struct

陣列:把很多型態一樣的東西連接起來

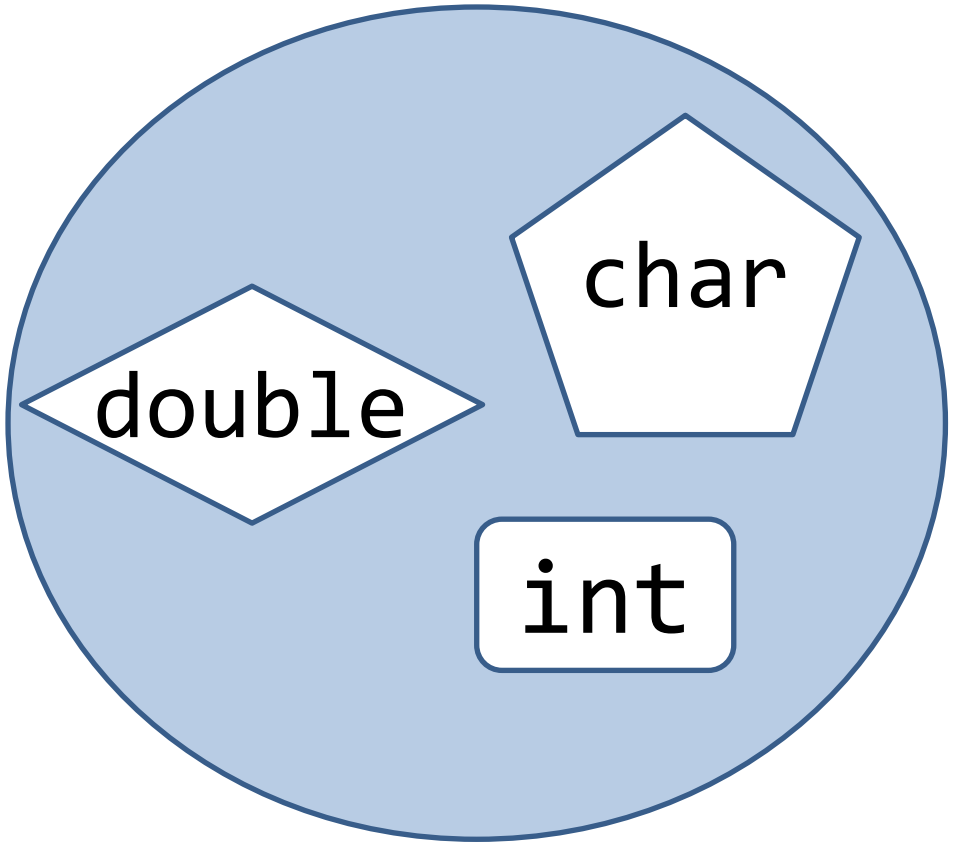
```
ex: int hour;
    int minute;
    int second;

int time[3];
```

Q:如果我想把不同型態的東西放在一起怎麼辦?

```
ex: int i;
    char c;
    double d;
```

在C++裡,我們可以定義一個新的型態,
這個新型態裡可以包含不同種類的東西



怎麼定義?

語法規定 ex: `int arr[10];`

`int function(double a);`

在新型態前加上struct

```
struct MyTypeName{
    //member1
    //member2
    ...
};
```

struct

struct

```
struct house {
    double volume;
    int numOfWindow;
    char doorColor[10];
};
```

struct

```
struct house {
    double volume;
    int numOfWindow;
    char doorColor[10];
};
int main(){
    house h1;    //h1 is a type name
    return 0;
}
```

struct

不能直接在struct裡給值

只能在main或其他函數裡用.的方式給值

struct

```
ex: h1.volume = 234.5;
    h1.numOfWindow = 5;
    h1.doorColor = "brown";
```

struct

```
ex: h1.volume = 234.5;
    h1.numOfWindow = 5;
    h1.doorColor = "brown";
    strcpy(h1.doorColor, "brown");
```

struct summary

宣告:

```
struct MyTypeName{
    //member1
    //member2
    ...
};
```

使用:

```
MyTypeName variableName;
variableName.member? = ???;
```

```
struct house {
    double volume;
    int numOfWindow;
    char doorColor[10];
};
int main () {
    house h1;
    house h2;
    ...
    house h10;
    return 0;
}
```

```
struct house {
    double volume;
    int numOfWindow;
    char doorColor[10];
};
int main () {
    house h[10];
    return 0;
}
```

```
int main(){
    house h1, h2;
    h1.??? = ???;
    ...
    h2.??? = ???;
    ...
    h1 = h2;
    //h1.??? = h2.???
    //...
    return 0;
}
```

