# APCS檢測衝刺班

舉辦方：左營高中

左營高中 214 23 柯安聰

# 目錄

# 課程說明

這個課程是為了讓學生準備APCS考試而設計的，APCS是大學程式設計先修檢測，是由教育部推動的一項具公信力的程式能力檢定，目的是讓具備程式設計能力的高中職生能夠檢驗學習成果，並供作大學選才的參考依據。APCS考試分為兩種類型，一種是程式設計觀念題，一種是程式設計實作題，考生可以選擇使用Python、C、Java或C++等四種語言作答。

在這個課程中，我們學習了C語言的基本語法和應用，例如變數、運算子、流程控制、函數、陣列、指標等。我們也學習了一些重要的程式設計概念和演算法，例如資料結構、排序、搜尋、遞迴等。我們每周都會做一些練習題和模擬試題，來提升我們的解題能力和速度。

# 簡報前言

這份簡報統整了課堂上所教的程式與概念，我想藉由著整理這些程式和概念，來更好的了解到有關程式的相關訊息，同時也能去釐清簡報中的各個部分。

因此在後續的簡報中，將會講解每行程式(有些經常出現的程式會不再過多解釋)，並且會附上其輸出結果。

而簡報中的程式都是跟著老師一步步的教學所輸入的，其註解是老師上課補充、我自己另外補充的部分，另外我在有些概念上也自己去另外查詢了資料，因此簡報中會有些額外的內容。

# 上課內容

2/25(1)

這段程式通常是用來測試程式是否能運作時的所第一個會使用到的程式碼。
其中的("hello world")，就很像是在歡迎能順利完成運作的程式的誕生。
第一行是用來導入C語言的函式庫

```
Hello world!
```

```c
#include<stdio.h>
int main(){
    printf("Hello world!");
    return 0;
}
```

這一段程式碼則是運用到了一些排版的用法，像是跳行或是跳格，我們在原先的基礎程式之上再另外加上了這些運作，使的文字在輸出後能有換行或跳格，而通常是使用於避免輸出的字串或結果混亂。

```
Hello world!
Hello world!    Hello world!
```

```c
#include<stdio.h>
int main(){
    printf("Hello world!\n");
    //   \n跳行
    printf("Hello world!\t");
    //   \t跳格
    printf("Hello world!");
    return 0;
}
```

# 上課內容

2/25(2)

此段程式則是基礎的加法運算，先定義整數 a、b，再設定a、b的數字，如果定義出的a或b不是整數，就會出現以下狀況，例：
a=0.5 b=5 則此刻的a輸出則會是0

a=3.444 b=5 則此刻a輸出則會是3
但如果把第三行的int改成float，就能按照小樹的樣子去進行輸出了，

因此我們可以從上述舉例得知，當定義的 int並非整數時，則會以無條件捨去法來得到整數數字。

從第6～８行，可以看見一個為％ｄ的型態格式，
而此就是用於十進制的整數輸出，當然還會有其他的型態格式，例如％ｏ，一樣是用於輸出數字，但卻會變成八進制的整數輸出。而如果要輸出小數的話則改成%f，通常會算至小數點後六位。

```
1   #include<stdio.h>
2   int main(){
3       int a,b;
4       a=4;
5       b=5;
6       printf("%d\n",a);
7       printf("%d\n",b);
8       printf("%d\n",a+b);
9       return 0;
10  }
```

# 上課內容

2/25(3)

首先第一行為導入標準輸入輸出的函式，可使其他程式有其運作。
第二行為定義主函式。

第三行是先定義兩個小數 a=4.005，b=5.7。

第四、五行都是宣告此 a、b變數的值。

第六行輸出a，也就是4.005。

第七行輸出b，也就是5.7。

第八行輸出a+b，也就是輸出4.005+5.7

第九行回傳0以正常結束函式。

```c
#include<stdio.h>
int main(){
    float a,b;
    a=4.005;
    b=5.7;
    printf("%f\n",a);
    printf("%f\n",b);
    printf("%f\n",a+b);
    return 0;
}
```

```
4.005000
5.700000
9.705000
```

# 上課內容

2/25(4)

```c
#include<stdio.h>
int main(){
    int a,b;

    printf("請輸入整數：");
    scanf("%d",&a);
    printf("%d\n",a);

    printf("請輸入整數：");
    scanf("%d",&b);
    printf("%d\n",b);

    printf("%d\n",a+b);
    return 0;
}
```

第二行進入到了主程式。

第三行定義整數變數a、b。

第五行輸出括號的內容。
第六行儲存輸入的a值。
第七行輸出儲存完的a值

第九行輸出括號內的內容。
第十行儲存輸入的內容(b)。
第十一行輸出b的值。

第十三行輸出a+b的值。

```
請輸入整數：20
20
請輸入整數：60
60
80
```

# 上課內容



這程式使用到了可以用於儲存輸入的資料的 scanf，也就是在輸出此段程式後，會先定義 a、b的小數型態，再輸出print內的"enter a number:"
此時就會因為需要輸入一個數字（資料）才能儲存進Scanf當中，而當輸入了數字之後，就會再依照著之後的程式碼去完成其他的輸出。

如上一頁有大概的說明%f，是一個用於小數點的型態

%f通常默認的輸出小數會到小數點後的六位數，而%.3f則是只取道小數點後的三位數

```c
#include<stdio.h>
int main(){
    float a,b;
    printf("enturn a number:");
    scanf("%f",&a);
    printf("%f\n",a);
    printf("%.3f\n",a);
    return 0;
}
```

```
enturn a number:4.5
4.500000
4.500
```

# 上課內容

2/25(6)

這段程式的作用是輸出字元（單獨一個字，空白鍵也算一個字元，這在ASCII碼會有相關資訊），第三行的的char用來定義字元c(代號)，而在程式顯示出 ENTER A CHAR之後，就會讓使用者輸入一個字元來定義 c這個代號，然後會利用第七行的getchar來把輸入的字元給儲存，功能就像是scanf，而第九行的部分是用來跳一行用的，第十則是輸出剛才輸入的字元，第十一行是單獨輸出一個 A字。
也可以利用數個字元來串成字串（含兩個以上的字元）

總結：
第三行，定義c的字元
第五行，輸出ENTER A CHAR
第九行，跳行
第十行，輸出c的字元
第十一行，輸出A

```
ENTER A CHARW

WA
```

```c
#include<stdio.h>
int main(){
    char c;
    /*顯示*/
    printf("ENTER A CHAR");
    //讀取字元
    c = getchar();
    //顯示字元
    putchar('\n');
    putchar(c);
    putchar('A');
    return 0;
}
```

# 上課內容



第三行中，在定義a字元的char後面接[10]式代表著a擁有十個格子，可以用來十個字元（超過十個則不儲存於a），從而形成一段話或詞。
stdin用來讀取輸入的字串並儲存。

fgets是等到輸入完資料後按下回車鍵（enter)的確認步驟。

第九行、第十一行皆為同樣的輸出結果，只是為不同的表達方式。

```
Input a word:hi frinds
hi frinds

hi frinds
```

```c
#include<stdio.h>
int main(){
    char a[10]; //陣列有10個格子
    /*顯示*/
    printf("Input a word:");
    //讀取字串
    fgets(a, sizeof(a),stdin);
    //顯示字串
    puts(a); //puts顯示字串
    putchar('\n'); //跳行符號
    printf("%s",a); //printf顯示字串
    return 0;
}
```

# 上課內容

2/25(8)

這個程式的功能一台簡易的四則運算的計算機。

先定義整數a、b(如果要輸入小數的話，把int改成fioat就行了)

再來就是用scanf儲存使用者輸入的數字

再利用11～１４行的四則運算，就可以算出答案了

＋加
－減
＊乘
／除

```
Input a:5
Input b:3
a + b = 8
a - b = 2
a * b = 15
a / b = 1
```

```c
#include<stdio.h>
int main(){
    int a,b; //兩個整數a和b

    printf("Input a:"); //顯示
    scanf("%d", &a); //讀取數字並儲存到a

    printf("Input b:"); //顯示
    scanf("%d", &b); //讀取數字並儲存到b

    printf("a + b = %d\n", a + b); //顯示a和b相加的結果
    printf("a - b = %d\n", a - b); //顯示a和b相減的結果
    printf("a * b = %d\n", a * b); //顯示a和b相乘的結果
    printf("a / b = %d\n", a / b); //顯示a和b相除的結果

    return 0;
}
```

# 上課內容

2/25(9)

此乘是是用來運算小數的加法，第三行先定義兩個浮點數為ａ、ｂ（因為後面的ａ＋ｂ＝ｃ也會是小數，所以先定義）

再先利用４～７行先把想要計算的ａ、ｂ兩數存起來，再利用第九行的ｃ定義成ａ＋ｂ

而後第十一行就可以用來計算結果了，而該行裡的％.1f指的是，輸出的數字只有小數點後一位（因為放在第一個，所以第一個輸入的數字才會變）

```c
#include<stdio.h>
int main(){
    float a, b, c; //有兩個浮點數分別是a和b
    printf("Input a float number K:"); //顯示
    scanf("%f", &a); //讀取數字並儲存至a
    printf("Input a float number M:"); //顯示
    scanf("%f", &b); //讀取數字並儲存至b

    c = a + b; //a+b的結果儲存至c

    printf("K + M = %.1f + %.2f = %.3f\n", a ,b ,c);

    return 0;
}
```

```
Input a float number K:4.8987
Input a float number M:5.9634
K + M = 4.9 + 5.96 = 10.862
```

而%2.f也是同理。

# 上課內容

2/25(10)

