

# C++資料結構與程式設計

排序與搜尋

NTU CSIE

# Outline

## 排序

- 泡沫排序法 ( Bubble Sort )
- 選擇排序法 ( Selection Sort )
- 插入排序法 ( Insertion Sort )

## 搜尋

- 循序搜尋法 ( Linear Search )
- 二分搜尋法 ( Binary Search )

# 排序 (Sorting)

## ▶ 簡介

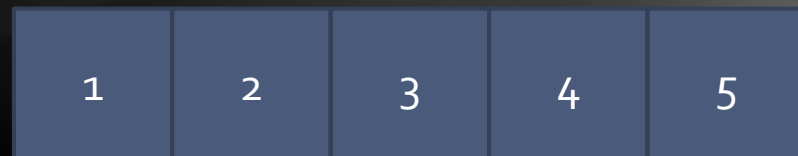
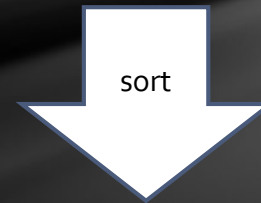
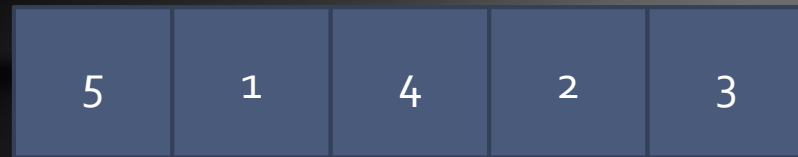
- 將一群資料按照某一規則排列，使其具有遞增（減）的關係

## ▶ 分類與比較

- 執行效率
- 記憶體空間

## ▶ 用途

- 資料搜尋
- 進階的分析與處理



# 泡沫排序法 ( Bubble Sort )

## ▶ 簡介

- 將相鄰的資料兩兩比較大小，決定是否交換

## ▶ 範例程式碼 ( 將資料由小排到大 )

```
#include <stdio.h>
void swap(int *a, int *b)
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;
}
void print(int n, int *p)
{
    int i;

    for(i=0; i<n; i++)
    {
        printf("%d ", p[i]);
    }
    printf("\n");
}
```

```
int main()
{
    int data[5] = {34,12,5,66,1}; // 欲排序的資料
    int i, j;
    int n=5;

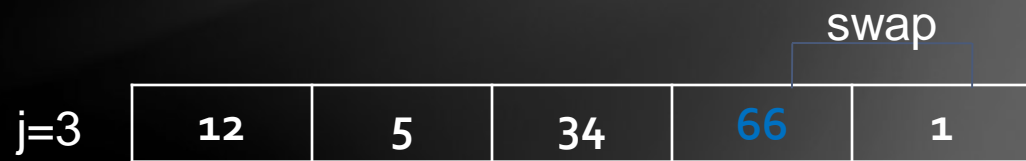
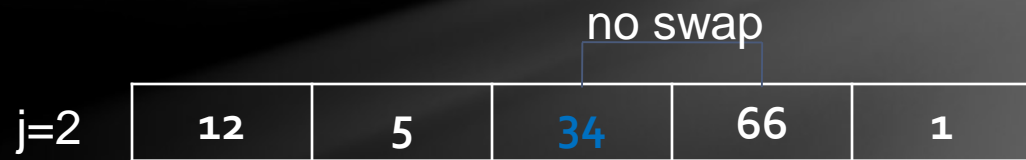
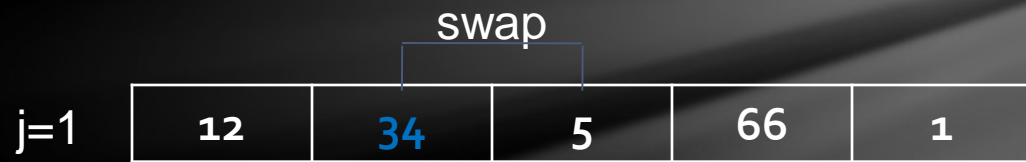
    for(i=n; i>1; i--)
    {
        for(j=0; j<i-1; j++)
        {
            if(data[j+1] < data[j])
            {
                swap(&data[j+1], &data[j]);
            }
        }
    }
    print(n, data);
    return 0;
}
```

