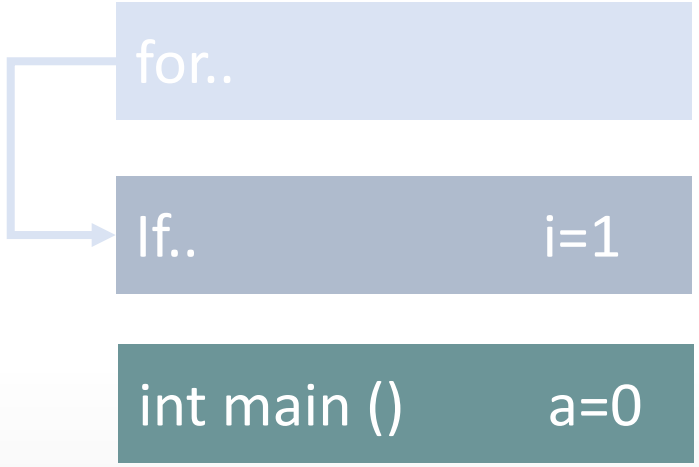
Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
  cout << a;
}
```

If..	i=1
int main ()	a=0

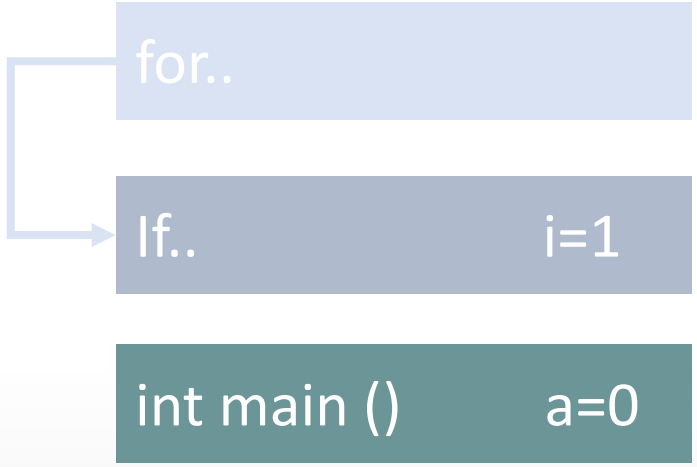
Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
    cout << a;
}
```



Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
    cout << a;
}
```



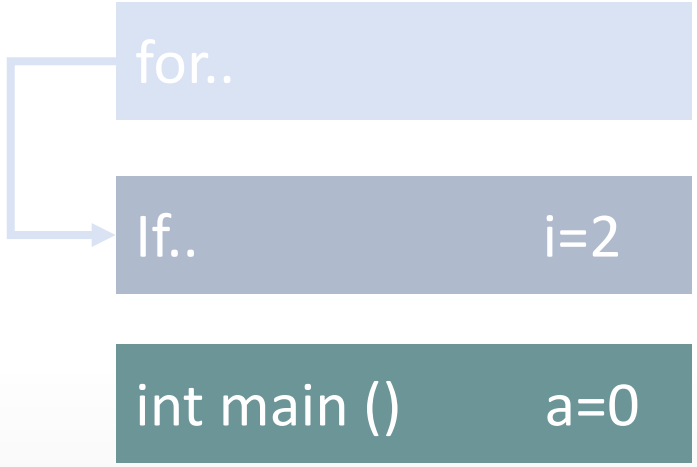
Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
    cout << a;
}
```

for..	b=3
If..	i=1
int main ()	a=0

Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
    cout << a;
}
```



Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
  cout << a;
}
```

If..	i=2
int main ()	a=0

Calling Stack

```
int main() {
    int a = 0 ;
    if ( a == 0 ) {
        int i = 0 ;
        for (; i < 2 ; i++)
            int b = 3 ;
    }
  cout << a;
}
```

If..	i=0
int main ()	a=0

Recursion

/ 使用相同的方法，解決重複性的問題

呼叫方式

① 呼叫A

```
void A(){
    ...
    ...
    A();
}
```

① 呼叫B

```
void A(){
    ...
    B();
    ...
}

void B(){
    ...
    A();
    ...
}
```

② 呼叫A

基本規則

規則

① 需要有終點

② 每次呼叫都要讓問題變更小

③ 不能有循環出現

```
void f(int n){
    return f(n)+f(n);
}
```

小試身手

factorial(階乘)

① 請用遞迴的方式達成

② 可以試著畫圖看看

```
int factorial(int $n) {

    4! = 4 * 3!
    3! = 3 * 2!
    2! = 2 * 1!

    .....

}
```

小試身手

factorial(階乘)

① n! = n * (n-1) !