此程式是用於字元和字串（b是以設定個10個字元來拼湊成字串）

第五～十一行就是指輸入ａ、ｂ兩個定義的東西再進行儲存。

第十三行就是用來輸出ａ、ｂ

```
Input a character:k
Input a word:hello
You have input 'k' and "hello
"
```

```c
#include<stdio.h>
int main(){
    char a, b[10];

    printf("Input a character:");
    a = getchar();

    fflush(stdin); //清除stdin緩存

    printf("Input a word:");
    fgets(b, sizeof(b), stdin);

    printf("You have input \'%c\' and \"%s\" ",a,b);
    // 要顯示單引號 \'
    // 要顯示雙引號 \"
    return 0;
    }
```

# 上課內容

2/25(11)

此程式是教簡單的算子判斷

>是大於的判斷
<是小於的判斷
>=是大於等於的判斷
<=是小於等於的判斷
==是等於的判斷
!=是不等於的判斷

```c
#include<stdio.h>
int main(){
    printf("10 > 5 \t\t %d\n", 10 > 5);     //顯示10大於5的結果，True
    printf("10 < 5 \t\t %d\n", 10 < 5);     //顯示10小於5的結果，False
    printf("10 >= 5 \t\t %d\n", 10 >= 5);   //顯示10大於等於5的結果，True
    printf("10 <= 5 \t\t %d\n", 10 <= 5);   //顯示10小於等於5的結果，False
    printf("10 == 5 \t\t %d\n", 10 == 5);   //顯示10等於5的結果，False
    printf("10 != 5 \t\t %d\n", 10 != 5);   //顯示10不等於5的結果，True

    return 0;
}
```

```
10 > 5          1
10 < 5          0
10 >= 5                 1
10 <= 5                 0
10 == 5                 0
10 != 5                 1
```

若是輸出的式子符合判斷，則會輸出 1
反之若是不符合判斷，則輸出 0

1 = true　　0 = false

# 上課內容

2/25(12)

```
Input a integer a:5
Input a integer b:6
Input a float c:7.58
Input a float d:9.36
Input a character e:C
Input a character f:K
5 > 6           0
5 < 6           1
5 >= 6          0
5 <= 6          1
5 == 6          0
5 != 6          1
7.580000 > 9.360000             0
7.580000 < 9.360000             1
7.580000 >= 9.360000            0
7.580000 <= 9.360000            1
7.580000 == 9.360000            0
7.580000 != 9.360000            1
C > K           0
C < K           1
C >= K          0
C <= K          1
C == K          0
C != K          1
```

這主要就是將數字進行比較,
而英文字也會對應到不同的數字

將運算子合併再一起使用

```c
#include<stdio.h>
int main(){
    int a,b;
    float c,d;
    char e,f;

    //輸入整數、浮點數和字元
    printf("Input a integer a:");scanf("%d", &a);
    printf("Input a integer b:");scanf("%d", &b);
    printf("Input a float c:");scanf("%f", &c);
    printf("Input a float d:");scanf("%f", &d);
    printf("Input a character e:");fflush(stdin);e = getchar();
    printf("Input a character f:");fflush(stdin);f = getchar();
    //顯示兩個整數的邏輯判斷
    printf("%d > %d \t\t %d\n", a, b, a > b);       //  >是大於的判斷
    printf("%d < %d \t\t %d\n", a, b, a < b);       //  <是小於的判斷
    printf("%d >= %d \t\t %d\n", a, b, a >= b);     //  >=是大於等於的判斷
    printf("%d <= %d \t\t %d\n", a, b, a <= b);     //  <=是小於等於的判斷
    printf("%d == %d \t\t %d\n", a, b, a == b);     //  ==是等於的判斷
    printf("%d != %d \t\t %d\n", a, b, a != b);     //  !=是不等於的判斷
    //顯示兩個浮點數的邏輯判斷
    printf("%f > %f \t\t %d\n", c, d, c > d);       //  >是大於的判斷
    printf("%f < %f \t\t %d\n", c, d, c < d);       //  <是小於的判斷
    printf("%f >= %f \t\t %d\n", c, d, c >= d);     //  >=是大於等於的判斷
    printf("%f <= %f \t\t %d\n", c, d, c <= d);     //  <=是小於等於的判斷
    printf("%f == %f \t\t %d\n", c, d, c == d);     //  ==是等於的判斷
    printf("%f != %f \t\t %d\n", c, d, c != d);     //  !=是不等於的判斷
    //顯示兩個字元的邏輯判斷
    printf("%c > %c \t\t %d\n", e, f, e > f);       //  >是大於的判斷
    printf("%c < %c \t\t %d\n", e, f, e < f);       //  <是小於的判斷
    printf("%c >= %c \t\t %d\n", e, f, e >= f);     //  >=是大於等於的判斷
    printf("%c <= %c \t\t %d\n", e, f, e <= f);     //  <=是小於等於的判斷
    printf("%c == %c \t\t %d\n", e, f, e == f);     //  ==是等於的判斷
    printf("%c != %c \t\t %d\n", e, f, e != f);     //  !=是不等於的判斷

    return 0;
}
```

# 上課心得



在這周的程式設計課程當中，我學到了不少的東西，就像是最基本的導入函式、輸出字元字串、基本四則運算 . . . . . .
也因此我在這周的課程當中也將基礎的概念學的至少有些概念，在距離自己第一次上這門課程的一個禮拜前，我在一些偶然之下開始學習了寫程式，雖然網路上都有許多課程教學，但因為我是自己去摸索寫程式，因此也時常遇到一些困難，而我藉由這堂課了解到了許多基本的相關概念，在老師的講解下也大致明白了寫程式的用途，像現在科技不斷的進步， AI已經成了人類生活中密不可分的一項技術，而 AI的誕生就是藉由著一行行的程式而 產出的。
不僅僅是如此，老師也有提到了其他的程式語言，像是 python和C++或是C語言，這些語言通常是現在幾個比較耳熟能詳的程式語言，尤其是現在最熱門的 python，在同時，老師也有講到了其實程式語言的最根本概念基本都是相同的，像是每個程式語言都會有函式、輸出、運算等等許多公式，只是不同語言的表達方式不同而已，而 python之所以熱門，是因為它是目前最簡單易懂上手的程式語言，有許多學校目前也都是以 python來教導學生。
在這堂課老師也有講到 APCS，一個鑑定程式語言的考試，免費報名的並且可以對高中升學有所幫助，因此老師也鼓勵我們盡量參與，畢竟能在升學的路上給自己增添新的技能特色。
總結來說，在這一周的課程中我弄明白了 C語言的相關基本用法，以及程式語言鑑定的 APCS，和程式語言如何運用之類的相關知識，解答到了我不少的疑惑，屬實是收穫滿滿。
第一次打出像第 16頁那種的很長很長的程式，心裡挺有成就感的。

# 上課內容

3/4(1)

右圖為運算子的用法，圖中：
|| 代表 OR(兩件其中一件為真即真)
&&代表 AND(兩件均為真即真,均為假即假)
! 代表 NOT(不是判斷結果的結果)
^^代表 XOR(兩件事均為真即假，均為假即為假，一真一假為真)

```
1  #include<stdio.h>
2  int main(){
3      int a,b,c;
4      //printf("Input number A:");
5      scanf("%d %d %d", &a, &b, &c);
6      printf("%d\n", a>b && a>c);     // 0 and 1 = 0
7      printf("%d\n", a>b || a>c);     // 0 or 1 = 1
8      printf("%d\n", !(a>b && a>c)); // not(0 and 1)  = not(0) = 1
9      printf("%d\n", !(a>b || a>c)); // not(0 or 1)  = not(1) = 0
10     return 0;
11 }
```

```
3
5
2
0
1
1
0
```

圖中第三行可知，先定義整數a,b,c再利用第五行的scanf進行儲存，先以a=3 b=5 c=2為舉例，六到九行則是輸出( )內的結果。
以第六行先比較a>b(3>5)=0(False) &&(AND) a>c(3>2)=1(True)可得兩輸出False和True也就是0和1，而&&(AND)則是代表兩件事都為真才是真，因此第六行為 0(False)
第七行比較a>b(3>5)=0(False) ||(OR) a>c(3>2)=1(True)可得兩輸出0(False)和1(True)，而||(OR)則代表兩件事情其中一件為真就是真，因此第七行輸出為 1(True)
第八行比較a>b(3>5)=0(False) &&(AND) a>c(3>2)=1(True)可輸出0(False)和1(True)，由&&可得輸出結果為0，但應!(NOT)而輸出結果變為1(NOT 0 即為 1)
第九行比較a>b(3>5)=0(False) ||(OR) a>c(3>2)=1(True)可輸出0(False)和1(True)，由||(OR)可知輸出結果為1，但因!(NOT)而輸出結果變為0(NOT 1 即為 0)

# 上課內容

3/4(2)

右兩圖是運算子的判斷法
五～八行對應的是AND的判斷方法(兩件事情均為真即真，均為假即假，一件事情為真，另一為假，則輸出 0以代表錯誤事件)
第五行兩件事均為0，因此輸出為0
六、七行其一件事為1，另則0，故輸出0
十一~十四行為OR的判斷方法(其中一件事情為真，即為真)
第十一行兩件事均為假，故輸出為 0
十二～十三行均有其中一件事為真，故輸出為1
十七到二十行為XOR的判斷法(兩件事均為真即假，均為假即為假，一真一假為真，簡單來說就是有互斥的效果。)
十七、二十行其一均為真，其一均為假，故輸出均為0
十八、十九行均為其一真另一假，輸出 1
而二三、二四行為NOT的判斷方法(不是結果的結果，非0即1)
故輸入0、1，輸出的結果為1、0

```
AND
0 AND 0          0
0 AND 1          0
1 AND 0          0
1 AND 1          1
OR
0 OR 0   0
0 OR 1   1
1 OR 0   1
1 OR 1   1
XOR
0 XOR 0          0
0 XOR 1          1
1 XOR 0          1
1 XOR 1          0
NOT
NOT 0    1
NOT 1    0
```

```c
#include<stdio.h>
int main(){
    //能單AND運算只需要1個" & "符號
    printf("AND\n");
    printf("0 AND 0 \t %d \n", 0 & 0);
    printf("0 AND 1 \t %d \n", 0 & 1);
    printf("1 AND 0 \t %d \n", 1 & 0);
    printf("1 AND 1 \t %d \n", 1 & 1);
    //能單OR運算只需要1個" | "符號
    printf("OR\n");
    printf("0 OR 0 \t %d \n", 0 | 0);
    printf("0 OR 1 \t %d \n", 0 | 1);
    printf("1 OR 0 \t %d \n", 1 | 0);
    printf("1 OR 1 \t %d \n", 1 | 1);
    //XOR運算是" ^ "符號
    printf("XOR\n");
    printf("0 XOR 0 \t %d \n", 0 ^ 0);
    printf("0 XOR 1 \t %d \n", 0 ^ 1);
    printf("1 XOR 0 \t %d \n", 1 ^ 0);
    printf("1 XOR 1 \t %d \n", 1 ^ 1);
    //NOT運算前面加上" ! "符號
    printf("NOT\n");
    printf("NOT 0 \t %d \n", !0);
    printf("NOT 1 \t %d \n", !1);
    return 0;
}
```

