



2018

程式設計
沒有加強班

程式設計與實習(二)

BY 孫茂勛

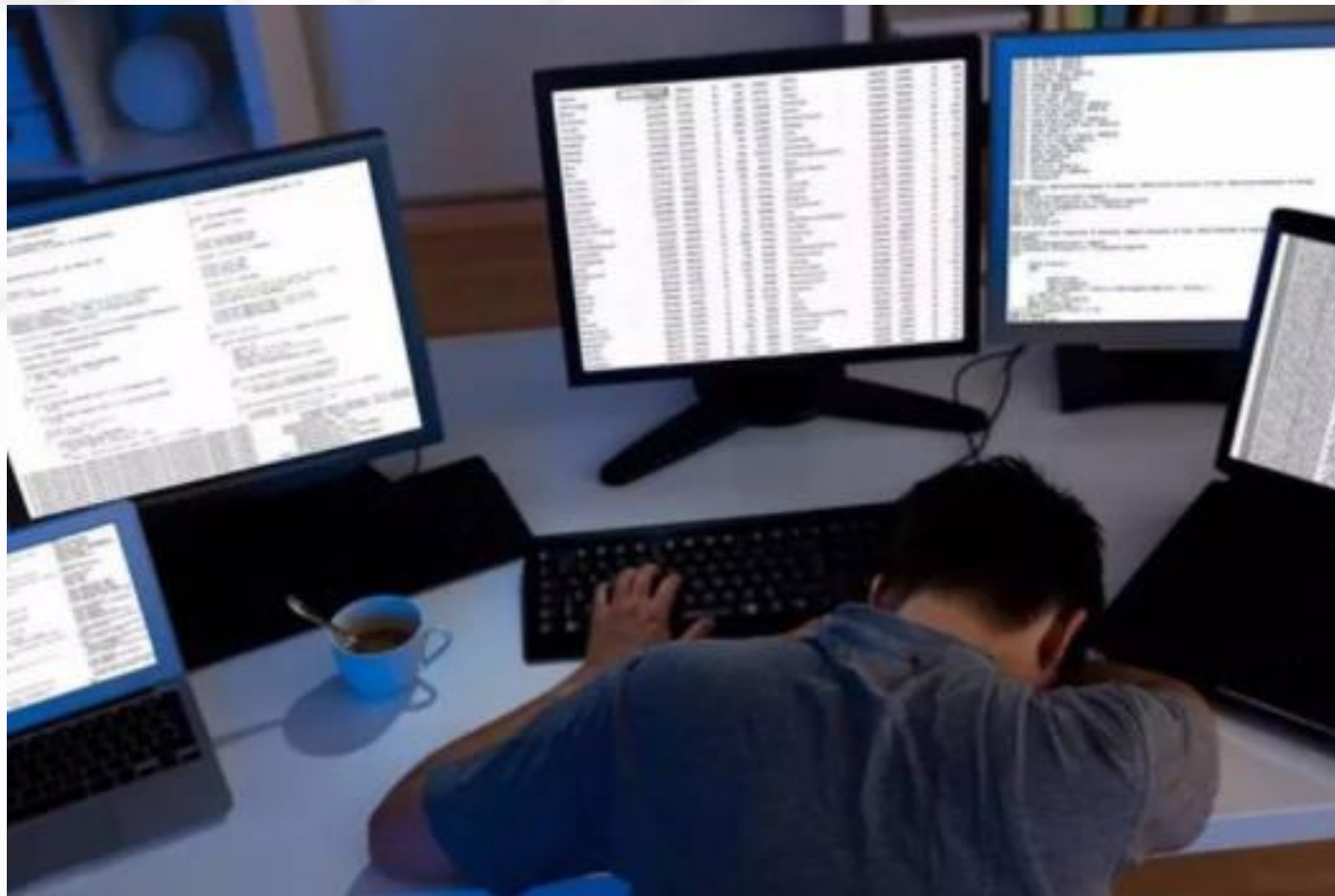
Email:JOHN85051232@GMAIL.COM

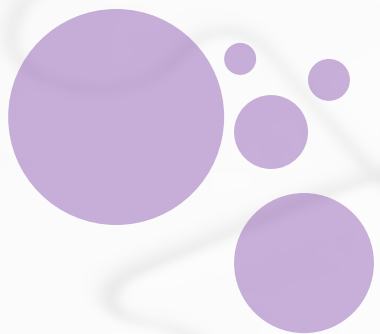
工程師日常

- 今天你在某家科技公司上班，你花了一整天都在Coding某件老闆交代的工作。
- 當你好不容易完成工作內容的時候，你的老闆跑過來問你的進度，你很開心的告訴他：
「老闆老闆我全部都完成了!!」
「喔～不錯喔...那客戶資料@#%^&*的排序你那邊怎麼做的」
「我用氣泡排序法去完成的」
「...你這樣寫一定不行啦，換個排序法重寫」



於是你熬夜加班繼續工作

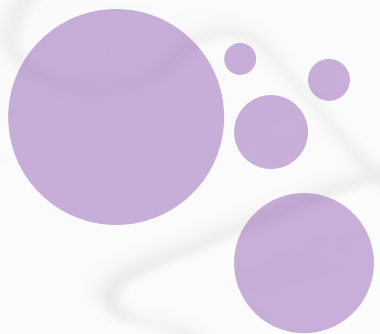




究竟

🔍Q：老闆是怎麼立刻知道你做的事情有問題？

1. 老闆存心刁難
2. 老闆想讓你加班
3. 老闆有學過資料結構和演算法，能夠快速的判斷程式碼的執行效能



程式效能比較

- ❖ 同樣的程式，用迴圈寫和用遞迴寫哪一種效能比較好？
- ❖ 排序法有很多種(Bubble sort、Insertion sort...)差異又在哪裡？
- ❖ 怎樣快速的評估程式碼執行效能？

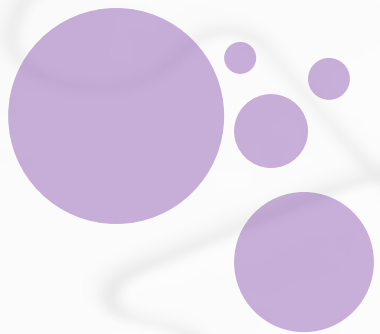


程式效能比較

在資料結構、演算法中常用下列兩種指標來比較程式效能好壞：

- ⚡時間複雜度：執行一段程式所要花費的時間
- ⚡空間複雜度：執行一段程式所占用的記憶體空間

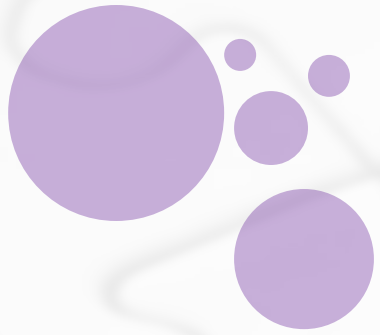
大部分看時間複雜度都會用Big-O的指標，也就是執行時間的上限(最壞的情況下執行會耗費多少的時間)。



Time Complexity

基本概念：

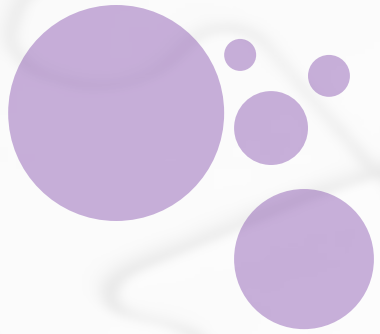
- ❖ 程式碼中，執行一次就算一個單位。
- ❖ 在迴圈中的程式碼，依據迴圈的判斷式決定他會執行幾次。



練習一下

下方程式碼執行了幾次？

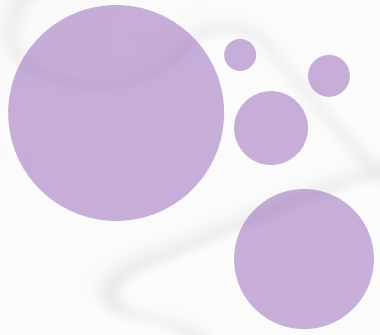
```
int main()
{
    for(int i = 0 ; i < 10 ; i++)
    {
        dosomething();
    }
    printf("hello world");
    return 0;
}
```