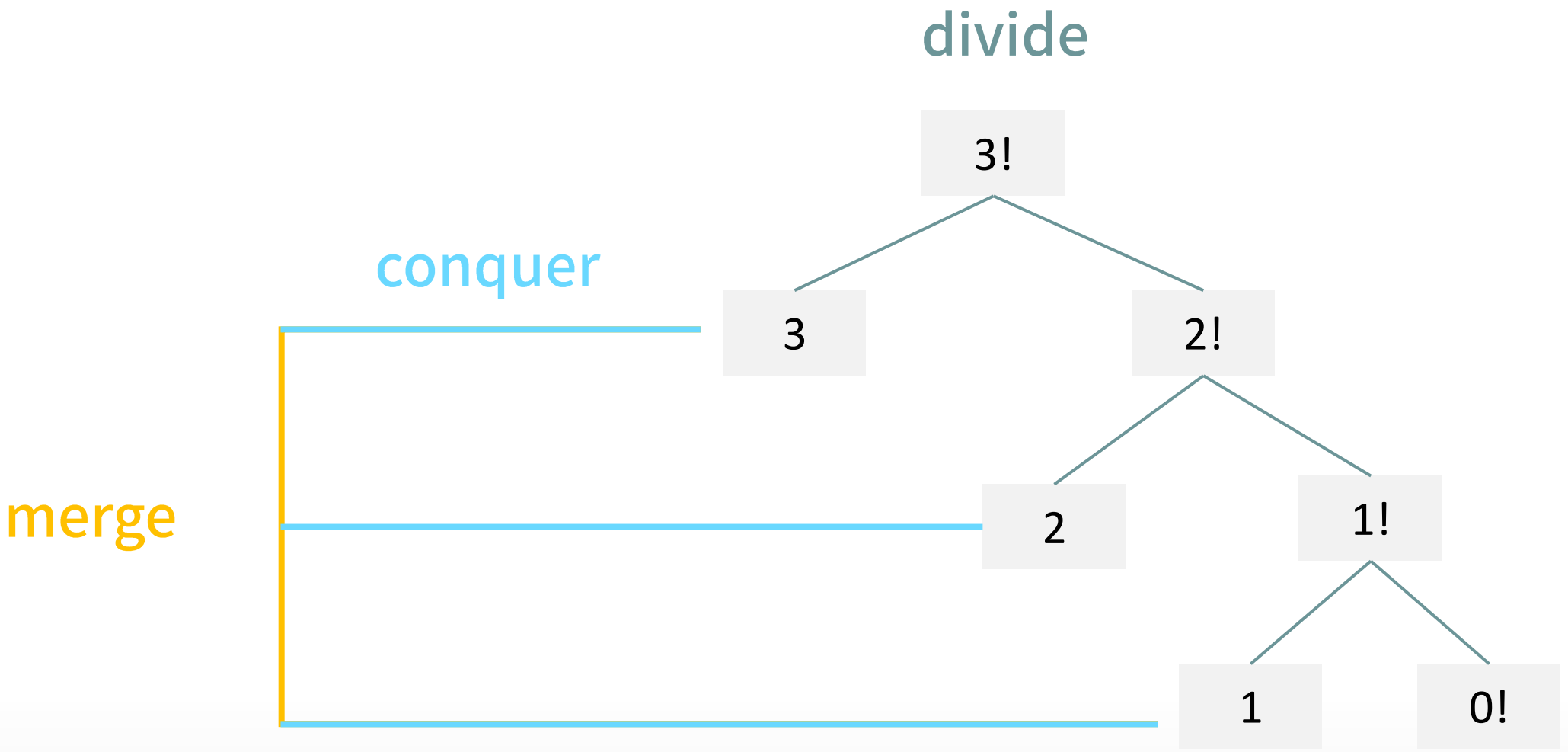
② 0! = 1

```
int factorial(int $n) {

    if ($n == 0)
        return 1;

    return $n * factorial($n - 1);

}
```