# 上課內容

3/4(3)

右圖的程式演示了自增和自減的運算符
第三行先定義了變數i=0
第四行則輸出了變數i，即為先前定義的0
第六行使得變數i+1，即為0+1，此時i等於1
第七行輸出了變數i，而因為前面的i+1，故輸出1
第八行使得變數i+1也就是在1+1，此時的i為2
第九行輸出了變數i，由前面的定義i等於2，輸出2
第十行使得變數i+1，即為2+1，此時i=3
第十一行則是輸出i，此時i為3，故輸出3
第十三行則是跳行的符號
第十五行定義了i-1，也就是從原本的3減1，i為2
第十六則是輸出先前的i-1，故輸出為2
第十七行使得變數i-1，也就是2-1，i=1
第十八行輸出i，此時的i=1，故輸出為1
第十九行使得變數i-1，即為1-1，i=0
第二十行則是輸出變數i，此時i=0，故輸出0
i+1可表達成:i++,++i(有先後問題),i=i+1
i-1可表達成:i--,--i(有先後問題),i=i-1
這些表達方法時常會用在迴數當中

```
i = 0
i = 1
i = 2
i = 3

i = 2
i = 1
i = 0
```

```c
#include<stdio.h>
int main(){
    int i = 0;
    printf("i = %d\n", i); // i = 0

    i = i + 1; // 這時候 i = 1
    printf("i = %d\n", i); // i = 1
    i++; // i = i + 1;
    printf("i = %d\n", i); // i = 2
    ++i; // i = i + 1;
    printf("i = %d\n", i); // i = 3

    putchar('\n');

    i = i - 1; // 這時候 i = 2
    printf("i = %d\n", i); // i = 2
    i--; // i = i - 1;
    printf("i = %d\n", i); // i = 1
    --i; // i = i - 1;
    printf("i = %d\n", i); // i = 0

    return 0;
}
```

# 上課內容



右圖程式為if的條件語句，當輸入條件符合假設的條件的話，則會輸出對應結果，若否則會對應至else if(可多次使用)中繼續判斷，如對應其條件的話就會輸出其對應結果，若為否，則會進入到else進行最後判斷，以此判斷相符條件。
第三行定義了兩個變數，一個是 input，即為輸入的數字，另一個為 number，
為要猜測到的數字。經由四、五行來輸入儲存input後，開始判斷條件。
若輸入為40，則會對應到第七行，此時的input=number，會輸出第八行的結果
若輸入為50，會對應不到第一條件的 if而繼續到第十一行的else if進行判斷，
而此時input>number(50>40)故輸出第十二行的結果。
若輸入為30，此時input<number，則會對應到第十五行的else(30<40)，則輸出第十六行的結果並結束判斷。

```c
#include<stdio.h>
int main(){
    int input=0, number=40;
    printf("Input a number:");
    scanf("%d",&input);

    if(input == number){
        printf("你猜對了%d\n",number);
    }

    else if(input > number){
        printf("你輸入數值比%d大%d\n", number, input-number);
    }

    else{
        printf("你輸入數值比%d小%d\n", number, number-input);
    }
    return 0;
}
```

```
Input a number:40
你猜對了40
```

```
Input a number:50
你輸入數值比40大10
```

```
Input a number:30
你輸入數值比40小10
```

# 上課內容

3/4(5)

此程式主要表達的是如用用 if條件和分數範圍來判斷
第三~五行輸入一個變數並儲存
第七～十一行表示正確用法
在表達分數範圍的時候，並需要用&&(AND)，表示有達成兩著條件，就以輸入60來說理應輸出要為C，對的判斷在第十行，也就是分數小於70但大於等於60，也就是輸出C
但在錯誤的判斷中，因為在兩者的條件用的是 ||(OR)也就是符合其中一條件即為解，故在第十六行中，滿足了其中一項條件(60<80)，因此輸出為B，但不符合理應的輸出C。
總結來說，當用到範圍判斷時，會依據程序不同而有不同用法。

```c
#include<stdio.h>
int main(){
    int score=0;
    printf("Input a score:");
    scanf("%d",&score);

    printf("(對的)");
    if(score>=80) printf("學生的分數是A\n");
    else if ((score>=70) && (score<80)) printf("學生的分數是B\n");
    else if ((score>=60) && (score<70)) printf("學生的分數是C\n");
    else printf("學生的分數不及格F\n");

    //(錯的)
    printf("(錯的)");
    if(score>=80) printf("學生的分數是A\n");
    else if (score>=70 || score<80) printf("學生的分數是B\n");
    else if (score>=60 || score<70) printf("學生的分數是C\n");
    else printf("學生的分數不及格F\n");

    return 0;
}
```

```
Input a score:60
(對的)學生的分數是C
(錯的)學生的分數是B
```

# 上課內容



此程式使用了一個用法 switch，為一種流程控制陳述式，也就是根據變數的值來執行不同的代碼區塊。
第三～六行先定義一個變數 score並輸入儲存。
第七行則是先用了兩個條件來判斷，也就是此變數需要大於等於0且小於等於10，若否則會跳到第二十四行的輸出結果
，而若輸入範圍在兩條件當中，則會開始判斷，第八行的Switch就是先依照輸入的變數，來去執行對應到的代碼區塊(case)當中，假設輸入的變數為 5，則此時的switch就會對應到第二十一行當中，並且輸出其對應結果。而當輸出完後，也就是會結束了此程式，但為了避免將第十二行後的printf內容中的文字輸出，則要使用到 break來中止程式運作，在輸出完前面提到的 switch對應到的case 5結果，也就是輸出完第十二行後，在下一行接一個 break就可以中斷程式的運算以免將之後的 printf內容輸出。

```
Input a score (0~10):5
5
學生的分數不及格F
```

```c
#include<stdio.h>
int main(){
    int score = 0;
    printf("Input a score (0~10):");
    scanf("%d",&score);
    printf("%d\n", score);
    if((score>=0)&&(score<=10)){
        switch(score){
            case 10:
            case 9:
            case 8:
            printf("學生的分數是A\n");
            break;
            case 7:
            printf("學生的分數是B\n");
            break;
            case 6:
            printf("學生的分數是C\n");
            break;
            default:
            printf("學生的分數不及格F\n");
        }
    }
    else{
        printf("你輸入的數值不在0到10的範圍內\n");
    }

    return 0;
}
```

# 上課內容

3/4(7)

右圖的程式是延續第23頁評分機制的函式。
第三、四行主要是讓使用者輸入字元（評分的等級），以此去讓後續程式能繼續判斷。
第五行的程式主要是清除一些未讀取的資料，以避免影響結果。
第八行的switch語句，會根據變數值來執行不同的代碼區塊。
像是輸入了第九～十四行的case內的字元，就會跳出第十五、十六行的結果，簡單來說就是switch是一個分支，而case是其內的更多分支。
而第十七行的break就像是用來停止前面的case(九～十四行），以讓程式不會繼續執行到後面的部分（十五、十六行），然後再去裡用switch語句繼續執行十七行後的程式。
例如使用者若是使用了第十八行～第二十一行的 case裡的字元，就不會去運行到十五、十六行，直接去輸出二十二、二十三行。
而第二十五行的default就是用於沒有出現在case內的特定字，轉而去輸出其他結果，有點類似 else。

```
Input a character (A~E):A
Wonderful.
It feels so great!
```

```
Input a character (A~E):D
It so sad.
It so sad.
```

```
Input a character (A~E):H
Invalid character.
Invalid character.
Invalid character.
```

```c
#include<stdio.h>
int main(){
    char input;
    printf("Input a character (A~E):");
    fflush(stdin);
    input = getchar();

    switch(input){
        case 'A':
        case 'a':
        case 'B':
        case 'b':
        case 'C':
        case 'c':
        printf("Wonderful.\n");
        printf("It feels so great!\n");
        break;
        case 'D':
        case 'd':
        case 'E':
        case 'e':
        printf("It so sad.\n");
        printf("It so sad.\n");
        break;
        default:
        printf("Invalid character.\n");
        printf("Invalid character.\n");
        printf("Invalid character.\n");
    }
    return 0;
}
```

# 上課內容

3/4(8)

右圖的程式是一種迴圈，迴圈在程式語言中是一種常見的控制流程，主要的作用是使得代碼多次運行，通常是會運行到特定的次數，直到不符合迴圈中的條件，則會停止運行迴圈，繼續進行之後的代碼。

```c
#include<stdio.h>
int main(){
    //for迴圈
    for(int i=0; i<10; i++){
        printf("%d\t",i);
    }
    return 0;
}
```

```
0	1	2	3	4	5	6	7	8	9
```

# 上課內容

3/4(9)

右圖的程式是主要是以兩個迴圈所組合成的，又稱作「雙重迴圈」，如果一個迴圈算是一個條件的話，那雙重迴圈就算是條件中的條件，先達成第一個迴圈中的條件後，再去進入到第二個迴圈運行，若是代碼都有在兩個迴圈的條件中，則會輸出之後設定的程式。

