進階課程

減少程式碼的常用技巧

這裡講的是真的少打一點，不是全部縮成一排。減少程式碼可以方便閱讀且在bebug(找錯誤)比較方便

定義時賦予值

```
int a b;
a=87;
```

可以簡化成`int a=87, b;`一行

在輸出行內進行運算

```
a=x+y;
cout<<a
```

可以簡化成`cout<<x+y;`一行

if else省去大括號

if else裡面如果只有一行指令可以省去大括號`{}`

```
if(a>b){
    cout<<a;
}else{
    cout<<b
}
```

可以簡化成`if(a>b) cout<<a; else cout<<b;`一行

for

```
int main()
{
   for (i = 0; i < max; i++){
       printf("Number of spaces: %i\n", space);
   }
   return 0;
}
```

Switch

```
switch(運算式){
    case 常數1:
        [敘述區塊1;]
        [break;]
    [case 常數2:]
        [敘述區塊2;]
        [break;]
        。。。
    [default:]
        敘述區塊n;]
}
```

觀察這個

```
int a = 1
switch(x){
    case 1:
         printf(“A”);
         break;
    case 2:
        printf(“B”;
    case 2:
        printf(“C”);
         break;
}
```

陣列 Arrrrrrrrrrrray~~

陣列將同一型態同一作用的變數排在一起，宣告時與一般的變數不同，請見以下範例：

```c
#include<stdio.h>

int main() {
    int arr[5];
    arr[0]=10;
    arr[1]=10;
    arr[2]=20;
    arr[3]=20;
    arr[4]=30;
    printf("%d %d %d %d %d\n", arr[0], arr[1], arr[2], arr[3], arr[4]);
    return 0;
}
```

> - **索引(Index)**
> 表示順序或編號的數值，在程式設計中通常從 0 開始

可以很清楚的看到，陣列在宣告的時候，變數名稱後面需加上 `[n]` ，其中n為陣列的長度。

之後在存取的時候，不能只寫 `array = 0;` ，因為陣列是一長串的變數，只是他們的名字都相同，這時候編譯器會不知道我們到底要讀寫哪一個變數，所以在讀寫的時候，必須要加上它們各自的編號。

1號叫做 `array[0]`

2號叫做 `array[1]`

陣列初始化

用 `{}` 可以在 **宣告的時候** 指派陣列中的多個值，值的數量不能超過陣列的大小

```c
#include<stdio.h>

int main(){
    int arr[10]={10, 10, 20, 20, 30};
    printf("%d %d %d %d %d\n", arr[0], arr[1], arr[2], arr[3], arr[4]);
    return 0;
}
```

結合迴圈，可以快速地設定、使用陣列的內容

```c
#include<stdio.h>

int main(){
    int arr[100], n;
    scanf("%d", &n)
    for(int i=0; i!=n; i++)
        arr[i]=i;
    for(int i=0; i!=n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

如果要讓陣列全部為0，可以指派 `{0}`

```c
#include<stdio.h>
#include<stdlib.h>

int main(){
    int arr[10]={0};
    for(int i=0; i!=10;i++)
        printf("%d ", arr[i]);
    return 0;
}
```

定義

語法是: `#define <目標文字> <替換文字>`

用 `#define` 定義的函式，執行速度會比自訂函式還快，但函式功能複雜時不好編寫，且容易造成執行結果錯誤

```c
#include<stdio.h>
#define plus + //定義函式
#define x 1 //定義變數
#define y 2 //定義變數
#define say printf

int main(){
    say("%d\n", x plus y);
    return 0;
}
```

搜尋法

循序搜尋法(Sequential Search)

定義

從第一個資料開始取出，依序一一與「目標資料」相互比較，直到找到所要元素或所有資料均尋找完為止，此方法稱「循序搜尋」。

優點

1. 程式容易撰寫。
2. 資料不須事先排序(Sorting)。

缺點

搜尋效率比較差(平均次數=(N+1)/2)，不管是否有排序，每次都必須要從頭到尾找一次。

演算法

```
int main() {
  int i, n=10, key=5;
  int list[10] = {0,1,3,5,7,9,2,4,6,8,10};
  for (i = 0; i < n; i++){
    if (list[i] == key) break;
    }
    printf("第%d項", i);
    return 0
}
```

二分搜尋法(Binary Search)

定義

如果資料已先排序過，則可使用二分法來進行搜尋。二分法是將資料分成兩部份，再將鍵值與中間值比較，如鍵值相等則找到，小於再比前半段，大於再比後半段。如此，分段比較至找到或無資料為止。

優點

搜尋效率佳(平均次數=Log2N)。

缺點

1. 資料必需事先排序。
2. 檔案資料必需使是可直接存取或隨機檔。

演算法

```
#include<stdio.h>

int main() {
   int Temp[5] = {1,2,3,4,5};
   int Low = Temp[0],High = Temp[4];
   int Middle = (Low + High) / 2; //搜尋中間值
   int Key = 4;
     while (Temp[Middle] != Key){ //找到資料
         if (Temp[Middle] < Key)
           Low = Middle + 2; //改變左半部
           else High = Middle - 1; //改變右半部
           Middle = (Low + High) / 2; //改變中間值
   }
   printf("%d 排在第 %d 個順位", Key, Middle);
   return 0
   }
```

排序法

變數交換

```
#include<stdio.h>

int main() {
    int a[]={1,87,69,3}, tmp, i, b;
    printf("排序前:%d %d %d %d\n", a[0], a[1], a[2], a[3]);
    for(i=0;i<3;i++){
        for(b=0;b<3;b++){
            if(a[b]>a[b+1]){
                tmp=a[b];
                a[b]=a[b+1];
                a[b+1]=tmp;
            }
        }
    }
    printf("排序後:%d %d %d %d", a[0], a[1], a[2], a[3]);
    return 0;
   }
```

氣泡排序(Bubble sorting)

資料結構中最簡單之排序法。所謂氣泡排序法就是相臨資料互相比較，若發現資料順序不對，就將資料互換。依次由上往下比，則結果將如氣泡般，依次由下往上浮起。

分析

1. 比較之回合數=資料數(n)-1。
2. 每一回合至少有一資料可以排列至正確之次序。
3. 時間複製度，最差與平均時間O(n2)
4. 需要一個額外(元素)空間。
5. 為一穩定排序。
6. 資料量小時，使用效果佳。

原理

1. 每一回合逐一比較相臨資料，依排序之順序交換位置。
2. 每回合至少會有一次交換位置，至沒交換位置則停止。

演算法

氣泡排序法之程式

```
#include<stdio.h>

int main() {
  int i, j, k, t = 1, Temp, sp;
  for (i = n - 1; i > 0; i--) {
    sp = 1;
    for (j = 0; j <= i; j++)
      if (A[j] > A[j + 1]) { //兩數交換位置
        Temp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = Temp;
        sp = 0;
      }
    if (sp == 1) break;
  }
  return 0
}
```

APCS範例題目

http://cs.cysh.cy.edu.tw/elective/c++/201806APCS.pdf