Practice #235

sorting(排序)

簡單的兩種排序法:

插入排序法(insertion sort)

選擇排序法(selection sort)

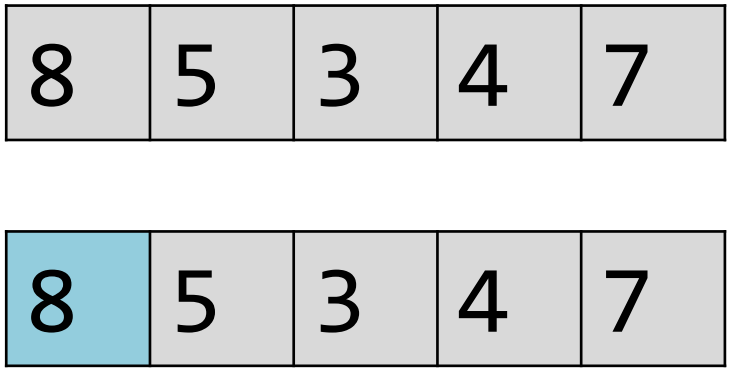
insertion sort

已排序

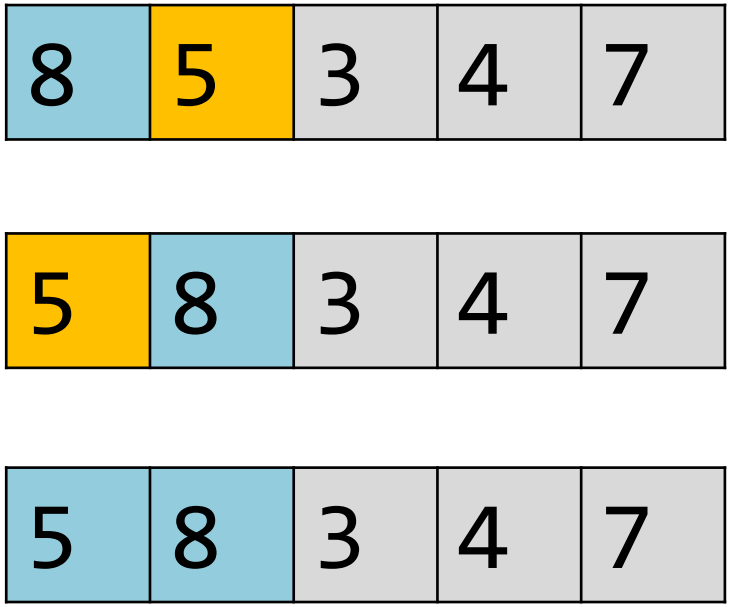
排序中

未排序

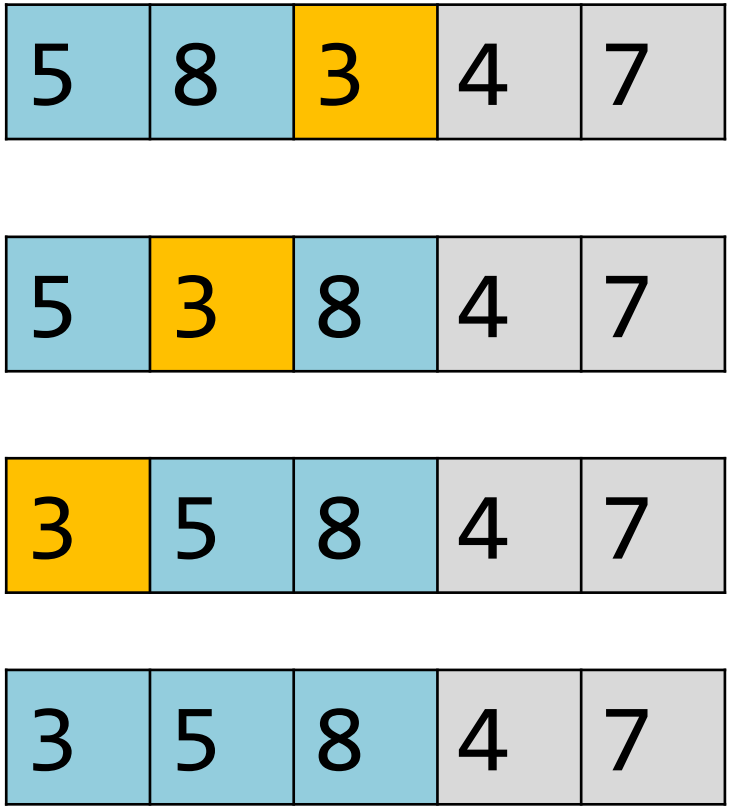
insertion sort



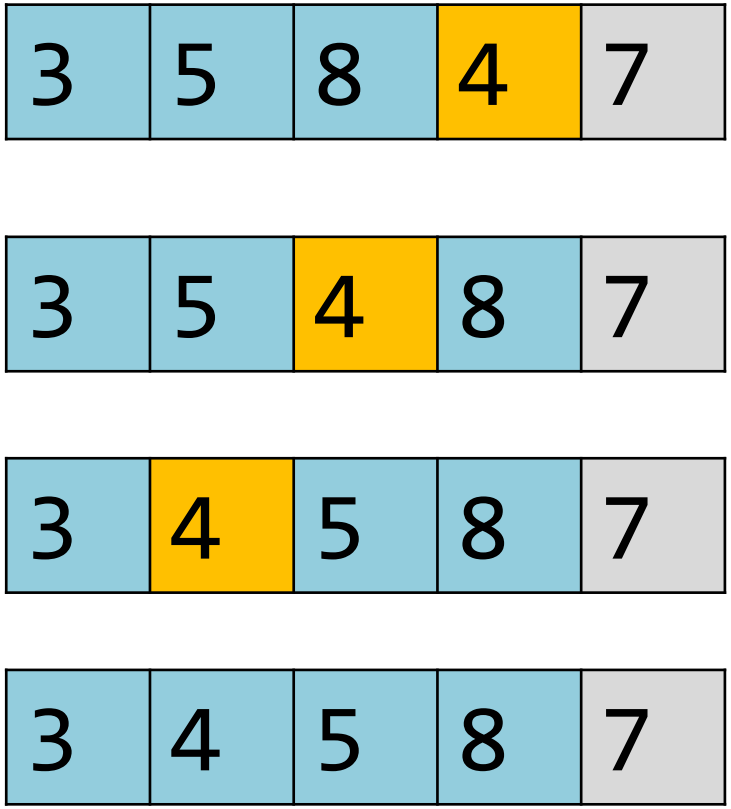
insertion sort



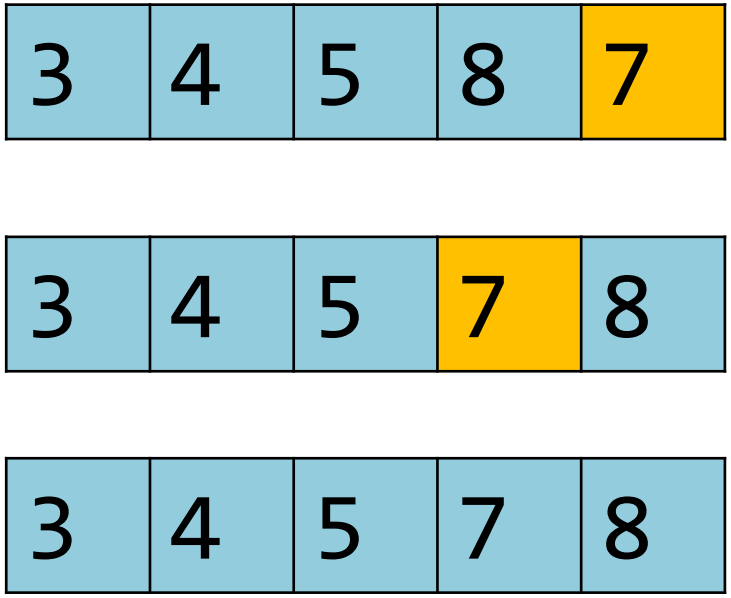
insertion sort



insertion sort



insertion sort



insertion sort

- Video

insertion sort

```
void insertion_sort(int a[], int size) {
    for (int i = 0; i < size; i++) {
        int temp = a[i];
        for (int j = i - 1; j > -1; j--) {
            if (temp > a[j])
                break;
            else if (temp <= a[j]) {
                a[j+1] = a[j];
                a[j] = temp;
            }
        }
    }
}
```

Practice

第一行輸入一個正整數n(1<n<100),第二行輸入n個數字,利用insertion sort排序,輸出排序後的結果

Sample Input:

8

12 6 5 10 9 7 14 3

Sample Output:

3 5 4 7 9 10 12 14

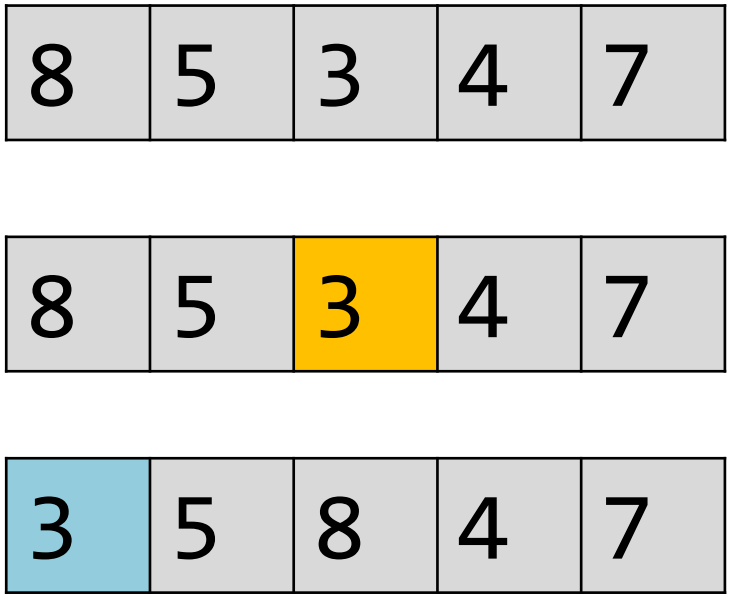
selection sort

已排序

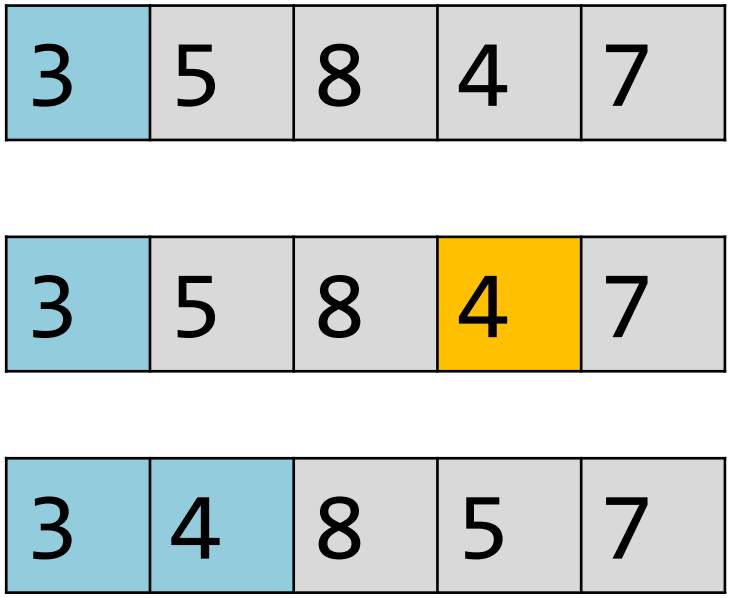
排序中

未排序

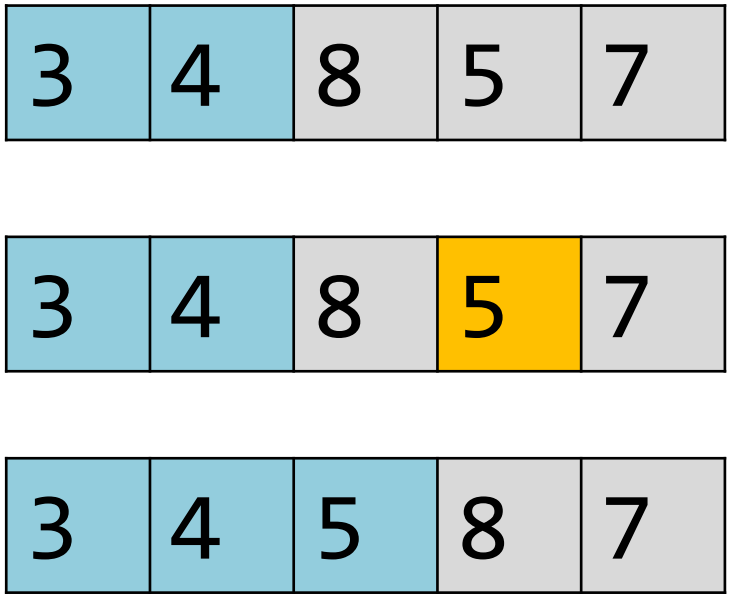
selection sort



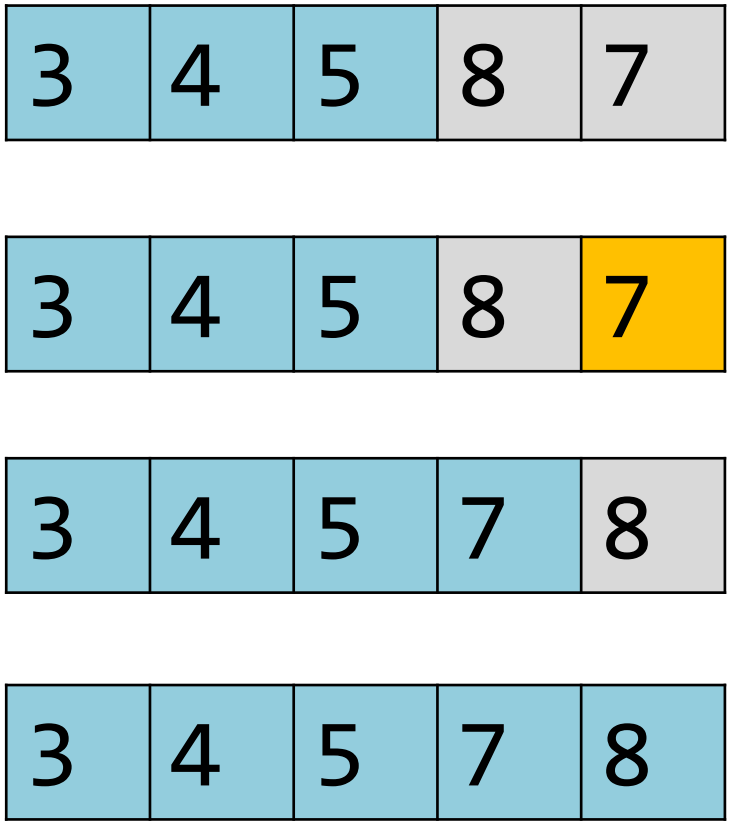
selection sort



selection sort



selection sort



selection sort

- Video

selection sort

```
void selection_sort(int a[], int size) {