第四行為第一個主要的迴圈，先制訂一個數為i=0，這個再之後會成為決定運算次數的數字，而先以i=0開始套入迴圈中，也就是來到了第五行，輸出i，也就是0。

而再到了第六行的迴圈，也就是到了第二個迴圈，首先一樣先定義了j=0以做為初始數字，再送到了第七行，但因為此時的i=j=0，不符合第二個迴圈中的條件(i<j)，因此不輸出*而後再經過第九行進行了一個跳行的動作(此時i=0，j=0)結束了兩個迴圈的第一次運算，但因為主要的第一個迴圈的條件都還符合(i<10)因此再重新回到第四行的第一個迴圈當中，而此時的i+1，因此現在的i為1，再進行到第二個迴圈，有符合條件j<i(0<1)，因此輸出一個*，而此時j+1=1，不符合條件，於是不繼續執行第二個迴圈(每當第一個迴圈重新執行時，j都會以0繼續開始第二迴圈)

而再回到第一個迴圈，此時i=1有符合<10的條件，因此i=1+1=2，經過第五行輸出一個2後，又到了第二個迴圈，此時(i=2，j=0)

有符合j<i因此輸出一個*，並且j+1=1，因為還符合條件，所以程式會繼續在第二個迴圈中運行，因為j<i因此再輸出一個*，所以總共輸出**，以此類推。直到i=10的時候結束運算。

```c
#include<stdio.h>
int main(){
    //for迴圈
    for(int i=0; i<10; i++){ //第一層迴圈在計算數數
        printf("%d\t",i);
        for(int j=0; j<i; j++){ //第二層迴圈是列印*
            printf("*");
        }
        putchar('\n');
    }
return 0;
}
```

```
0
1       *
2       **
3       ***
4       ****
5       *****
6       ******
7       *******
8       ********
9       *********
```

# 上課內容

3/4(10)

右圖所要說明的是while迴圈，也就是先根據前面輸入或輸出的結果來判斷，如果有符合while迴圈中的條件，則會以該輸出或結果來進行while迴圈內的內容，其特性在於需要不符合while中的條件，才會繼續進行之後的代碼，否則會一直循會在while迴圈中。

第三～五行是定義一個數字為input並且儲存，而若此定義數字有符合while迴圈中的條件，則開始進行第八～十行的程式。

這邊我以輸入input=25，因為符合while的條件（25<0，25>50），因此不執行while內的運作，直接跳到第十二行輸出Great。

以輸入input=90，因為符合while的條件（90<0 or 90>50），因此執行while內的運作，開始進行第八～十行的程式。

以輸入input=-90，因為符合while的條件（-90<0 or -90>50），因此執行while內的運作，開始進行第八～十行的程式。

```c
#include<stdio.h>
int main(){
    int input;
    printf("\nInput a number (0~50):");
    scanf("%d",&input);
    // while迴圈
    while((input<0)||(input>50)){ // OR的判斷用||
        printf("\nYour input number is invalid.");
        printf("\nInput a number (0~50):");
        scanf("%d",&input);
    }
    printf("\nGreat.");

    return 0;
}
```

```
Input a number (0~50):25

Great.
```

```
Input a number (0~50):90

Your input number is invalid.
Input a number (0~50):-90

Your input number is invalid.
Input a number (0~50):25

Great.
```

# 上課內容

3/4(11)

右圖為APCS考題中的一個題目。
第三行先定義了數列A的內容為8,7,6,5,4,3,2,1。
第四行利用迴圈性質，讓i=0,i=1,i=2…i=7，其目的是讓數列中的每個數字都能經過第六行print出來。
第八~十四行定義了一個迴圈，其作用是排序數列中的 內容。
第九行為布林值，若是A[j]>A[j+1](如果第j個數字大於第j+1個數字)，則會進行第十~十二行的內容。

第十~十二行以數字舉例，先假設 j=0，則會達成第九行的條件(8>7)。
第十行會讓A[0]=A[0]+A[1]，對應數列就是A[0]=8+7=15
此時的A[0]=15
第十一行會使A[1]=A[0]-A[1]，指A[1]=15-7=8
此時的A[1]=8
第十二行會使A[0]=A[0]-A[1]，指A[0]=15-8=7
此時的A[0]=7。
以此類推，直至沒有達成迴圈或者布林 值的條件。

```c
#include<stdio.h>
int main(){
    int A[8]={8,7,6,5,4,3,2,1};
    for(int i=0; i<8; i++){
        for(int k=0; k<8; k++){
            printf("%d\t", A[k]);
        }
        for(int j=i; j<7; j++){
            if(A[j] > A[j+1]){
                A[j] = A[j] + A[j+1];
                A[j+1] = A[j] - A[j+1];
                A[j] = A[j] - A[j+1];
            }
        }
        putchar('\n');
    }
    return 0;
}
```

```
8   7   6   5   4   3   2   1
7   6   5   4   3   2   1   8
7   5   4   3   2   1   6   8
7   5   3   2   1   4   6   8
7   5   3   1   2   4   6   8
7   5   3   1   2   4   6   8
7   5   3   1   2   4   6   8
7   5   3   1   2   4   6   8
```

# 上課內容



此程式為數列與迴圈的使用，
第三行先定義了三個整數變數 i,n,A[100]，而其中A是長度為100的數列。
經由第四行(假設n=6)，則就會儲存n=6。
接下來進入到迴圈內，首先
i=0，則A[0]=0，輸出0後，i=i+1=1。
因為滿足條件i不等於n(i≠n)。
i=1再進入迴圈中，第五行 i=i+1=2，則
A[2]=2，輸出2後，i=i+1=3，因為滿足條件i不等於n(i≠n)，再次進入迴圈。此時 i=3。
i=3進入迴圈後i=i+1=4，A[4]=4，則輸出4，而後i=i+1=5，在進入迴圈後i=6。
因迴圈條件不滿足(i≠6)因此退出迴圈，結束運算。

```c
#include<stdio.h>
int main(){
    int i, n, A[100];
    scanf("%d",&n);
    for(i=0; i!=n; i=i+1){
        A[i] = i;
        printf("%d ",A[i]);
        i = i + 1;
    }
    return 0;
}
```

```
6
0 2 4
```

# 上課內容

3/4(13)

右圖為簡單的迴圈運算，首先第三行先定義 i, j=0, 而開始進行第四行的迴圈, 此時 i=0, j=0, 而接著進行到第五行 , j=i+1(0+1)=1, 到第六行後輸出(i,j)=(0,1)。而就以繼續迴圈, 經過第四行後 i+j=(0+1)=1, 再接著到第五行j=(i+1)1+1=2, 輸出(i,j)=(1,2) , 依此類推……, 而直到i=>128的時候就會不符合迴圈條件(i<128)而結束迴圈。

```c
#include<stdio.h>
int main(){
    int i, j=0;
    for(i=0; i<128; i=i+j){
        j=i+1;
        printf("%d %d\n", i, j);
    }
    return 0;
}
```

```
0 1
1 2
3 4
7 8
15 16
31 32
63 64
127 128
```

# 上課內容



在第一層迴圈中，當i=0時，會無法滿足第二層迴圈的條件j<i(0<0)，因為無輸出所以再回到第一層迴圈。
此時i符合第一層迴圈條件i<5(0<5)因此i+1=0+1=1
當i=1時，送到第二層迴圈，有符合條件j<i，因此執行第二層迴圈的程式(第五行)，也就是輸出[i+j]=[1+0]=[1]。而若在第二層迴圈繼續執行的話會不合條件j<i(2<1)，因此回到第一層迴圈。
當i=2時，送進第二層迴圈，此時則會輸出[i+j]=[2+0]=[2]。
當i=3時，送進第二層迴圈，會輸出[i+j]=[3+0]=[3]，但因為還符合第二層迴圈條件因此可以繼續運作(2<3)，此時i=3，j=2，所以繼續執行第五行，則會輸出[i+j]=[3+2]=[5]。
當i=4時，送進第二層會圈，會輸出[i+j]=[4+0]=[4]，但因為還符合第二層迴圈條件因此可以繼續運作(2<4)，此時i=4，j=2，所以繼續執行第五行，則會輸出[i+j]=[4+2]=[6]。
而當i=5不符合第一層迴圈的條件i<5(5<5)，因此結束迴圈

```c
#include<stdio.h>
int main(){
    for (int i=0; i<5; i=i+1){
        for (int j=0; j<i; j=j+2){
            printf("[%d]",i+j);
        }
    }
    return 0;
}
```

```
[1][2][3][5][4][6]
```

# 自我心得



這周的課程也同樣教了許多概念，但最主要讓我感到困難的迴圈配數列、排序法的部分(第二十八、二十九頁)，這兩個部份我重複試了許多次，第二十八頁的排序用法我花了很久才能明白，雙重迴圈配上數列讓我一開始看不懂，我詢問過了老師，也去網路上看了這種題目的講解，雖然大致能明白它的運作原理，但若是要依據程式寫出每步的結果和運行作用，讓我實在不知道該從何下手，我個人在試著解釋那部分時，也時常因為弄不明白而將簡報內容重複修修改改了很多次，我也有試著將這段程式丟進AI讓他去試著運算，但得出來的答案(1, 2, 3, 4, 5, 6, 7, 8)卻跟實際運作的輸出結果(7, 5, 3, 1, 2, 4, 6, 8)並不一樣，我在想或許是AI將它當作成了泡沫排序法，也因此我又重複地去請教老師，翻查大量講解的影片，才能大概理解。

而二十九頁的數列迴圈也是因為運算出來的答案跟老師原本運算的答案不同，而花費了許多時間，所以我只好去詢問老師以解決問題。

# 上課內容



右圖程式是運用代號與迴圈

第三～五行分別是數列。分別為整數、小數、字元的數列，長度為10。

第六行定義整數j=0，用於後面的迴圈。

第八~十行為for迴圈，讓輸出i為(1,2,3,4,5,6,7,8,9)