練習一下

剛開始看不出來？宣告個變數來記錄總共跑了幾次

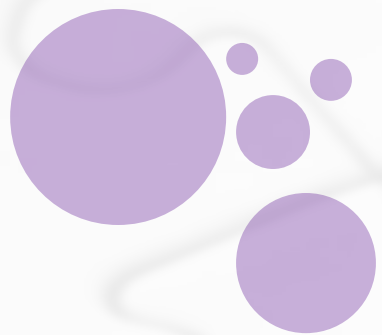
```
int main()
{
    int count = 0; //紀錄總共執行了幾行程式碼
    for(int i = 0 ; i < 10 ; i++)
    {
        dosomething();
        count++;
    }
    printf("hello world");
    count++;
    return 0;
}
```



練習一下

下方程式碼執行了幾次？

```
int main()
{
    for(int i = 0 ; i < N ; i++)
    {
        dosomething();
    }
    printf("hello world");
    return 0;
}
```



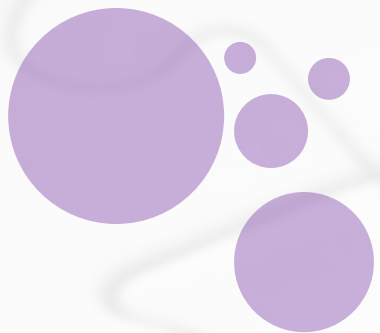
Time Complexity

時間複雜度(Time Complexity)

常見的時間複雜度：

🔲 $O(2^n)$ 、 $O(N^k)$ 、 $O(N)$ 、 $O(N \log N)$ 、 $O(1)$

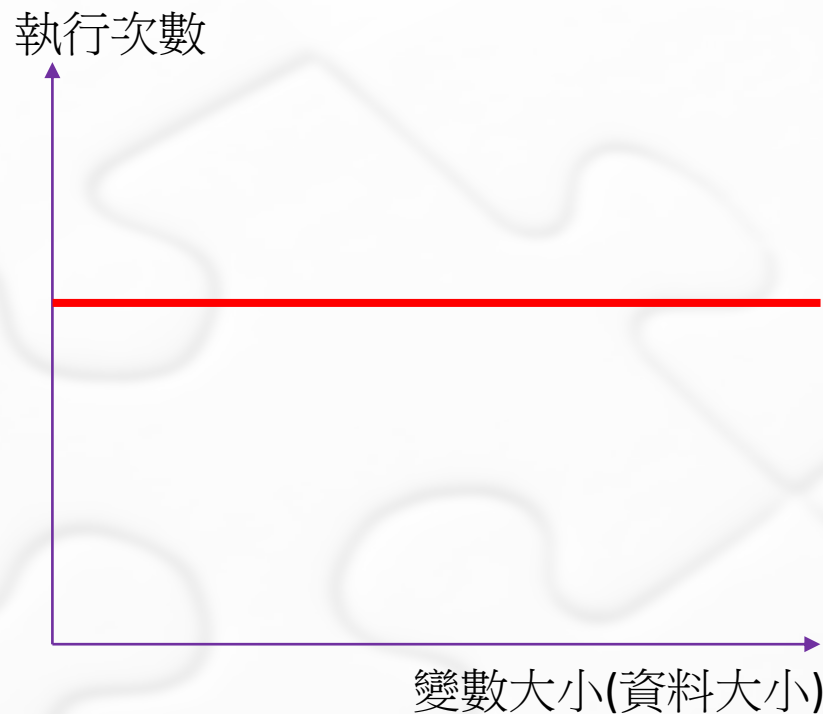
🔲 如何判斷？看程式碼執行的次數



$O(1)$ (常數時間)

🧩 在程式碼中只會執行常數時間，不會隨變數的值而改變

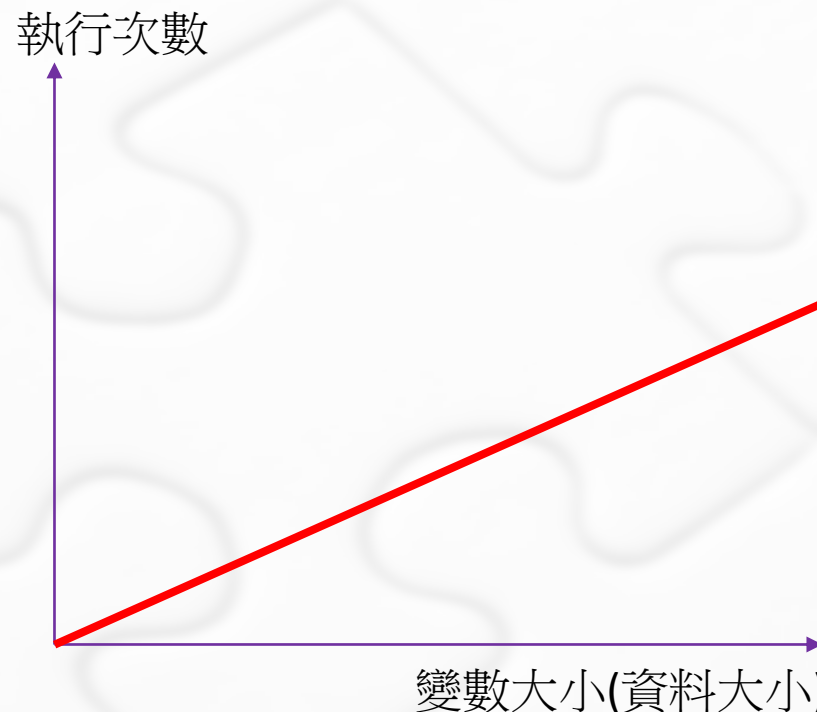
```
Void swap(int *a ,int *b)
{
    A = a^b;
    B = a^b;
    A = a^b;
}
int main()
{
    printf();
    scanf();
    swap();
}
```

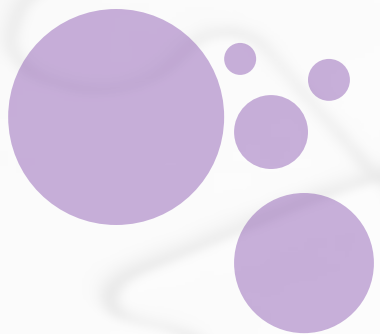


$O(N)$ (線性時間)

- 在程式碼中執行次數會依據變數的不同而改變，此時 O 的大小取決於執行的次數
- 下面迴圈會隨著 N 的值不同而執行 N 次，所以時間複雜度是 $O(N)$ (線性時間)

```
for(int i = 0 ; i < N ; i++)  
{  
    printf(i);  
}
```

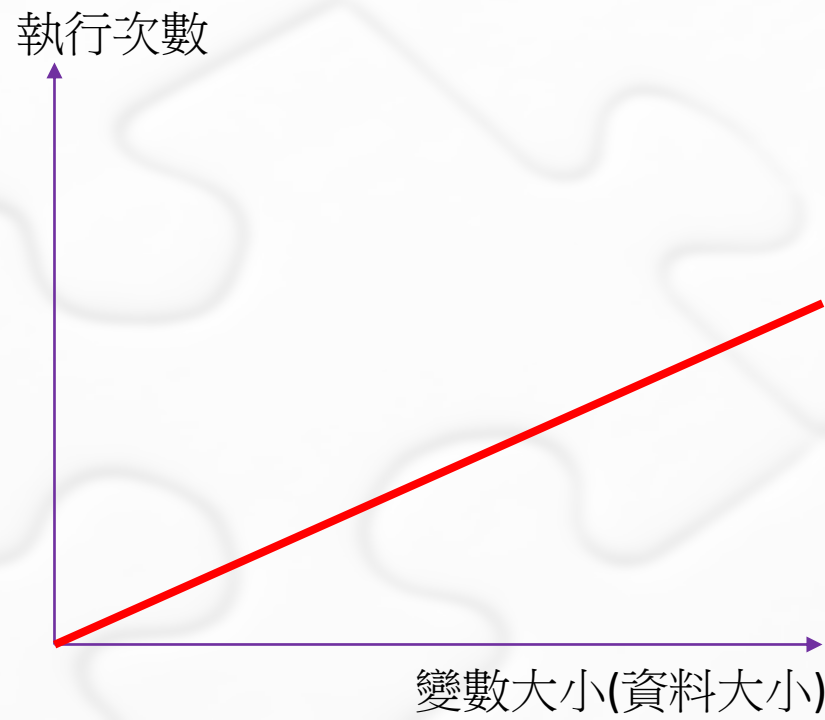




$O(N)$ (線性時間)