    for (int i = 0; i < size - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < size; j++)
            if (a[j] < a[minIndex])
                minIndex = j;
    int temp = a[minIndex];
    a[minIndex] = a[i];
    a[i] = temp;
    }
}
```

Practice

第一行輸入一個正整數n(1<n<27),第二行輸入n個英文字母,利用selection sort排序,輸出排序後的結果

Sample Input:

8

c g s x k d a e

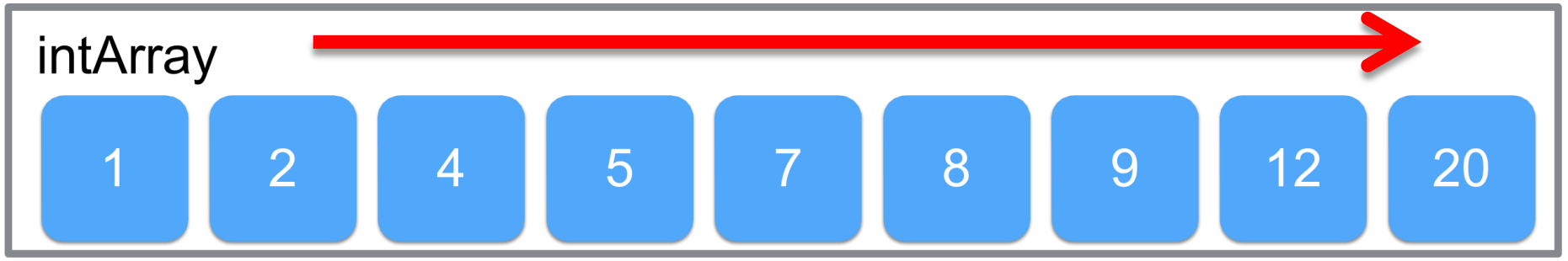
Sample Output:

a c d e g k s x

二分搜尋
binary search

資訊之芽/語法班
Gin

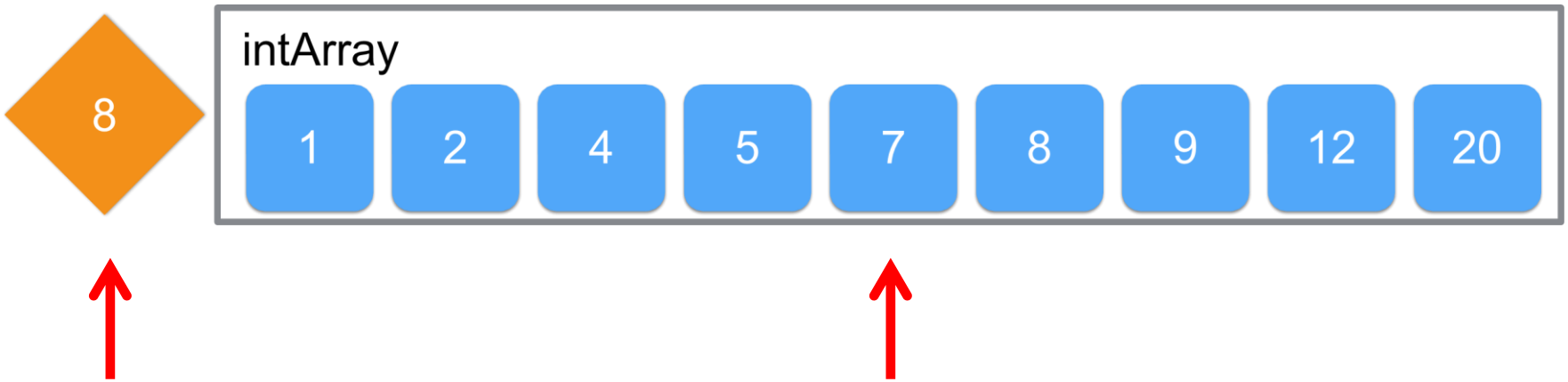
線性搜尋(linear search)



- 從頭掃到尾,速度很慢
- 沒有善加利用到已經排序的性質

二分搜尋(binary search)

- 又稱二元搜尋
- 把它切半搜,再切半搜,再切半搜,...,找到了!
- 使用這個方法搜尋,陣列一定要經過排序!
  (由小到大或由大到小)



我們要找的　　有9個元素,且經過排序的陣列


中間(mid)
8
intArray
1
2
4
5
7
8
9
12
20
7 < 8 所以會在 mid 的右半邊！
8
intArray
1
2
4
5
7
8
9
12
20


中間(mid)
8
intArray
1
2
4
5
7
8
9
12
20
8 < 9 所以會在mid 的左半邊！
8
intArray
1
2
4
5
7
8
9
12
20

8 == 8,找到8了,回傳它在陣列中的位置 (索引值)

return 5;