第十二～十四行為while迴圈，用迴圈來放置小數及字元。

第十七～十九行為為for迴圈，用來輸出上面兩個迴圈的結果 。

ascii為編碼，不同的符號都會有其對應的數字代號。

```c
#include<stdio.h>
int main(){
    int number[10];
    float score[10];
    char ascii[10];
    int j=0;

    for(int i=0; i<10; i++){
        number[i] = i;
    }

    while(j < 10){
        score[j] = j;
        ascii[j] = j + 65;
    }
    j=0;
    for(int i=0; i<10; i++){
        printf("%d\t%f\t%c \n", number[i], score[i], ascii[i]);
    }
    return 0;
}
```

```
0       0.000000        A
1       1.000000        B
2       2.000000        C
3       3.000000        D
4       4.000000        E
5       5.000000        F
6       6.000000        G
7       7.000000        H
8       8.000000        I
9       9.000000        J
```

# 上課內容

3/11(2)

右圖為輸出二維數列的程式
第二、三行定義了兩個常數。用於後面的行列
第五行是定義了二維數列 abc的行列大小，為2*10。
第七～九行的雙重迴圈用來計算數列的第 i行的第j列，例如i=0，j=0。則
abc[0][0]=(0*1)*(0*2)=0,
這邊定義j=0是因為要計算矩陣的第一行。
而當i>row時則跳脫迴圈，來到之後的雙層迴圈，
也就是第十二行～第十四行的迴圈，這個迴圈和上一個的雙重迴圈一樣，但是是用來計算矩陣的第二行，
然後再輸出矩陣abc[i][j]。
第十五、十六行用來跳格、跳行。

```c
#include<stdio.h>
#define ROW 2
#define COLUMN 10

int main(){
    int abc[ROW][COLUMN];
    for(int i=0; i<ROW; i++){
        for(int j=0; j<COLUMN; j++){
            abc[i][j]=(i*1)*(j*2);
        }
    }
    for(int i=0; i<ROW; i++){
        for(int j=0; j<COLUMN; j++){
            printf("%d\t",abc[i][j]);
        }
        putchar('\n');
        printf("\n");
    }
    return 0;
}
```

```
0	0	0	0	0	0	0	0	0	0

0	2	4	6	8	10	12	14	16	18
```

# 上課內容

3/11(3)

右圖程式主要用一種方式將字元矩陣的字打出來，這邊會提到一個概念為「初始化」，像是第三行可以看到一個被定義的字列，但沒有定義其長度，就會根據原本的字列 內的字數來當作初始化的長度（12）。
第五～七行的迴圈用來輸出
Good Morning
而第八行是用來清除緩衝區的資料，讓後面程式可以讀取新的資料。
像是輸出Good Moring之後，可以按其後面輸入一筆新的資料，在按下Enter來讓其讀取，假設輸入了hello，則就會讓原本定義的Text的內容變成hello。
這樣再經過後面的程式（第十二行～第十五行）這樣就可以輸出
hello

```c
#include<stdio.h>
int main(){
    char text[] = {'G','o','o','d',' ','M','o','r','n','i','n','g'};
    int length = sizeof(text)/sizeof(text[0]);
    for(int i=0;i<length;i++){
        printf("%c",text[i]);
    }
    fflush(stdin);
    for(int i=0;i<length;i++){
        scanf("%c",&text[i]);
    }
    printf("\n\n");
    for(int i=0;i<length;i++){
        printf("%c",text[i]);
    }
    return 0;
}
```

```
Good Morning
hello
```

# 上課內容

3/11(4)

右圖程式主要是以不同方式來得出相同輸出，第二行的程式為一個指令，在此程式中是用字列的位置去尋找其對應輸出。
第五行定義一個整數長度，其值為text的長度除以char的長度，也就是28/1=28，用來表示text的長度，作用於第七行。
而在經過第五～八行的迴圈，就會輸出Hello, good morning everyone。
第九行用來讓輸出結果跳行。
第十行則是輸出text的內容Hello, good morning everyone。
第十一行用來讓輸出結果跳行。
第十二行就是上述提到的，利用字列位置去尋找出其對應輸出，假設要尋找的是第十個字，則text[10]就是第十個字d。

```c
#include<stdio.h>
#include<string.h>
int main(){
    char text[] = "hello, good morning everyone.";
    int length = sizeof(text)/sizeof(char);
    for (int i=0; i<length; i++){
        printf("%c", text[i]);
    }
    putchar('\n');
    printf("%s",text);
    putchar('\n');
    printf("%c", text[10]);
    return 0;
}
```

```
hello, good morning everyone.
hello, good morning everyone.
d
```

# 上課內容



右圖程式，主要是先輸入一串字，而後會輸出其字串長度。
設輸入的字串為such a beautiful day，也就是buf內的內容。
第三行先定義BUF為80，其作用是將後面的數列長度設置為80。
第六行是輸出括號內的字，而put跟print的差別，就是put無法放入變數。
第七行主要是用來輸入字串。
第八行則是定義了一個 size_t類型的變數 Length，為空結束前的長度。
第九行則是定義整數length2的長度為buf的長度/char的長度=80/1=80
第十行輸出buf。
第十一行輸出buf，但因為是用put，因此會在遇到空結束時停止，也就只輸出空格前的 such。
第十二、十三行則是輸出length、length2，也就是前面得到的4、80。

```c
#include<stdio.h>
#include<string.h>
#define BUF 80
int main(){
    char buf[BUF];
    puts("Please input a word...");
    scanf("%s", buf);
    size_t length = strlen(buf);
    int length2 = sizeof(buf)/sizeof(char);
    printf("%s\n", buf);
    puts(buf);
    printf("Length of buf = %d\n", length);
    printf("Length of buf = %d\n", length2);
    return 0;
}
```

```
Please input a word...
such a beautiful day
such
such
Length of buf = 4
Length of buf = 80
```

# 上課內容

3/11(6)

右圖程式主要說明的是strcpy用法，可以將一個陣列的內容複製給其他陣列。
這邊假設先輸入how are you
第三行先定義了一個LENGTH為100
第五行宣告了一個陣列buf的長度為100。
第六行則是輸出括號內的句子。
第七行則是將輸入的句子儲存在buf，總長度不能超過LENGTH-1的句子(第一個字會從0開始算。)
第九行則定義一個整數lenOfPart1為strlen(buf)也就是輸入的buf長度。
第十行為定義字元陣列abc的長度為lenOfPart1，也就是和buf一樣的長度。
第十一行則是讓字元陣列buf的內容複製給陣列abc，所以此時的陣列buf、abc有著同樣內容。
第十三、十四行就會輸出兩個的內容，因為內容一樣，所以輸出的結果也會一樣。

```c
#include<stdio.h>
#include<string.h>
#define LENGTH 100
int main(){
    char buf[LENGTH];
    puts("Please input a sentence...");
    fgets(buf, LENGTH, stdin);

    int lenOfPart1 = strlen(buf);
    char abc[lenOfPart1];
    strcpy(abc, buf);

    printf("buf = %s\n", buf);
    printf("abc = %s\n", abc);
    return 0;
}
```

```
Please input a sentence...
how  are you
buf = how  are you

abc = how  are you
```

# 上課內容

3/11(7)

右圖程式主要是將兩個句子結合。
第四、五行先定義了兩個字串的內容。
第七、八行則是先定義了整數 len，用於表示陣列abc的長度。
第十行所用到的memset，主要是讓字串abc的內容都變成空結束，也就是 \0，其主要功能就是為了讓後面的abc、str能順利連結，若是沒有使用memset，則會讓abc內未定義的內容與str互相衝突。
而第十一行會讓陣列 abc連結str1，可是因為原本abc沒有內容，所以輸出的結果就為Hello,everyone.(為陣列abc現在的內容)
第十二行是在abc後面再加入str2，則會輸出Hello,everyone.Good morning.

```c
#include<stdio.h>
#include<string.h>
int main(){
    char str1[] = "Hello, everyone.";
    char str2[] = "Good morning.";

    int len = strlen(str1) + strlen(str2) + 1;
    char abc[len];

    memset(abc, '\0', len);
    strcat(abc, str1);
    strcat(abc, str2);

    printf("%s\n", abc);
    return 0;
}
```

```
Hello, everyone.Good morning.
```

# 上課內容

3/11(8)

右圖的運用到了數列及布林值。
先定義了LEN為80，用於表示後面的陣列長度。
第五、六行定義了password、buf的陣列。
第七行則是輸出括號內的內容、輸入內容。
第八行則是儲存了輸入的內容。
第九行則是運用strncmp比較了password、buf的長度，相同的話為0，buf後面的6表示比較前六個數字，若是前六個數字相同的話，則會進入到第十行，反之則進入到的十三行。（因為只比較前六個，所以只須前六個相同就好，後面數字不同也不會被比較到。）

```c
#include<stdio.h>
#include<string.h>
#define LEN 80
int main(){
    char password[]="123456";
    char buf[LEN];
    puts("Please enter your password...");
    fgets(buf, LEN, stdin);
    if(strncmp(password, buf, 6) == 0){
        puts("Welcome...");
    }
    else{
        puts("Password Error...");
    }
    return 0;
}
```

```
Please enter your password...
12345678
Welcome...
PS C:\Users\user\OneDrive\桌面\C> cd
Please enter your password...
123456
Welcome...
PS C:\Users\user\OneDrive\桌面\C> cd
Please enter your password...
822963
Password Error...
```

# 上課內容

3/11(9)