相同地，下面迴圈會隨著K的值不同而執行K次，所以時間複雜度是 $O(K)$ (線性時間)

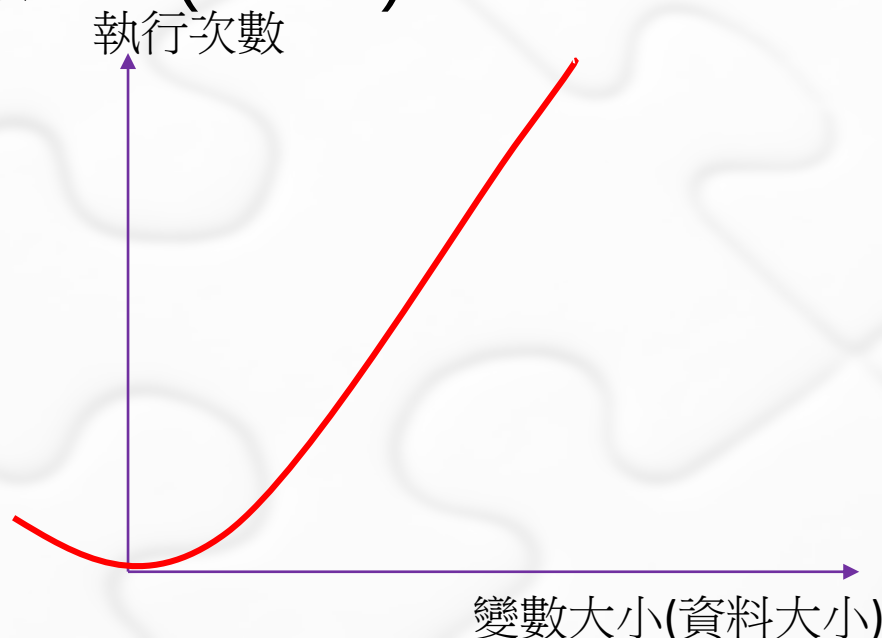
```
for(int i = 0 ; i < K ; i++)  
{  
    printf(i);  
}
```

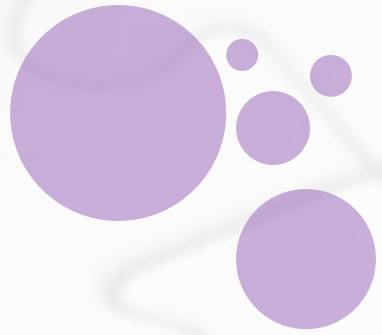


$O(N^2)$ (多項式時間)

- 下面迴圈會隨著N的值不同而執行 N^2 次，所以時間複雜度是多項式時間 $O(N^2)$
- P.S 這跟bubble sort的寫法有87%像(都是2層迴圈)，所以bubble sort的時間複雜度也是 $O(N^2)$

```
for(int i = 0 ; i < N ; i++)  
{  
    for(int i = 0 ; i < N ; i++)  
    {  
        printf(i);  
    }  
}
```



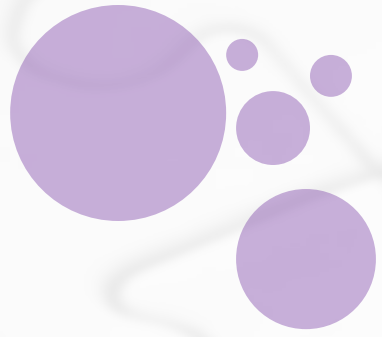


$O(N^2)$ (多項式時間)

多項式時間包含所有多項式的時間複雜度

$O(N^2)$ 、 $O(N^3)$...都稱作多項式時間

Q：有一段程式碼總共執行了 N^2+3N+1 次，那他的時間複雜度怎麼看？

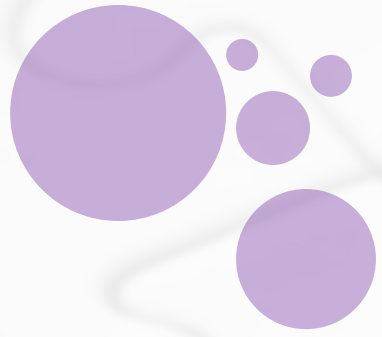


$O(N^2)$ (多項式時間)

❖ Q：有一段程式碼總共執行了 N^2+3N+1 次，那他的時間複雜度怎麼看？

❖ 找最大次方項當作整體的時間複雜度，也就是 $O(N^2)$ 。

❖ WHY？



$O(N^2)$ (多項式時間)

when $n \rightarrow 2$:

$$N^2 = 4$$

$$3N = 6$$

1

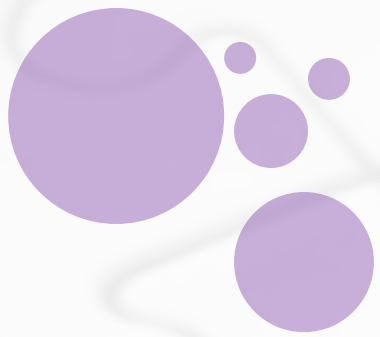
when $n \rightarrow 1000$:

$$N^2 = 1000000$$

$$3N = 3000$$

1

Q：當你有1000000元之後，走在路上發現有3000塊錢在地板上你會去撿嗎？



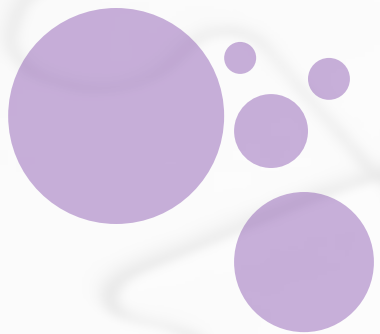
Time Complexity

🔲 最後來個難一點的，還記得binary search嗎？

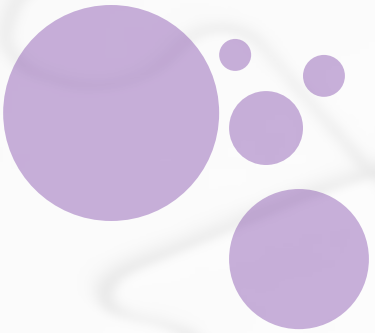
你用linear search => $O(N)$

如果用binary search => 只要 $O(\log N)$

🔲 舉個例子大概就是：你上班工作10000天才做的完的工作，如果只花了100天就做完了。



🔵 以後再寫程式之餘，如果對於相同的功能有不同的想法出現，可以透過時間複雜度來估計一下哪種會比較好。



Time Complexity in CPE

❖ 將測資筆數代入 N 來做為一個參考判斷。

❖ 再執行時間限制1秒的情況下：

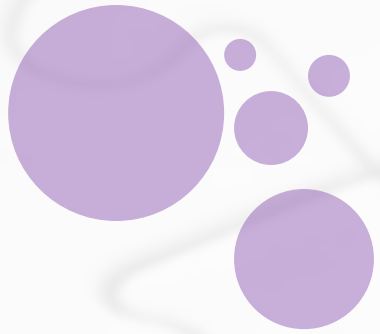
❖ 1000000：可行的寫法

❖ 10000000：有點勉強的寫法

❖ 100000000：你有較大的機率會接收到TLE

Ex：一個 $O(N^2)$ 的問題(Ex：要你針對 N 個數字做基礎排序)，
當測資數量最多有1000筆時，時間限制1秒下...