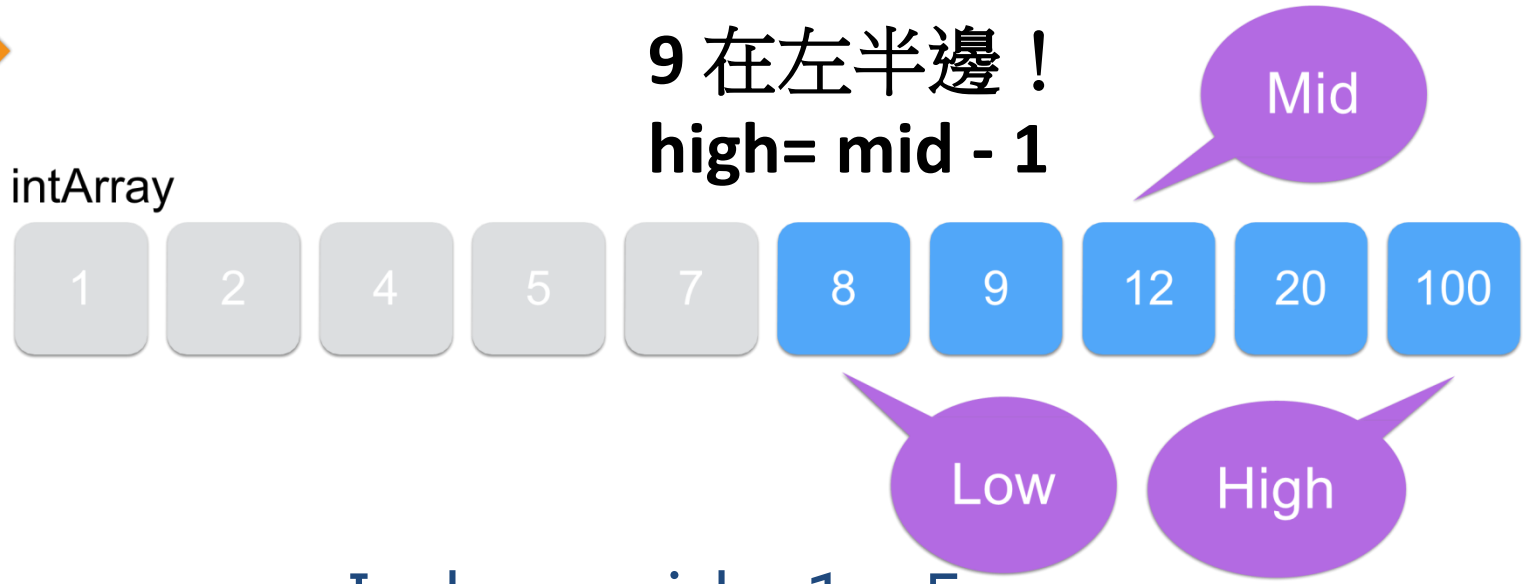
影片範例：http://ppt.cc/gh4kf


9
Index : (0 + 9) / 2 = 4
Mid
intArray
1
2
4
5
7
8
9
12
20
100
Low
High
9 在右半邊！
low = mid + 1
Index : 0
Index : 9

Index : (5 + 9) / 2 = 7

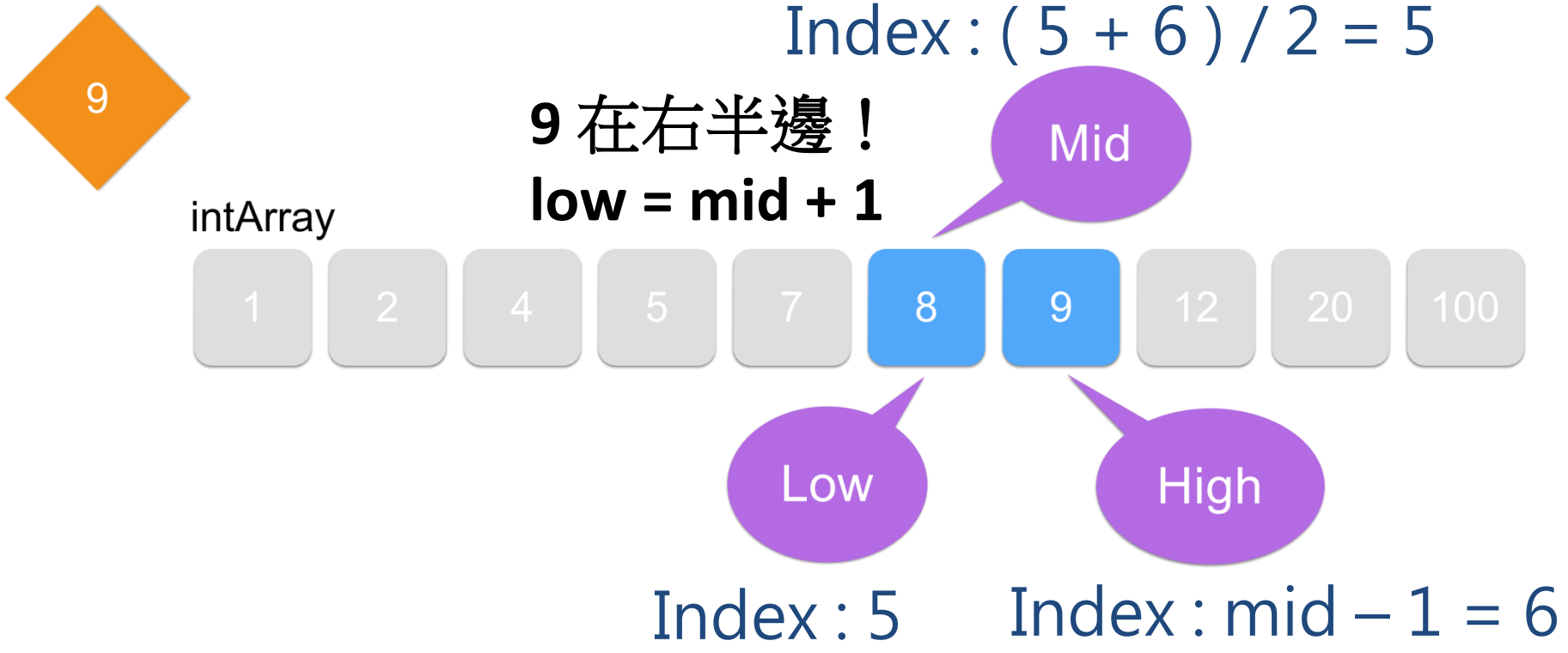
9

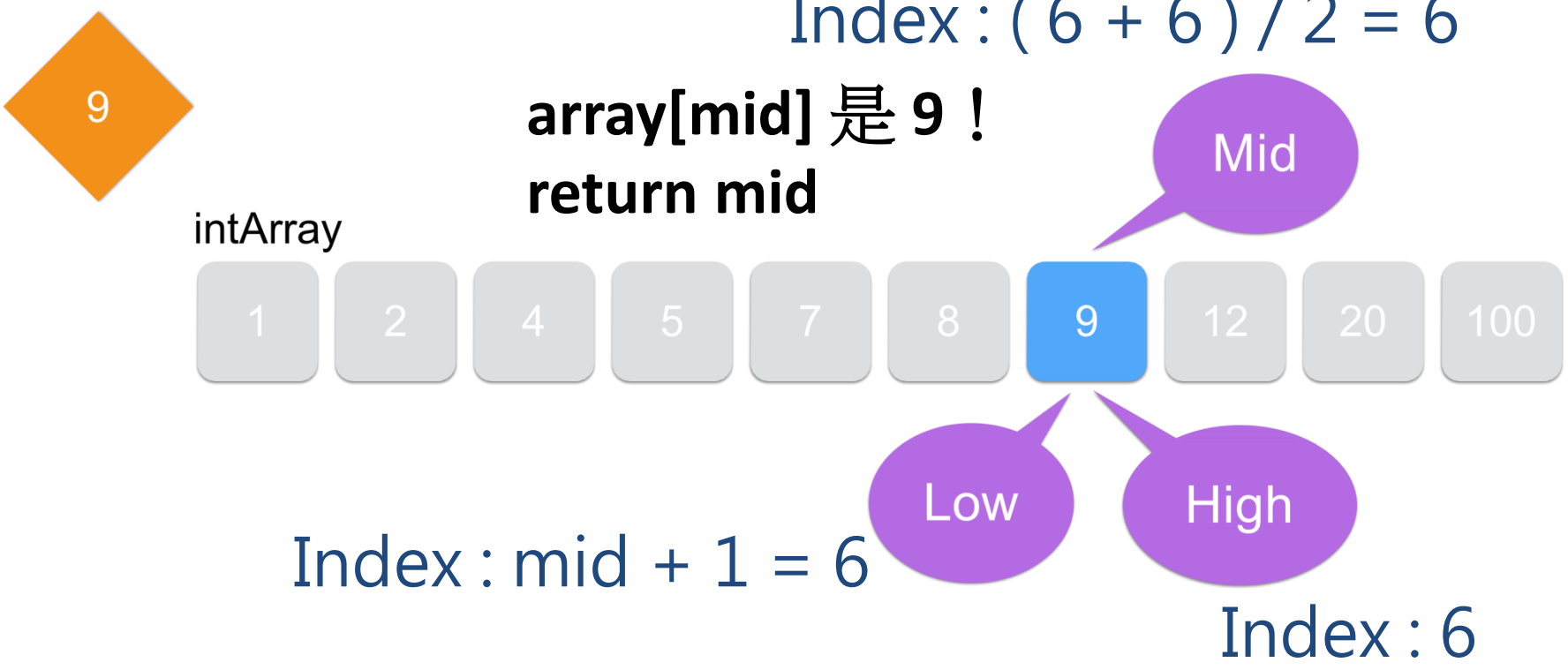
9 在左半邊！
high= mid - 1



Index : mid+ 1= 5

Index : 9


9
Index : (5 + 6) / 2 = 5
9 在右半邊！
low = mid + 1
Mid
intArray
1
2
4
5
7
8
9
12
20
100
Low
High
Index : 5
Index : mid – 1 = 6


9
Index : (6 + 6) / 2 = 6
array[mid] 是 9 !
return mid
Mid
intArray
1
2
4
5
7
8
9
12
20
100
Low
High
Index : mid + 1 = 6
Index : 6

```
int binarySearch(int array[], int length, int key)
{
  int low = 0;
  int high = length - 1;
  while(low <= high) {
    int mid = (low + high) / 2

    1. array[mid] = key : 找到了，回傳 mid
    2. array[mid] > key : 在左半邊，把 high 變成 mid - 1
    3. array[mid] < key : 在右半邊，把 low 變成 mid + 1
  }
  如果跑到這裡，也就是while迴圈外，代表沒找到這個數。
}
```

Homework

#236