右圖程式主要是用來搜尋字串 內的特定字元。
第三行先定義LEN為80，用以後面的陣列長度。
第五行先定義兩個長度為 80的陣列。
第七～十一行則是用來輸入字以及儲存輸入 內容。
第十四行則是用來去除多餘的空結束，確保不會影響到後續程式的運作。
第十六行的loc為一個指標，而strstr則是用來搜尋source和search內所要找的字，假設沒有找到的話，loc＝＝NULL就會進入到第十八行，然後輸出第十九行的內容(說明位置)。
假設找到的話就會進入到第二十一行後輸出第二十二行。舉例來 說
How are you，尋找a，則再第四個位置，則回傳位置。
若要尋找z，則找不到，就會輸出第二十二行。

```
Input any sente
how are you
Input your search...
a
Found in the 4th place
PS C:\Users\user\OneDrive\
Input any sentence...
how are you
Input your search...
z
Cannot find matched...
```

```c
#include<stdio.h>
#include<string.h>
#define LEN 80
int main(){
    char source[LEN], search[LEN];

    puts("Input any sentence...");
    fgets(source, LEN, stdin);

    puts("Input your search...");
    fgets(search, LEN, stdin);

    //去除search最後的換行符號
    search[strlen(search)-1] = '\0';

    char *loc = strstr(source, search); // *loc 是指標

    if(loc == NULL){
        puts("Cannot find matched...");
    }
    else{
        printf("Found in the %dth place\n", loc - source);
    }

    return 0;
}
```

# 上課內容

3/11(10)

右圖程式是用來用矩陣來搜尋 內容。
第二、三行用來定義 ROW、COL，用來表示後面的矩陣行列。
第六~八行則是定義ArrayNumber矩陣的的數字。
第十一～第十八行則是用來 print出矩陣。
第二十一行先定義整數 search。
第二十二、二十三行用來輸出字串及輸入 內容並儲存，其輸入內容就是使用者要尋找的 內容。
第二十五行到第二十九行則是運用迴圈，先讓迴圈內的數字不斷尋找，直到找到了輸入的數字。
就會輸出第二十九行的 內容。
若是沒有找到的話，就會執行第三十行。

```c
#include<stdio.h>
#define ROW 2
#define COL 5
int main(){
    int i=0, j=0;
    int ArrayNumber[ROW][COL]={          //矩形二維的陣列
        {1, 2, 3, 4, 5},
        {123, 345, 567, 789, 1234}
    };

    while(i<ROW){
        while(j<COL){                    //將二維陣列print出
            printf("%d\t", ArrayNumber[i][j]);
            j++;
        }
        putchar('\n');
        i++;
        j=0;
    }

    int search;
    puts("Which column do you search?"); //搜尋陣列中的數字
    scanf("%d", &search);

    for(i=0; i<COL; i++){
        if(search==ArrayNumber[1][i]){
            printf("There is in %dth box.\n", i+1);
        }
    }
    printf("End...\n");
    return 0;
}
```

```
1       2       3       4       5
123     345     567     789     1234
Which column do you search?
345
There is in 2th box.
End...
```

# 上課內容

3/11(11)

此程式用來尋找被賦值的變數，在記憶體中的位置。
第三～五行將一些資料賦值。
第七~十六行則是輸出那些資料在記憶體的位置，其位置是指，當變數被賦值的時候，會透過編譯器在記憶體中儲存被賦值的變數的位置(用法為%p)。

```c
#include<stdio.h>
int main(){
    int num1=10, num2=567, num3=520;
    float num4=14, num5=3.144, num6=2.14;
    char char1='A', char2='b', char3='\n';
    //print出變數和它們在記憶體的位置
    printf("num1 = %d\tnum2 = %d\tnum3 = %d\n", num1, num2, num3);
    printf("Address of num1 = %p\tnum2 = %p\tnum3 = %p\n", &num1, &num2, &num3);
    //putchar('\n');putchar('\n');
    //print出浮點數和它們在記憶體的位置
    printf("num4 = %f\tnum5 = %f\tnum6 = %f\n", num4, num5, num6);
    printf("Address of num4 = %p\tnum5 = %p\tnum6 = %p\n", &num4, &num5, &num6);

    //print出字元和它們在記憶體的位置
    printf("char1 = %c\tchar2 = %c\tchar3 = %c\n", char1, char2, char3);
    printf("Address of char1 = %p\tchar2 = %p\tchar3 = %p\n", &char1, &char2, &char3);

    return 0;
}
```

```
num1 = 10       num2 = 567      num3 = 520
Address of num1 = 0061FF1C      num2 = 0061FF18 num3 = 0061FF14
num4 = 14.000000        num5 = 3.144000 num6 = 2.140000
Address of num4 = 0061FF10      num5 = 0061FF0C num6 = 0061FF08
char1 = A       char2 = b       char3 =

Address of char1 = 0061FF07     char2 = 0061FF06        char3 = 0061FF05
```

# 上課內容

3/11(12)

此程式和上一個程式有些異曲同工之處，但右圖的程式的程式多新增了指標的用法。
第三行先定義了一個變數 n=140。
第四、五行則是利用 n的記憶體位置，把指標變數*p、*q的值回傳n的記憶體。
第六~八行則是輸出其值
第九、十行則是輸出兩個變數的記憶體位置，但因為前面將p的值是訂為n的記憶體位置，故其輸出結果一樣。
我另外加了第八行的程式用來對比，以證明指標變數賦值後的輸出結果。
至於位甚麼第四、五行的指標要用 &來協助定義，則是因為這邊的指標變數是用來儲存 n的記憶體位置。

```c
#include<stdio.h>
int main(){
    int n = 10;
    int *p = &n;
    int *q = &n;
    printf("n = %d\n", n);
    printf("*p = %d\n", *p);
    printf("*q = %d\n", *q);
    printf("Address n = %p\n", &n);
    printf("Address p = %p\n", p);
    return 0;
}
```

```
n = 10
*p = 10
*q = 10
Address n = 0061FF14
Address p = 0061FF14
```

# 上課內容

3/11(13)

右圖是利用指標複製值給其他變數，這邊我另外增加了第九~十一行，讓數值可以做更改。

第三行先是賦予了變數的值。
第四行是將*p的值複製成了num的值。
第五行再用*p的值複製給num2。
第九、十行輸出括號內的字。
第十二行則是用來儲存要更改的值。
假設要更改的值為1234，其儲存的位置為num1，所以現在num1的值為1234。
則到了第十四行則是再將num2的值變成*p的值，因此現在num2的值也是1234，經過第十五行，輸出其三個的值，就會成功達到改變值的效果。

```c
#include<stdio.h>
int main(){
    int num1, num2 = 520;
    int *p = &num1;
    *p = num2;

    printf("Num1 = %d\nNum2 = %d\n*p = %d\n", num1, num2, *p);

    printf("Enter a new value for num1: ");
    puts("After pointer changed...");

    scanf("%d", &num1);

    num2 = *p;
    printf("Num1 = %d\nNum2 = %d\n*p = %d\n", num1, num2, *p);

    return 0;
}
```

```
Num1 = 520
Num2 = 520
*p = 520
Enter a new value for num1: After pointer changed...
1234
Num1 = 1234
Num2 = 1234
*p = 1234
```

# 上課內容

3/11(15)

右圖的程式主要是在說明指數型指標用法，原本老師教的方法是要展現錯誤的用法，但我這邊先將其改成正確用法，後續再補充其錯誤用法。

第三、四行先定義了小數和指數的值。
第五、六行則是將兩個指標的記憶體位置存為前面定義的兩個數字的位置。
第七～十四行則是像上一頁一樣，均說明了當數值改變時，也會對應到不同的記憶體位置。

這邊補充原本的錯誤用法，原本是直接將第五、六行的指標賦值為0、1000，但首先那兩個指標是用於記憶體的位置，還有0、1000沒辦法當作有效的地址使用，因此讓程式一開始出錯，只不過在到後續的數值改變時，因為其賦於的兩個值還是會作用，因此第七~十四行的記憶體位置仍會有其位置。

```c
#include<stdio.h>
int main(){
    float n = 0.001;
    double B =100;
    float *p = &n;
    double *q =&B;
    printf("Address of p\t%p\n", p);
    printf("Address of p+1\t%p\n", p+1);
    printf("Address of p+2\t%p\n", p+2);

    printf("Address of q\t%p\n", q);
    printf("Address of q+1\t%p\n", q+1);
    printf("Address of q+2\t%p\n", q+2);
    return 0;
}
```

```
Address of p    0061FF14
Address of p+1  0061FF18
Address of p+2  0061FF1C
Address of q    0061FF08
Address of q+1  0061FF10
Address of q+2  0061FF18
```

# 上課內容

3/11(14)

右圖程式則是一個很簡單的變數及其記憶體位置的概念。

第三行先定義了指標 *p=0。

第四行則是輸出 p=0的記憶體位置。

第五行輸出 p+1=1的記憶體位置。

第六行輸出 p+2=2的記憶體位置。

這主要的概念就是指不同的變數數 值在編譯器記憶體當中都會有不同的編號，可能會因使用者的電腦而不同，這部分我經過老師詢問後，他給我了以下的解答。

char:1位元組，int/float:四位元組。

double:8位元組。

```c
#include<stdio.h>
int main(){
    int *p = 0;
    printf("Address of p\t%p\n", p);
    printf("Address of p+1\t%p\n", p+1);
    printf("Address of p+2\t%p\n", p+2);
    return 0;
}
```

```
Address of p    00000000
Address of p+1  00000004
Address of p+2  00000008
```

# 上課內容

3/11(16)

```c
#include<stdio.h>
int main(){
    int abc[10]={0};
    printf("address of abc[]:\t\t%p\n",abc);

    for(int i=0; i<10; i++){
        printf("&abc[%d]:\t%p\n", i, &abc[i]);
    }
    return 0;
}
```