Divide and Conquer

/ 分而治之，各個擊破

基本步驟



什麼情況

問題特徵

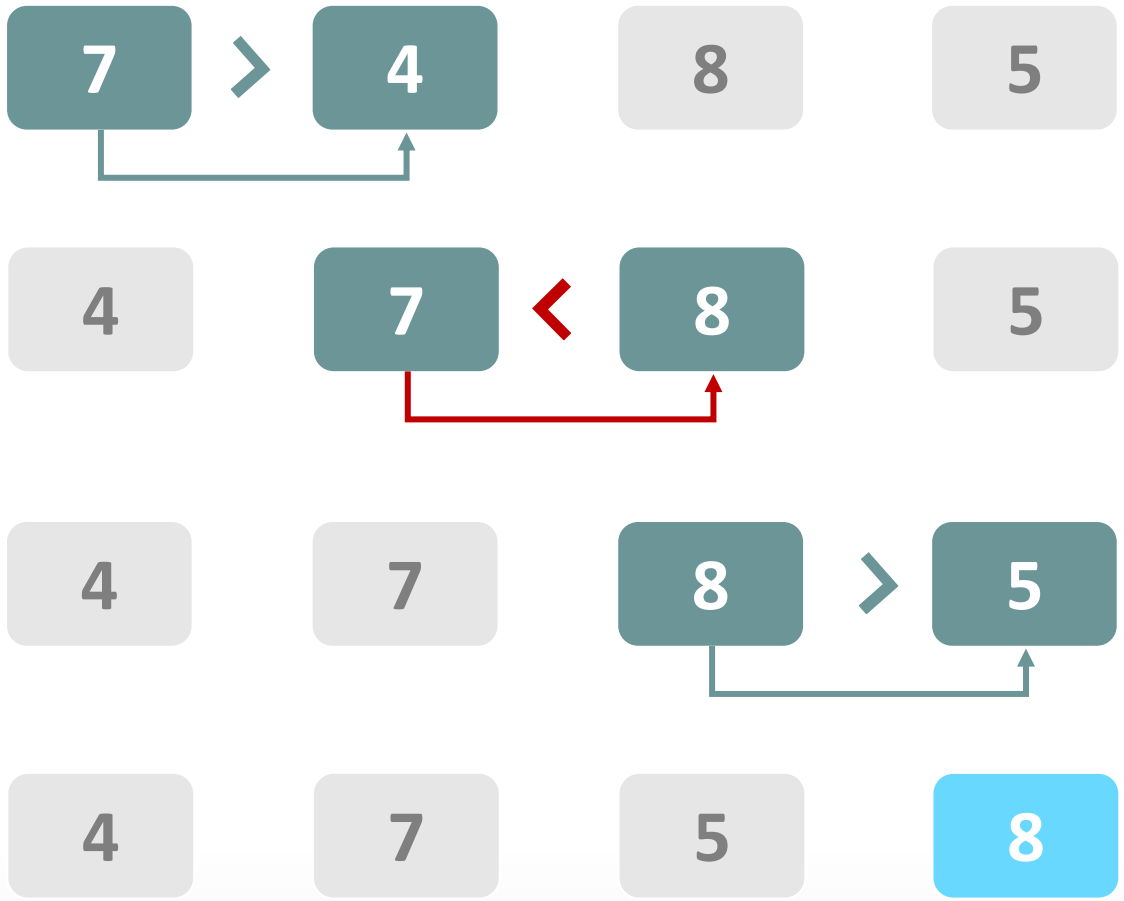
① 縮小到一定程度就可以被解決

② 可以被分解成相同的子問題

③ 子問題的解可以合併成該問題的解

④ 子問題之間是互相獨立的

Bubble Sort



Quick Sort