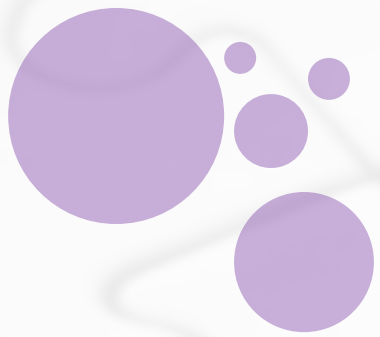
$1000^2 = 1000000$ ，代表 $O(N^2)$ 的演算法是個可行的解法。



休息一下

🔵我猜現在應該第二節剛開始一下下

🔵等下我們要來開始講**指標 & Linked List**



Pointer

今天有一種變數可以儲存某個記憶體位置，叫做指標變數

Ex : `int *b = &a;` //宣告b是個指標變數，指向(儲存)a變數的記憶體位置

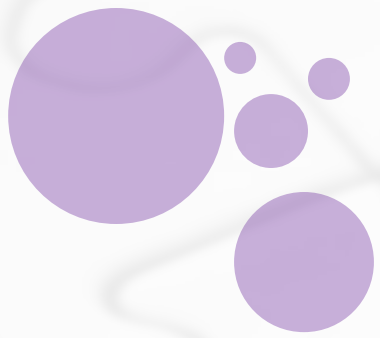
記憶體	a	b
	0x000001	0x000005
值	10	0x000001

- ❖ 變數b有自己的記憶體位址，但儲存的值是a的記憶體位址。
- ❖ 知道a變數的記憶體位址，我們就可以透過b去修改a。



Pointer

```
int num = 10;  
int *ptr = NULL; //宣告一個指標，不指向任何東西  
  
ptr = &num; //ptr指向num的記憶體位址  
  
printf("num的值 : %d\n", num);  
printf("num的記憶體位址 : %p\n", &num);  
  
printf("ptr指向位址的值 : %d\n", *ptr);  
printf("ptr指向位址的記憶體位址 : %p\n", ptr);  
printf("ptr的記憶體位址 : %p\n", &ptr);
```



New、Delete運算子

🔵 首先先宣告一個變數跟陣列來複習一下

`Int a = 0;`

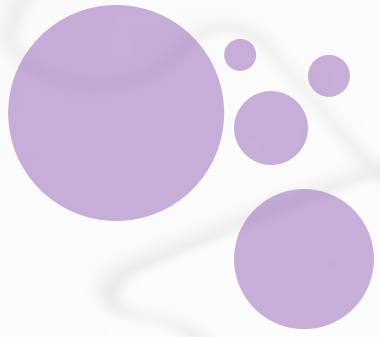
adress	0x0
value	0

`Int a[3] = {0};`


adress	0x0	0x4	0x8
value	0	0	0

🔵 電腦要給多少的記憶體空間都是程式執行前就知道的

🔵 Q：想在程式執行中根據變數大小產生陣列？

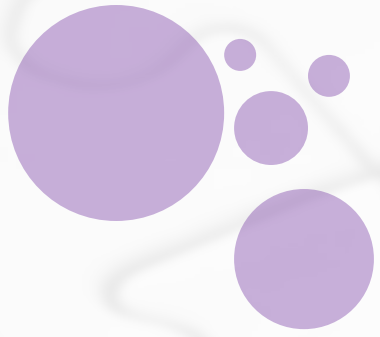


New、Delete運算子

 試試看

```
Int x = 3;  
Int arr[x] = {0};
```

C/C++ 無法用變數來宣告陣列(好啦我知道
C99以後可以)，那如果我想要每次執行程式
的時候陣列的大小都不一樣？



New、Delete運算子

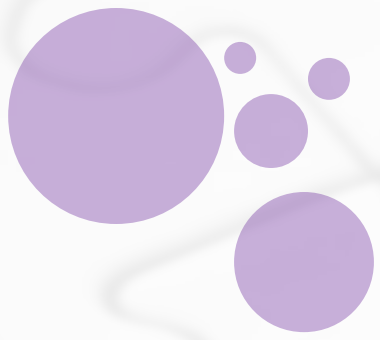
New：為變數動態產生一塊記憶體空間

Delete：將該變數的記憶體空間歸還(刪除)

```
int x = 3;  
int *arr = new int[x];//不能用arr[]，必須要用指標的方式
```

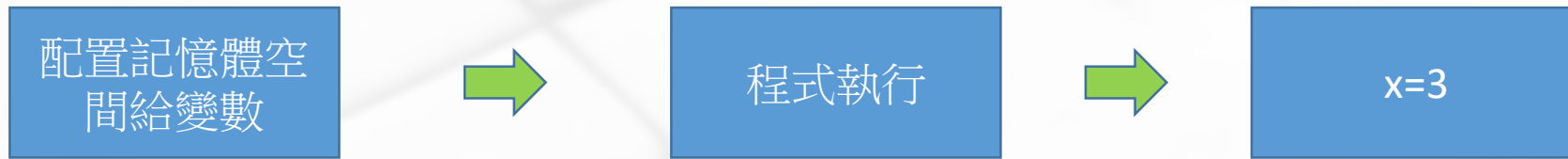
之前學過的變數宣告都是在**程式執行前**電腦就把記憶體**配置好了**，所以無法使用變數來宣告。

new可以讓電腦**動態配置記憶體**給變數，所以大小可以不固定。



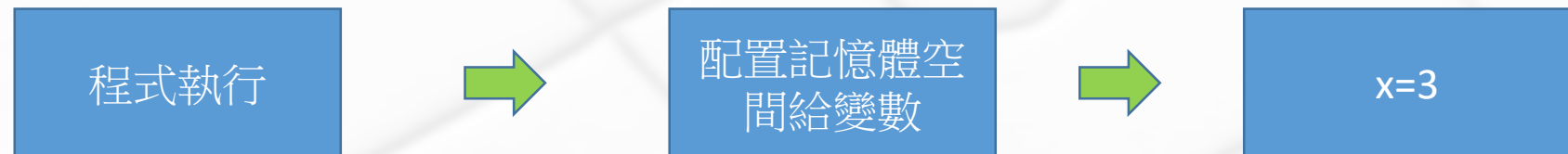
New、Delete運算子

之前的陣列宣告

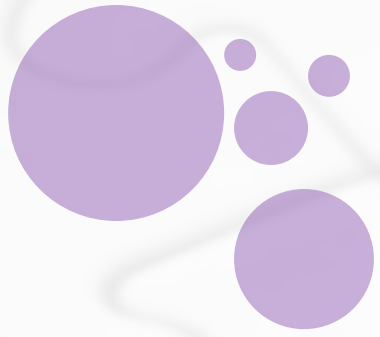


原本的變數宣告，記憶體空間再程式執行前完成，無法另外配置記憶體空間。

使用動態宣告



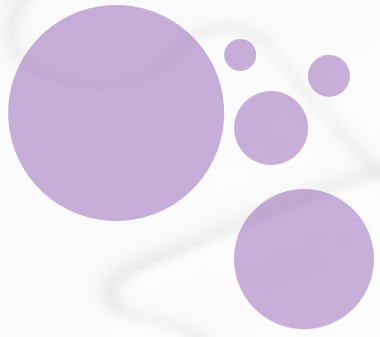
程式執行後才動態宣告新增記憶體給變數x。



New、Delete運算子

```
int *ptr = new int;//產生一個int的變數  
  
printf("ptr的值為:%d\n",*ptr);  
printf("ptr指向的記憶體位址:%p\n",ptr);  
printf("ptr的記憶體位址:%p\n",&ptr);  
delete ptr;  
//printf(...);
```

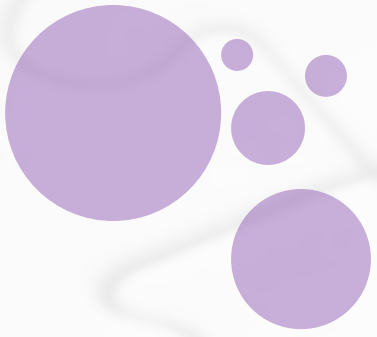
delete完後再printf一次變數試試看



New、Delete運算子

```
int x = 10;  
int *arr = new int[x];//用指標方式產生一個int的陣列  
for(int i = 0; i < x ; ++i)  
{  
    arr[i] = i;//配置空間不會主動做數值的初始化，要自己做  
    printf("%d ",arr[i]);  
}  
delete arr;//將這個指標的空間歸還給電腦
```

delete完後再printf一次陣列試試看



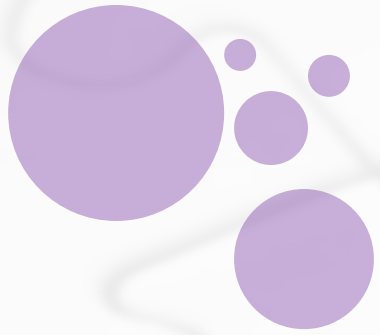
Pointer

```
int main()
{
    int *b;
    printf("%d",*b);
    //因為*b還沒儲存任何記憶體位址，無法將該位址的資料取出
    return 0;
}
```