右圖的程式增加了迴圈及陣列的用法。
其中第三行先使得數列中的數 值均為0,
也就是[0,0,0,0,0,0,0,0,0,0]。

第四行則是輸出陣列的記憶體位置。

第六、七行則是利用迴圈，讓陣列 abc的位置一直改變，可以看到從 i=0 ~10的輸出結果。

```
address of abc[]:               0061FEF4
&abc[0]:        0061FEF4
&abc[1]:        0061FEF8
&abc[2]:        0061FEFC
&abc[3]:        0061FF00
&abc[4]:        0061FF04
&abc[5]:        0061FF08
&abc[6]:        0061FF0C
&abc[7]:        0061FF10
&abc[8]:        0061FF14
&abc[9]:        0061FF18
```

# 上課內容

3/11(17)

右圖為講述函式回傳的用法的程式。
第二、三行先定義了一個函式 a( )，並且回傳為777，因此現在函式a( )為777。

第六、七行先定義了一個函式 f( )，目前f( )是空函式，還未有餒榮，但七行回傳了 a( )函式，因為a( )函式=777，再回傳給給 f( )，所以現在 f( )函式的值為777

第十~十四行來到了主程式，其中第十一行先用整數 abc 呼叫f( )，其意思就是abc=f( )，而此刻的f( )函式值為777，所以目前abc的值也同為777

再經過第十二行的print輸出abc的值，就可以得到其值為777。

```c
#include<stdio.h>
int a(){
    return 777;
}

int f(){
    return a(); //回傳a()
}

int main(){
    int abc = f(); //abc呼叫f()
    printf("abc = %d\n", abc);
    return 0;
}
```

```
abc = 777
```

# 上課內容

3/11(18)

右圖的程式是利用函數值的不同以得到不同的回傳。

第二、三行先定義兩個函數 f、f2，分別為整數函數、小數函數。

第五~九行則是會依據回傳的數值不同來回傳出不同的輸出，假設數值為整數，則回傳第六行，若數值為小數，則回傳第十行。

而要判斷的地方則是在第十三～二十一行的主程式
第十六～十八行函數為整數，則輸出第六行。
第十九行函數則為小數，則輸出第十行。

```c
#include<stdio.h>
void f(int); // f函數的原型
void f2(float); // f2函數的原型

void f(int){
    printf("This is recalled f(int).\n");
}

void f2(float){
    printf("This is recalled f(float).\n");
}

int main(){
    int number = 520;
    float number2 = 1.314;
    f(number);
    f(0);
    f(10);
    f2(number2);
    return 0;
}

```

```
This is recalled f(int).
This is recalled f(int).
This is recalled f(int).
This is recalled f(float).
```

# 上課內容

3/11(19)

此右圖程式是以呼叫函數以輸出結果。

第二行先定義了一個加法函式，其函式內容就為第三行的(a+b)。

同理，第六行定義了一個減法的函式，其內容就是第七行回傳的(a-b)。

來到第十行的主程式內，先定義了兩個整數x=1520，y=777。
第十二行則是輸出add函式的內容，也就是a+b=1520+777=2297
第十三行則是輸出minus的函式內容，也就是a-b=1520-777=743。

```c
#include<stdio.h>
int add(int a, int b){ //a = x = 1520; b = y = 777
    return (a + b);
}

int minus(int a, int b){ //a = x = 1520; b = y = 777
    return (a - b);
}

int main(){
    int x = 1520, y = 777;
    printf("x + y = %d\n", add(x, y));
    printf("x - y = %d\n", minus(x, y));

    return 0;
}
```

```
x + y = 2297
x - y = 743
```

# 自我心得



這周課程同樣教了許多東西，像是指標、函式回傳、記憶體位置，還有句子的不同表達方法。

而我在其中也遇到了不少困難，一開始在指標的部分有些聽不懂，尤其是輸出的記憶體位置是一串看不懂的，看似亂碼時則有規則的編碼時，雖然內心掀起了興趣，因為那些編碼就很像有時候在網站或者是在某些地方能看見的編碼，也是藉著這次課堂我才明白到原來這些就是所謂的記憶體編碼。

而且我在試著跑程式時，也遇到了一些困難，像是程式明明沒有出錯，但我運用的程式執行軟體卻一直顯示錯誤，我在這部分卡了好幾個小時，甚至想直接試著改掉整串程式，雖然最後還是會顯示錯誤，只不過我發現到或許只是因為某些緩存或者資料互相衝突，所以我用了C online這個網站協助我，我驚訝的發現到，原本跑不動的程式可以運作了。也是因此我解決了這次的困難(有次我發現原本使用的軟體運作不了，也是利用C online解決。)

# 上課內容

4/1(1)

右圖的程式為簡單的變數運算。
第三行定義了三個變數 i、j、k。

第四～七行則是輸出括號 內的內容。

第九~十二行同樣是輸出括號 內的內容，只不過這邊再將i、j、k分別加1。
這邊要注意加一的用法
i=i+1會使得後面程式的j都會變成i+1。
++k也會使得後面程式的k都變成k+1。
但使用j+1的話，則後面程式的j仍然是j。

按照上述的結果，就可以發現到第十四 ~第十六行輸出的結果就會有差別，那正是因為用法的不同，所造成的差別。

```
i = 0
j = 0
k = 0
*****
i = 1
j = 1
k = 1
*****
i = 1
j = 0
k = 1
```

```c
#include<stdio.h>
int main(){
    int i = 0, j = 0, k = 0;
    printf("i = %d \n", i);
    printf("j = %d \n", j);
    printf("k = %d \n", k);
    printf("*****\n");

    printf("i = %d \n", i = i + 1);
    printf("j = %d \n", j+1);
    printf("k = %d \n", ++k);
    printf("*****\n");

    printf("i = %d \n", i);
    printf("j = %d \n", j);
    printf("k = %d \n", k);

    return 0;
}
```

# 上課內容

4/1(2)

右圖的程式說明了函式與參數的用法。

第三行先定義了一個函式 increment。

第四行定義其函式為 a+12，所以後面傳入的數值會被+12。

第八～十六行則是主程式。
第九行定義了變數 x ＝ 100。
第十行輸出了 x的值，也就是100。
第十一行則是將 x帶入到了函式 increment。
所以這邊輸出的值會是x+12=100+12=112。
第十二行則是再輸出了一次 x，但因為這邊不會受到函式影響，所以值仍是100。
第十三行則是定義 x等於x進入函式後的結果，所以此時的x=12+100=112。
第十四行則輸出 x的值，因為上面先定義了 x的值，所以現在輸出的x值為112。

```c
#include<stdio.h>

int increment(int a){
    a = a + 12;
    return a;
}

int main(){
    int x = 100;
    printf("x = %d\n", x);
    printf("increment(x) = %d\n", increment(x));
    printf("x = %d\n", x); // 100
    x = increment(x); // x = 112
    printf("x = %d\n", x); // 112
    return 0;
}
```

```
x = 100
increment(x) = 112
x = 100
x = 112
```

# 上課內容

4/1(3)

右圖的程式新增了指數的用法。
假設輸入數字為10
第二～三行先定義了指數的函式，這邊的程式是主要是輸入的數字的平方

第六~九行則是用於指數的數字，也就是10的多少次方，其之後算出來的數字就是d，運用迴圈的原理讓數字能跟指數配合，而總結來說這邊的函式是b的c次方。

第十四～二十七行為主程式。
第十五行先定義兩個變數num、p。
第十七～二十一行則是用來儲存輸入的資料。
第二十三行則是輸出num(10)的平方，其用運到的函式為第三行定義的平方。
第二十四行則是輸出10的p次方，如果輸入三的話就是十的三次方，就是1000。

```c
#include<stdio.h>
int power2(int a){
    return (a * a);
}

int power(int b, int c){
    int d = 1;
    for(int i = 0; i < c; i++){
        d = d * b;
    }
    return d;
}

int main(){
    int num = 0, p = 0;

    printf("Enter a number: "); // 輸入數值 num
    scanf("%d", &num);

    printf("Enter number's power: "); // 輸入次方 p
    scanf("%d", &p);

    printf("Square of %d is %d \n", num, power2(num));
    printf("Power %d of %d is %d \n", p, num, power(num, p));

    return 0;
}
```

```
Enter a number: 10
Enter number's power: 3
Square of 10 is 100
Power 3 of 10 is 1000
```

# 上課內容

4/1(4)

右圖的程式同樣為指數用法，只是與第五十六頁不同的地方，為此程式是利用程式內的函式庫的程式。

第四行先定義兩個變數num、p。

第六～十行則是將輸入的資料儲存。

第十二行後半段則是利用指數(pow)內括號的方式來表達(num的p次方)。

```c
#include<stdio.h>
#include<math.h> // 數學函式庫 Library
int main(){
    int num = 0, p = 0;

    printf("Enter a number: "); // 輸入數值 num
    scanf("%d", &num);

    printf("Enter number's power: "); // 輸入次方 p
    scanf("%d", &p);

    printf("Power %d of %d is %.0f\n", p, num, pow(num, p));
    return 0;
}
```

```
Enter a number: 10
Enter number's power: 3
Power 3 of 10 is 1000
```

# 上課內容

4/1(5)

右圖的程式為除法後的餘數用法。

第三行先定義函式 gcd(除法)。
第六行定義兩個變數 m、n=0。這邊主要是先定義兩個變數讓編譯器知道變數的存在。

第八～十三行主要為儲存兩個輸入的數字（m、n)。

第十六行則是帶入函式 gcd(m/n)。
第十七～二十一行的布林值主要用於判斷函數。
若 n 能整除m，則進行第十七行，輸出就為 0(餘數)。

第二十、二十一行則是用於 n沒辦法整除m的情況，也就是餘數不等於 0，則會進入第二十、二十一行，就會回傳n除以m的函數(m%n)之後會運行
至第十一行(輸出結果)。

```
Enter two number: 100
3
GCD: 1
PS C:\Users\user\OneDrive
Enter two number: 100
4
GCD: 4
```

```c
#include<stdio.h>

int gcd(int, int);

int main(){
    int m = 0, n = 0;

    printf("Enter two number: ");
    scanf("%d %d", &m, &n);

    printf("GCD: %d\n", gcd(m, n));

    return 0;
}

int gcd(int m, int n){
    if(n == 0){
        return m;
    }
    else{
        return gcd(n, m % n); // m除以n的餘數
    }
}
```