# 泡沫排序法 ( Bubble Sort )

使用一迴圈將最大值換到陣列最後

如果有 $n$ 個數要判斷 $n-1$ 組數值

- 5個數要判斷4次



# 泡沫排序法 ( Bubble Sort )

再使用一迴圈連續對陣列把**最大值**放到最後

如果有 $n$ 個數要做 $n-1$ 次

- 5個數要做4次

i=5	34	12	5	66	1
-----	----	----	---	----	---

i=4	12	5	34	1	66
-----	----	---	----	---	----

i=3	5	12	1	34	66
-----	---	----	---	----	----

i=2	5	1	12	34	66
-----	---	---	----	----	----

1	5	12	34	66
---	---	----	----	----

# 選擇排序法 ( Selection Sort )

## ▶ 簡介

- 在一段資料中找出最大 ( 小 ) 值後，才做交換。

## ▶ 範例程式碼 ( 將資料由小排到大 )

```
#include <stdio.h>
void swap(int *a, int *b)
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;
}
void print(int n, int *p)
{
    int i;

    for(i=0; i<n; i++)
    {
        printf("%d ", p[i]);
    }
    printf("\n");
}
```

```
int main()
{
    int data[5] = {34,12,5,66,1}; // 欲排序的資料
    int i, j, pos; // pos: 紀錄目前最小值位置
    int n=5;
    for(i=0; i<n-1; i++)
    {
        pos = i;
        for(j=i+1; j<n; j++) // 找出最小值
        {
            if(data[j] < data[pos])
                pos = j;
        }
        // 把最小值跟第 i 個做交換
        swap(&data[i], &data[pos]);
    }
    print(n, data);
    return 0;
}
```

# 選擇排序法 ( Selection Sort )

使用一迴圈將**最小值**換到陣列**最前**

如果有**n**個數要判斷**n-1**個數值

- 5個數要判斷4次

- **只需交換一次**

	Index 0	1	2	3	4	最小值位置(pos)
						0
j=1	34	12	5	66	1	1
j=2	34	12	5	66	1	2
j=3	34	12	5	66	1	2
j=4	34	12	5	66	1	4
	1	12	5	66	34	4



# 選擇排序法 ( Selection Sort )

再使用一迴圈連續對陣列把**最小值**放到第**i**個位置

如果有n個數要做n-1次

- 5個數要做4次

	Index 0	1	2	3	4
i=0	34	12	5	66	1
i=1	1	12	5	66	34
i=2	1	5	12	66	34
i=3	1	5	12	66	34
	1	5	12	34	66

# 插入排序法 ( Insertion Sort )

## ▶ 簡介

- 將一段資料中最右(左)邊的資料當作key，然後往左(右)塞入此資料中作排序。

## ▶ 範例程式碼 ( 將資料由小排到大 )

```
#include <stdio.h>
void print(int n, int *p)
{
    int i;

    for(i=0; i<n; i++)
    {
        printf("%d ", p[i]);
    }
    printf("\n");
}
```

```
int main()
{
    int data[5] = {34,12,5,66,1}; // 欲排序的資料
    int i, j;
    int key; // 紀錄要被插入的值
    int n=5;
    for( i=1; i<n; i++)
    {
        key=data[i];
        for(j=i-1; j>=0 && data[j]>key; j--)
        {
            data[j+1] = data[j];
        }
        data[j+1] = key; //將key插入
    }
    print(n, data);
    return 0;
}
```

# 插入排序法 ( Insertion Sort )

使用一迴圈將key插到適當的位置

如果有n個數要判斷n-1個數值

- 2個數要判斷1次
- 不一定需要交換

```
key=data[i];  
for(j=i-1; j>=0 && data[j]>key; j--) {  
    data[j+1] = data[j];  
}  
data[j+1] = key; //將key插入
```

要被插入  
的值(key)

j=0	34	12	5	66	1
-----	----	----	---	----	---

# 插入排序法 ( Insertion Sort )

再使用一迴圈連續對陣列的值做插入

如果有 $n$ 個數要做 $n-1$ 次

- 5個數要做4次

key

i=1	34	12	5	66	1
-----	----	----	---	----	---

key

i=2	12	34	5	66	1
-----	----	----	---	----	---

key

i=3	5	12	34	66	1
-----	---	----	----	----	---

key

i=4	5	12	34	66	1
-----	---	----	----	----	---

1	5	12	34	66
---	---	----	----	----

# 小練習

(chap03\_ex1\_sort\_num.c)

寫一個讓使用者輸入之介面，功能如下

- (1) 泡沫排序法
- (2) 選擇排序法
- (3) 插入排序法

讓使用者任意輸入6個數字，將數字“由小到大”做排序並將每次排序的結果輸出。

# 回家作業 (chap03\_ex2\_sort\_char.c)

寫一個讓使用者輸入之介面，功能如下

- (1) 泡沫排序法
- (2) 選擇排序法
- (3) 插入排序法

讓使用者任意輸入6個字串(字元陣列長度最大128)，將字串”由英文字母順序大到小”做排序並將每次排序的結果輸出。

提示: 使用strcmp, strcpy來實作 (**string.h** )

strcmp(data[0], data[0])