如何解決？

1. `b = &a;` // `b` 指標參考其他變數的記憶體位址

2. `int *b = new int;` // 讓電腦生出一個整數的記憶體位址給 `b`

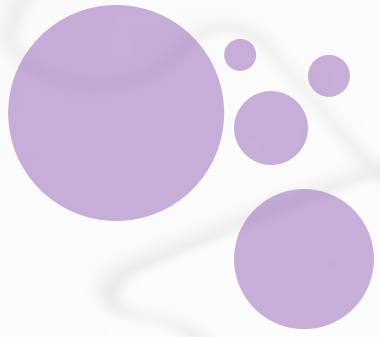


Pointer

比較一下兩個的差別

```
int main()
{
    int a = 10;
    int *b = &a;
    *b = 300;
    printf("%d",*b);
    return 0;
}
```

```
int main()
{
    int *b = new int;
    *b = 300;
    printf("%d",*b);
    return 0;
}
```

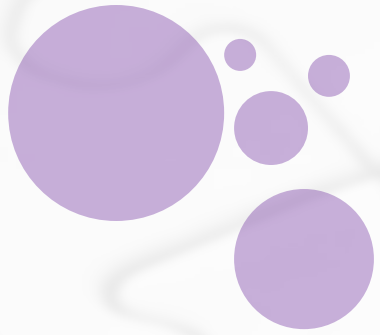


New、Delete運算子

動態宣告在C/C++有不同的function：

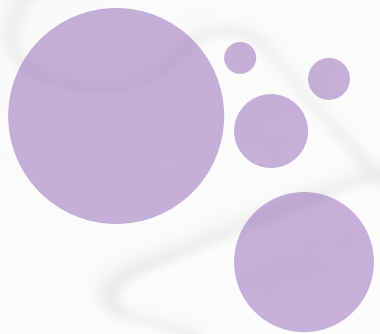
🔷 C：malloc、free

🔷 C++：new、delete



休息一下

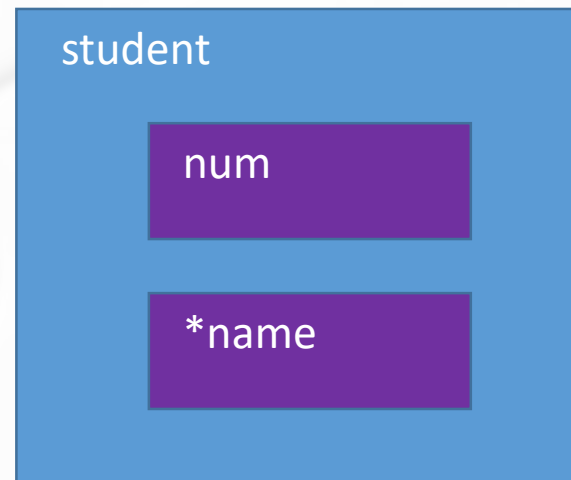
🔵我希望現在已經第三節了



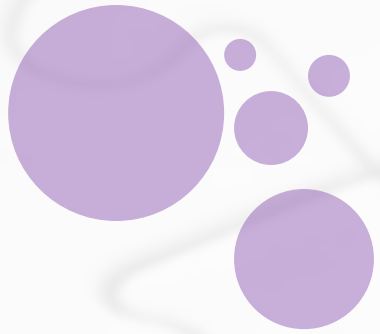
當struct遇上指標

複習一下struct是什麼：可以把東西包在一起的概念
Ex：學生的資料有座號、姓名.....

```
struct student
{
    int num;
    char *name;
};
```

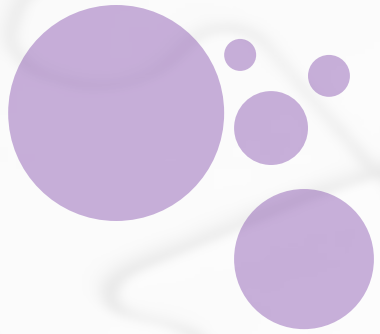


student.num
student.name



當struct遇上指標

```
...  
int main()  
{  
    student s;  
    s.num = 1;  
    s.name = "Abel";  
    printf("%d %s\n",s.num,s.name);  
}
```

當struct遇上指標

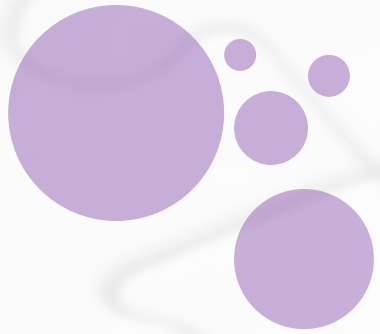
那struct的變數能不能是指標？當然可以

```
int main()
{
    student *p1 = new student;//動態配置記憶體
    system("pause");
    return 0;
}
```

功能和一般變數用.是一樣的效果，
只是指標取得內部成員變數的方式不太一樣

如何得到struct指標變數內的變數？

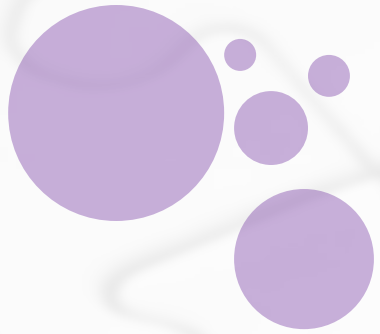
```
student->num
student->name
```



當struct遇上指標

試試看

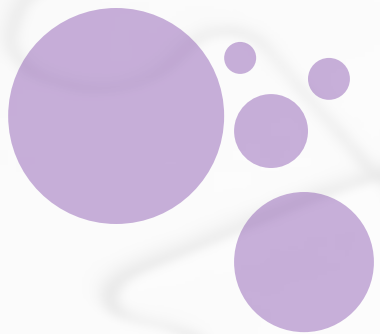
```
int main()
{
    student *p1 = new student;
    printf("%s %d\n",p1->name,p1->num);
    system("pause");
    return 0;
}
```



當struct遇上指標

剛剛的Code會亂碼，記得只要是變數都要乖乖給初始值

```
int main()
{
    student *p1 = new student;
    p1->name = "john";
    p1->num = 123;
    printf("%s %d\n",p1->name,p1->num);
    system("pause");
    return 0;
}
```

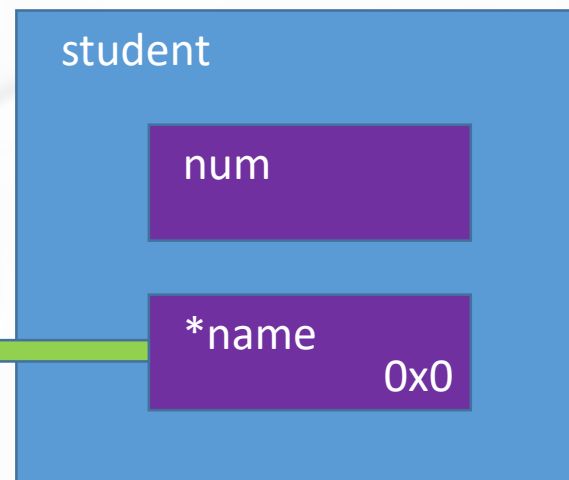


當struct遇上指標

struct變數內部的指標變數代表什麼？

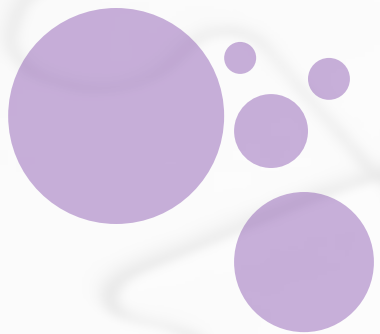
我們說過**指標是紀錄一個變數記憶體的位置**

`char *name;`
一個紀錄字串(字元陣列)開頭位址的指標變數



	↓			
adress	0x0	0x1	0x2	0x3
value	'j'	'O'	'H'	'N'

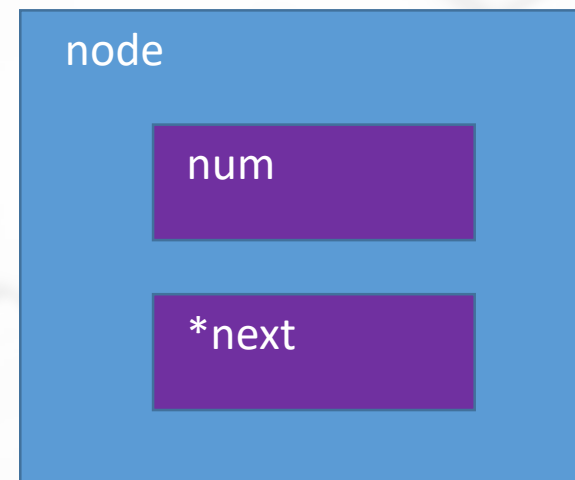
```
printf("%c",*name);//j  
printf("%p",name);//0x0
```

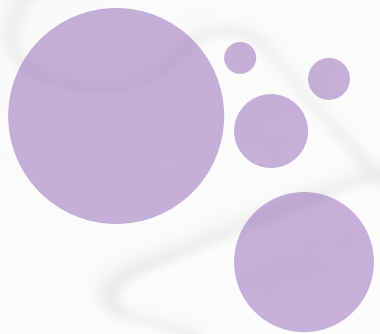


當struct遇上指標

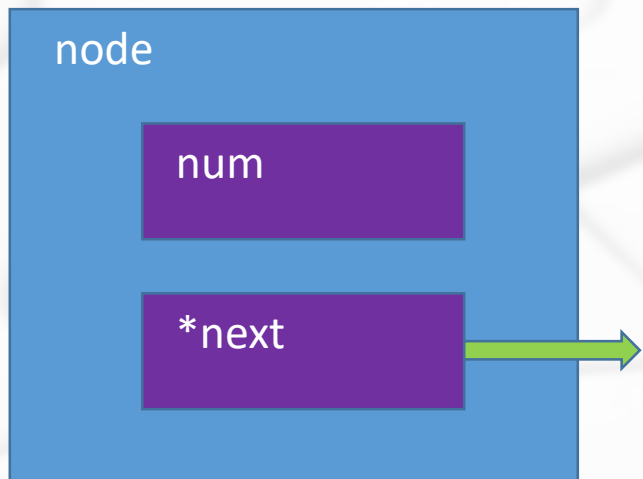
```
struct node
{
    int num;
    node *next; //宣告一個node結構的指標變數
};

int main()
{
    node *n1 = new node; //宣告一個node結構的指標變數
    ...
}
```





當struct遇上指標



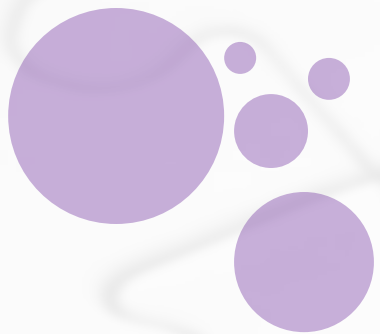
`node* next;`
可以用來紀錄node結構變數的記憶體位址



當struct遇上指標

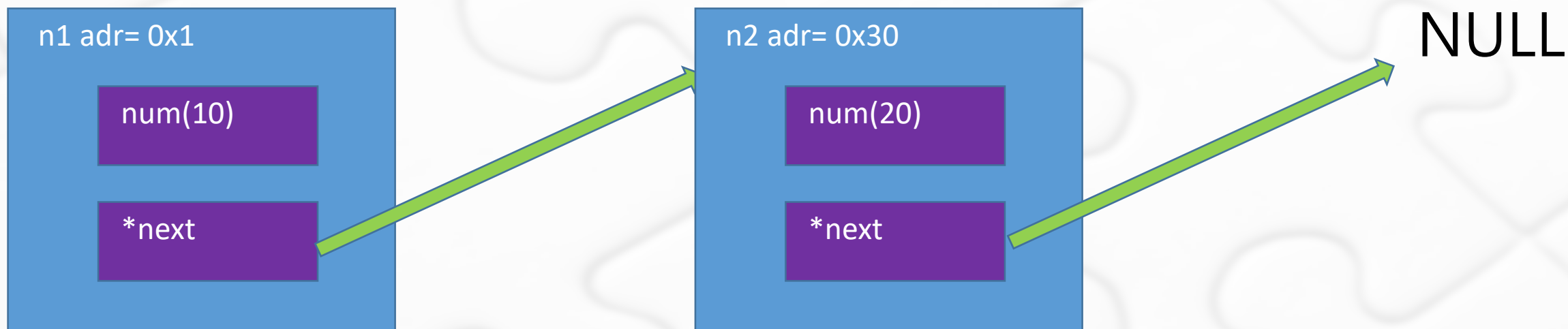
```
int main()
{
    //宣告node結構的指標變數
    node *n1 = new node;
    node *n2 = new node;
    n1->num = 10;
    n2->num = 20;
    n1->next = n2;
    n2->next = NULL;
    printf("n1的值:%d n1->next儲存的記憶體位置:%p\n", n1->num, n1->next);
    printf("n2的值:%d n2的記憶體位置:%p\n", n2->num, n2);

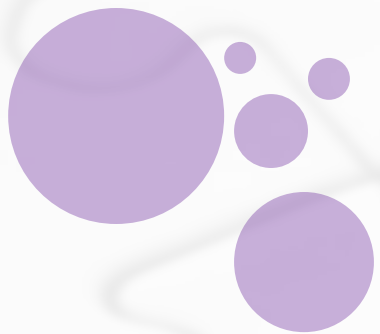
    system("pause");
    return 0;
}
```



當struct遇上指標

What Happen?



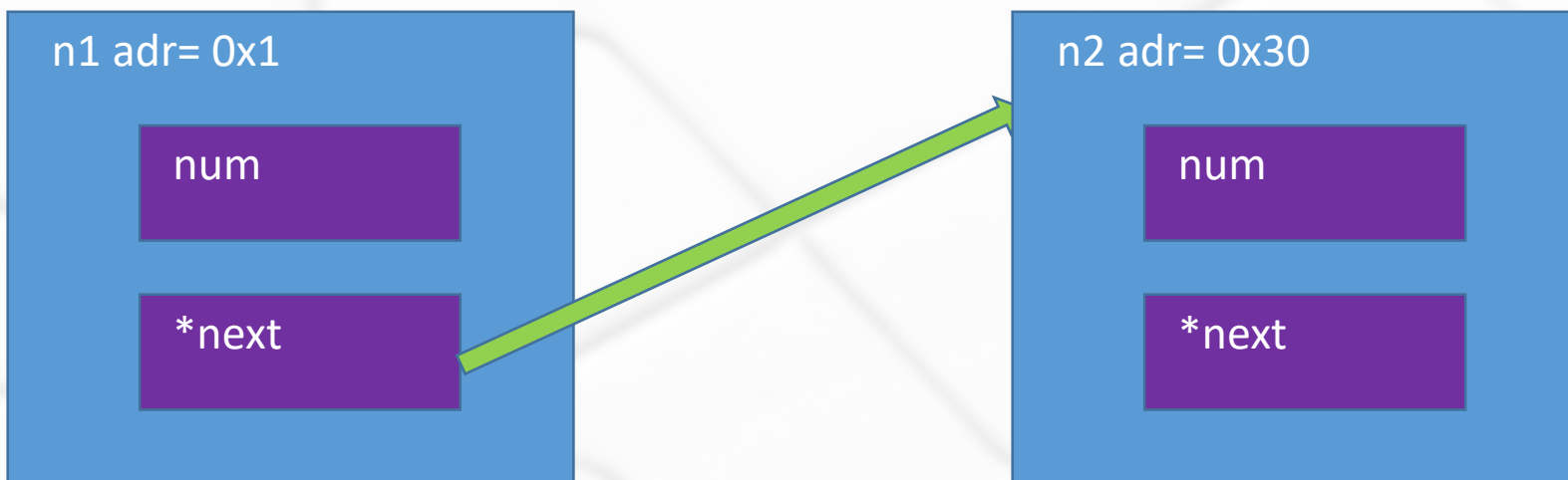


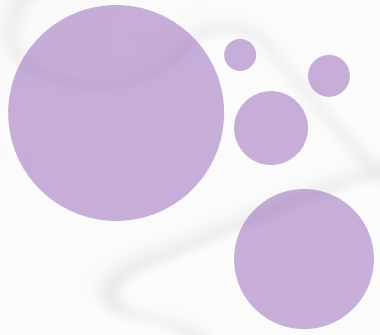
當struct遇上指標

```
printf("n2的值:%d\n", n1->next->num);
```