第一個字串大於第二個字串回傳正值，反之回傳負值。相等則為0

提示：char data[10][128]; // 欲排序的資料

傳進函式：void func(char p[][128]){printf("%s ", p[0]);}

# Visual C++ 的問題

```
'strcpy': This function or variable may be unsafe.  
Consider using strcpy_s instead. To disable deprecation,  
use _CRT_SECURE_NO_WARNINGS. See online help for details.
```

在 ProjectProperties -> Configuration Properties -> C/C++ -> Preprocessor -> Preprocessor Definitions

加入這兩行

\_CRT\_SECURE\_NO\_DEPRECATED

\_CRT\_NONSTDC\_NO\_DEPRECATED

# 大綱

## 排序

- 泡沫排序法 ( Bubble Sort )
- 選擇排序法 ( Selection Sort )
- 插入排序法 ( Insertion Sort )

## 搜尋

- 循序搜尋法 ( Linear Search )
- 二分搜尋法 ( Binary Search )



# 循序搜尋法 ( Linear Search )

## 簡介

- 在一群資料中，從頭搜尋到尾直到找到資料為止。

## 範例程式碼

```
int LinearSearch(int n, int *p, int value)
{
    int i;
    for(i=0; i< n; i++)
    {
        if(p[i] == value)
        {
            return i; // 找到: 傳回資料位置
        }
    }
    return -1; // 找不到: 回傳-1
}
```

# 循序搜尋法 ( Linear Search )

使用迴圈一個一個找資料存放之位置

- 資料越多找越久...
- 資料放越後面找越久...

data	<i>i</i>	<u>34</u>	12	5	66	1
		[0]	[1]	[2]	[3]	[4]
			<i>i</i>			
		<u>34</u>	<u>12</u>	5	66	1
		[0]	[1]	[2]	[3]	[4]
data				<i>i</i>		
		<u>34</u>	<u>12</u>	<u>5</u>	66	1
		[0]	[1]	[2]	[3]	[4]
					<i>i</i>	
		<u>34</u>	<u>12</u>	<u>5</u>	<u>66</u>	1
		[0]	[1]	[2]	[3]	[4]
data						<i>i</i>
		<u>34</u>	<u>12</u>	<u>5</u>	66	<u>1</u>
		[0]	[1]	[2]	[3]	[4]
		<u>34</u>	<u>12</u>	<u>5</u>	<u>66</u>	<u>1</u>
		[0]	[1]	[2]	[3]	[4]

要找的值

<u>1</u>
value

# 二元搜尋法 ( Binary Search )

## 簡介

- 對一群排序過的資料，使用二分法的方式做搜尋
- 範例程式碼(在一群資料中找出變數key的索引值)

```
int BinarySearch(int n, int *p, int value)
{
    int low=0, high=n, mid;
    while(low <= high)
    {
        mid = (low + high) / 2;
        if(p[mid] > value)
            high = mid - 1;
        else if(p[mid] < value)
            low = mid+1;
        else
            return mid;
    }
    return -1;
}
```

# 二元搜尋法 ( Binary Search )

## 使用二分法找資料

- 必須確保資料有經過排序

if(p[mid] < value)

low = mid+1;

要找的值

34

value

	low		mid		
data	1	5	<u>12</u>	34	66
	[0]	[1]	[2]	[3]	[4]

mid = (low + high) / 2;

data			low	mid	high
	1	5	12	34	<u>66</u>
	[0]	[1]	[2]	[3]	[4]

if(p[mid] > value)

high = mid - 1;

data			low	mid	high
	1	5	12	<u>34</u>	66
	[0]	[1]	[2]	[3]	[4]

# 小練習

(chap03\_ex3\_search\_num.c)

寫一個讓使用者輸入之介面，功能如下

- (1) 插入一**整數**於陣列(提示: 使用**插入排序法**)
- (2) 尋找一**整數**並印出 (使用**線性搜尋法**)
- (3) 尋找一**整數**並印出 (使用**二元搜尋法**)
- (4) 印出目前資料(由小到大)

# 回家作業

(chap03\_ex4\_search\_char.c)

將上一程式之輸入資料改為字串(字元陣列長度最大128)

提示：char data[10][128]; // 欲排序的資料

傳進函式：

```
void func(char p[][128])  
{  
    printf("%s ", p[0]);  
}
```