n2的記憶體位置





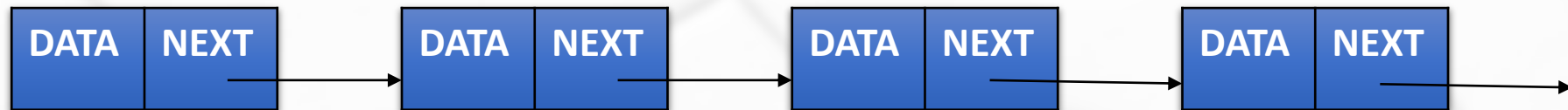
Linked list

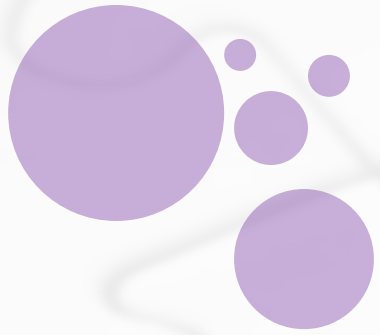
鏈結串列(Linked list)：

- ❖ 資料結構的一種

- ❖ struct + pointer

- ❖ 每一個節點(node)都包含指向下一個節點的指標





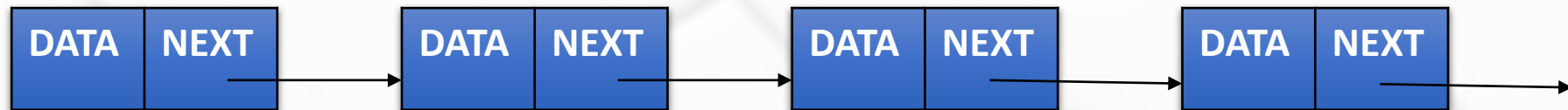
Linked list

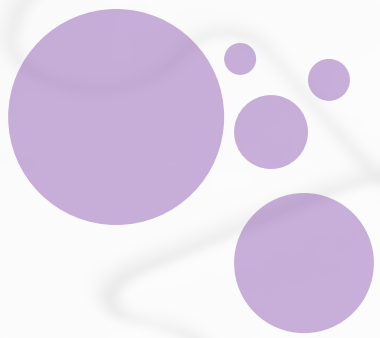
基本操作：

🔲 移動(走訪)

🔲 新增

🔲 刪除





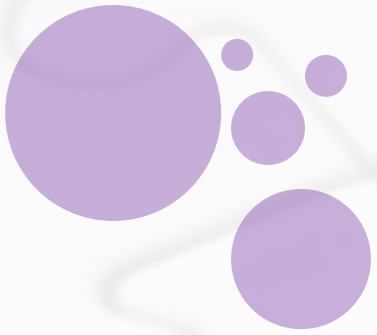
Linked list

先建好結構

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int num;
    node *next;
};
```

```
int main( )
{
    node *n1 = new node;
    node *n2 = new node;
    node *n3 = new node;
    n1->num = 10;
    n1->next = n2;
    n2->num = 20;
    n2->next = n3;
    n3->num = 30;
    n3->next = NULL;

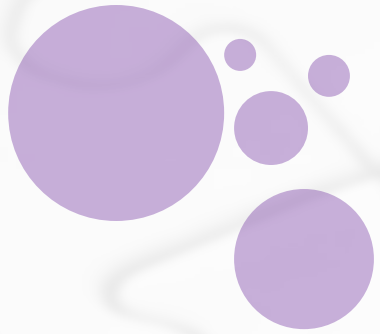
    system( "pause" );
    return 0;
}
```



Linked list

```
printf("%d %d %d \n",p1->num,p1->next->num,p1->next->next->num);
```



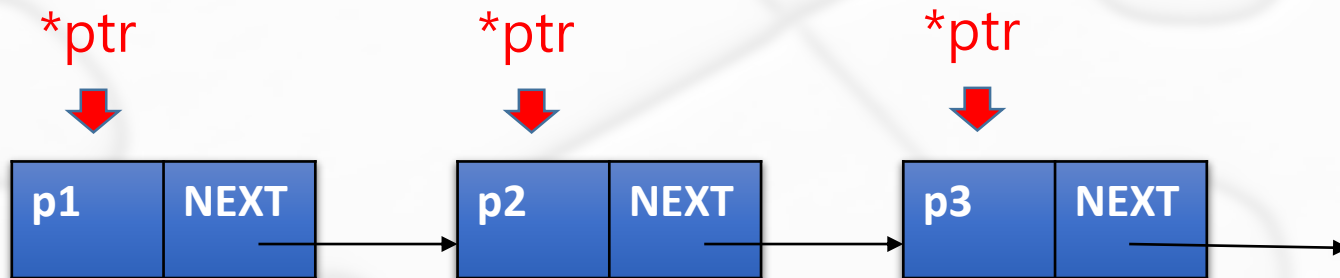


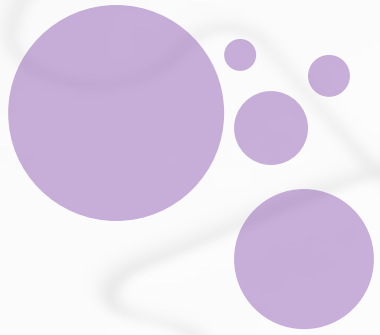
Linked list – 移動

```
node *ptr = p1;  
printf("%d",ptr->num);
```

```
ptr = ptr->next;  
printf("%d",ptr->num);
```

```
ptr = ptr->next;  
printf("%d",ptr->num);
```



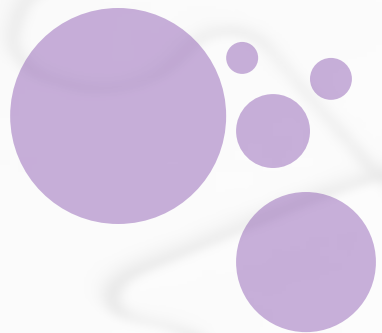


Linked list – 移動

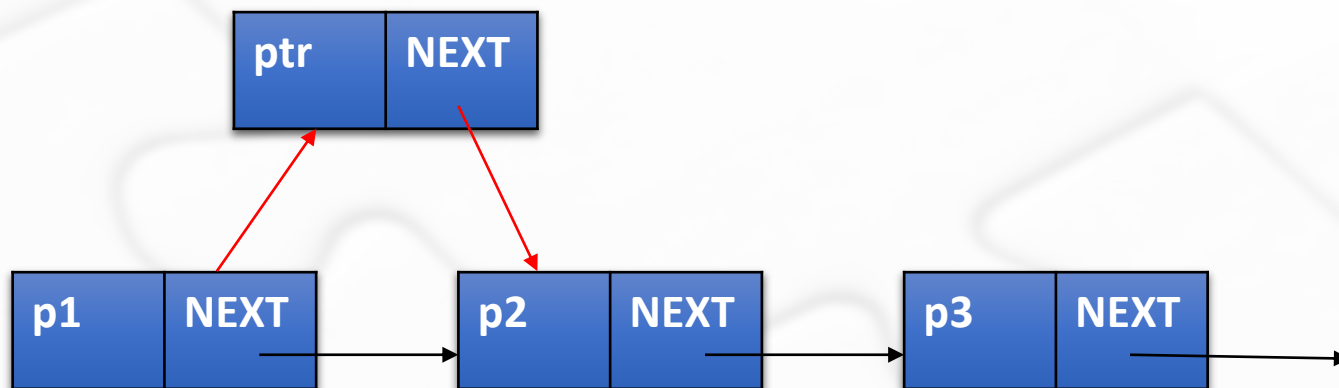
寫成迴圈

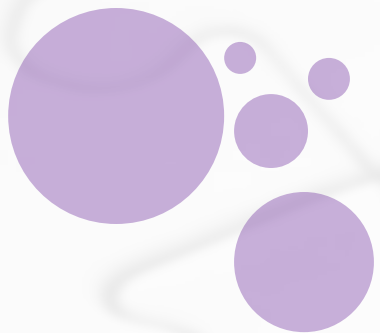
```
node *ptr = p1;  
while(ptr != NULL)  
{  
    printf("%d \n",ptr->num);  
    ptr = ptr->next;  
}
```





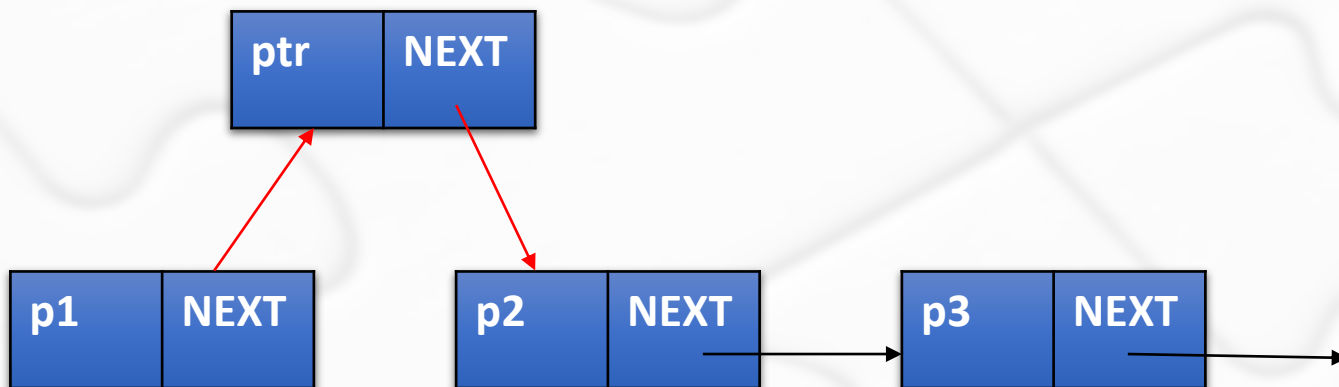
Linked list – 新增

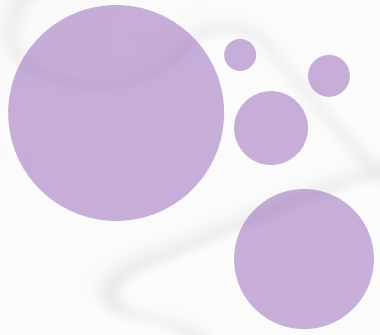




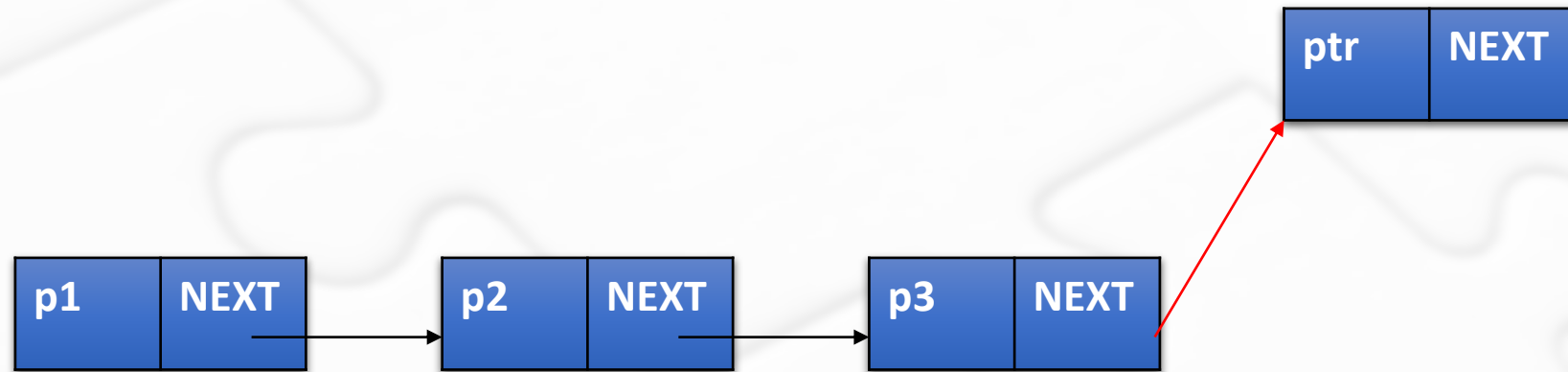
Linked list – 新增

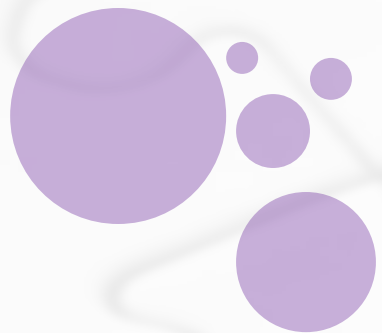
```
node *ptr = new node;//新增一個節點  
ptr->num = 40;  
p1->next = ptr;  
ptr->next = p2;
```



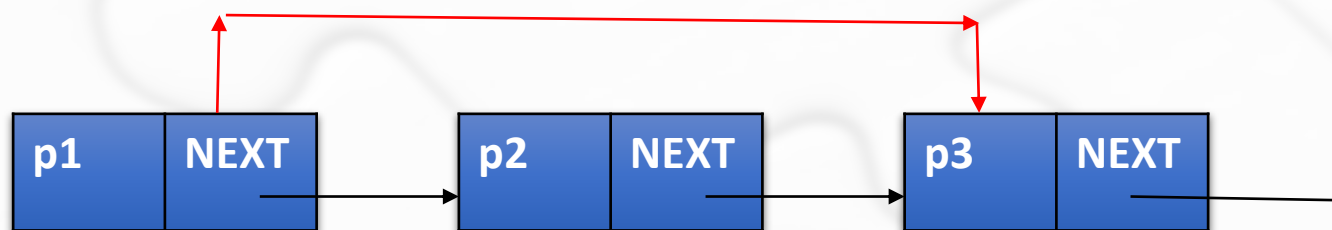


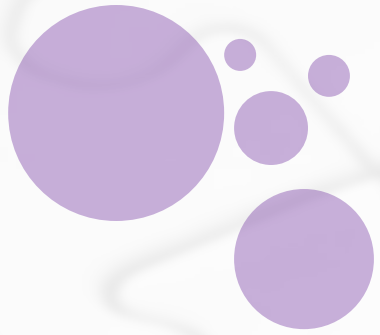
Linked list – 新增(2)





Linked list – 删除

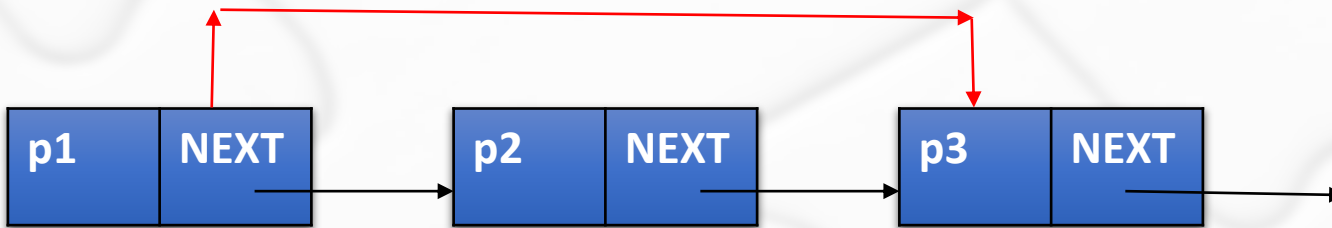


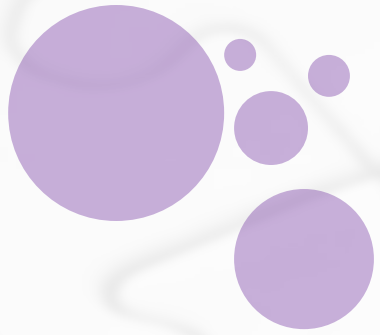


Linked list – 刪除

```
node *ptr = p1->next;  
p1->next = p1->next->next;
```

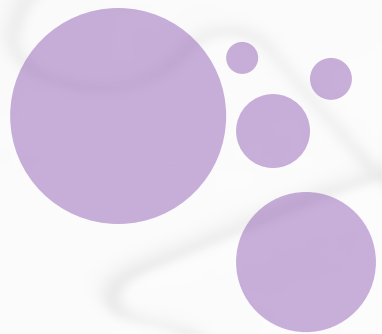
delete ptr;//刪除ptr所在位置的節點





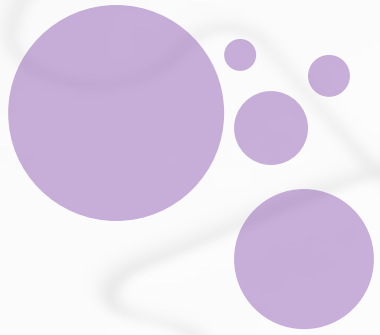
Linked list – 刪除(2)





現在應該下課了吧

🔵沒有就休息吧～下次再繼續講～～



點名摺

🔲 請用txt回答下列問題，並於下課之前上傳到E平台指定作業區。

🔲 請問下列的時間複雜度分別為何？

1. 1000

2. $0.5(n^2) + 100n + 3000000$

3. $2^N + 1000N$



THANK YOU