# 上課內容

4/1(6)

右圖為一個利用餘數判斷算法的程式。
假設帶入的值為4(i=4)。

f(4)先到第三行判斷i>0，此事為真，再進第四行判斷，4/2%2(4除以2後的餘數再除以2的值為0)，此事為真，運行至第五行，f(4-2)*4，回傳f(2)*4。

再進入到第三行判斷2>0(f2)，此事為真，再進入到第四行判斷(2/2)%2=0，此事為假(餘數為1)，故進行至第八行，f(2-2)*(-2)=0，回傳f(0)*-2。

再進入到第三行判斷，此時為f(0)，i=0，不符合判斷諞建，所以回傳0，f(0)=1，結束這部分的運算。

第十六~十八行進入至主程式，第十七行主要是輸出f(4)的值，而
f(4)=f(2)*4=f(0)*4*(-2)==1*4*(-2)=8，所以最終的輸出結果為-8。

```c
#include<stdio.h>
int f(int i){
    if(i>0){
        if(((i/2)%2)==0){
            return f(i-2)*i;
        }
        else{
            return f(i-2)*(-i);
        }
    }
    else{
        return 1;
    }
}

int main(){
    printf("%d\n", f(4));
    return 0;
}
```

```
PS C:\Users
-8
```

# 上課內容

4/1(7)

右圖主要為設定數值去運行的程式，先以f(14)為假設。

進入到第十二行的判斷(14<2)此事為假，所以至第十六行進行運算，得到的輸出結果為14+f(7)。

所以目前再利用f(7)帶入運算，先至第十二行(7<2)此事為假，所以至第十六行進行運算，得到的輸出結果為 7+f(3)，若是i的數字無法整除的話，則用商進行運算。

因此目前再利用f(3)帶入運算，先至第十二行(3<2)，此事為假，所以再進第十六行進行運算，則出的結果為 3+f(1)。

f(1)再進行至第十二行判斷(1<2)此事為真，所以進入到第十三行，得到的輸出結果為1。

接下來統整一下
，f(14)=14+f(7)=14+7+f(3)=14+7+3+1=25。
所以f(14)=25，經過第五行後就會輸出其結果。

```c
#include<stdio.h>
int f(int);

int main(){
    printf("%d\n", f(14));
    printf("%d\n", f(10));
    printf("%d\n", f(6));
    return 0;
}

int f(int n){
    if(n<2){
        return n;
    }
    else{
        return (n+f(n/2));
    }
}
```

```
25
18
10
PS C:\Users\u
25
18
10
```

# 上課內容

4/1(8)

右圖的程式主要是用於交換數字的程式。

第二行先定義了兩個變數 x=2，y3=。

第四～十二行定義的兩個式子，都是讓 x、y的數字進行交換，其運作方式如下：
第五行先定義一個變數 tem，他的值=x，也就是等於2。
第六行定義x的值會等於y值，所以現在的x為3。
第七行再定義y的值等於tem，所以現在的y值等於2。
這就成功地將兩數數 值交換。

第十四行到十八行為主程式，第十五行呼叫了 f2的函式，但此函式並非算式，而算是個資料（交換後的x、y）。來到第十六行後就可以開始計算輸出結果，因為交換後所以x=3，y=2。帶入到運算為：
(3-2)*(3+2)/2=1*5/2，算出來並非整數，但是輸出結果僅會是商，商等於 2，所以輸出為2。

```c
#include<stdio.h>
int x = 2, y = 3;
//交換
void f(int x, int y){
    int tem = x;
    x = y;
    y = tem;
}
void f2(){
    int tem = x;
    x = y;
    y = tem;
}
int main(){
    f2();
    printf("%d", (x-y)*(x+y)/2);
    return 0;
}
```

```
PS C:\Users\user\
2
```

# 上課內容

4/1(9)

右圖程式為設定數值去運算的程式，這邊先假設數值為10。
第二、三行先定義函式 f(n)，再定義一個變數 sum。
以變數為10開始進入計算
第四行(10<2)此事為假，所以進入到第七行。
第七行則是利用迴圈性質，將 i變為n後，再帶入進算。
第八行則是用來總和數字。
將n=10經由第八行計算，可得 sum=(1+2+ ... +10)=55。
再經由第十行， sum=55+f(10*2/3)=55+f(6)。n=6。

帶回第四行(6<2)此事為假，所以進入到第七行。
sum=55+(1+2+ ... 6)=55+21=76。
再經由第十行可得到 sum=76+f(2*6/3)=76+f(4)。n=4。

再帶回第四行(3<2)此事為假，所以進入到第七行。
sum=76+(1+2+3+4)=86。經由第十行可得 sum=86+f(2)。

再帶回第四行(2<2)此事為假，所以進入到第七行。
sum=86+(1+2)=89，再經第十行可得 sum=89+f(1)
帶回第四行(1<2)此事為真，所以 f(1)=0。
總結就會得到 f(10)=89，故第十五行輸出為 89

```c
#include<stdio.h>
int f(int n){
    int sum = 0;
    if (n<2){
        return 0;
    }
    for(int i=1; i<=n; i=i+1){
        sum = sum + i;
    }
    sum = sum + f(2*n/3);
    return sum;
}

int main(){
    printf("%d\n", f(10));
    printf("%d\n",2*2/3);
    return 0;
}
```

```
PS C:\Users\use
89
1
```

# 上課內容

4/1(10)

右圖為檔案複製的程式(63、64頁)。

第二行先輸入讀檔的函式庫，這樣才能使之後的程式運行。

第四～九行ㄒ先設定了一個主程式，第四行先設定了一個變數argc和一個檔案指標的字串矩陣 argv。
第五行定了一個指標 src，會用fopen.txt這個函式打開讀readfile.txt這個檔。
第六~九行為布林值，若是找不到此檔案的話，就會回傳第八行。

第十一行先定義了一個檔案指標 dest，會用fopen.txt這個函式打開writefile.txt檔。
第十二行也是個布林值，用來判斷此檔案是否存在，若是不存在的話則會輸出第十三行後回傳第十四行。

```c
#include<stdio.h>
#include<stdlib.h> //讀檔的函式庫

int main(int argc, char* argv[]){
    FILE* src = fopen("readfile.txt","r");
    if(!src){
        perror("Cannot open file.\n");
        return EXIT_FAILURE;
    }

    FILE* dest = fopen("writefile.txt", "w");
    if(!dest){
        perror("Cannot create file.\n");
        return EXIT_FAILURE;
    }

```

要先設置兩個檔， readfile、writefile
(讀取readfile檔複製到writefile)

# 上課內容



接續至第十七行，定義了一個變數 c。
第十八行則是先存取了 src的一個資料後賦值給c。
第十九行再將c的內容將資料儲存到dest檔案。
也就是說，此程式是利用函數和迴圈的方式，將readfile.txt的檔案一個一個字（避免一次讀檔太多導致資料出錯）複製給writefile.txt。

第二十二行則是用布林值判斷檔案是否有出錯（打不開或者讀取中發生異常），則會輸出第二十三行。

第二十六、二十七行用來關閉兩個檔案，避免檔案還在資料緩衝區，沒有寫進檔案，而造成後續出錯。

```
17    int c;
18    while ((c = fgetc(src)) != EOF){ //EOF=End of File
19        fputc(c, dest);
20    }
21
22    if(ferror(src) || ferror(dest)){
23        puts("Read-Write error!\n");
24    }
25
26    fclose(src);
27    fclose(dest);
28
29    return 0;
30
```

readfile.txt　writefile.txt ×

writefile.txt

```
1  Hello world!
2
3  How are you today?
4
```

# 上課內容

4/1(11)

右圖為BubbleSort排序法。
第二行定義PASS=60。
第三行導入泡沫排序法的函式。
第四行定義變數。
第五行為外層迴圈，主要目的是讓每個學生的分數跑過一遍。
第六行sp是用來紀錄交換過程
第七行為內層迴圈，主要是讓數字等於外層迴圈。
第八行為布林值，若是排序再第n個分數大於排在第n+1個的分數，就會使其交換。
第九行則是暫存數字，主要目的是讓
第十一~十三行的程式運行(交換數字)。
第十六行則是記錄過程，若sp=1代表沒有交換，sp=0代表有交換。

```
1  #include<stdio.h>
2  #define PASS 60 //定義PASS為 60，表示及格分數
3  void bubbleSort(int studentScore[], int studentNumber){
4      int i, j, k, t=1, Temp, sp;
5      for(int i=studentNumber-1; i>0; i--){
6          sp = 1;//定義變數
7          for(j=0; j<=i; j++){//內層迴圈
8              if(studentScore[j]>studentScore[j+1]){ //交換
9                  Temp = studentScore[j];
10
11                 studentScore[j] = studentScore[j+1];
12                 studentScore[j+1] = Temp;
13                 sp = 0;
14             }
15         }
16         if (sp==1) break;
17     }
18 }
```

# 上課內容



接續至第十九行，為主程式。
第二十行儲存學生人數。
第二十一行儲存學生分數。
第二十二行儲存不及格的最高者。
第二十三行儲存及格的最低者。
因為學生分數不可能會 -1或101，所以先設置兩數表述還未找到兩者的資料。
第二十五、二十六行用來輸入後儲存學生人數。
第二十八～三十行則是用來將學生的人數限制在1～20人。

第三十三行輸出括號內的字。
第三十四行用迴圈讀取每個學生分數。
第三十五行輸入學生分數後儲存。
第三十六行限制學生的分數在 0~100，若是違反則輸出第三十七行。若為合理範圍則至三十八行儲存。

```
19  int main(){
20      int studentNumber = 0;//學生人數
21      int studentScore[20];//學生分數
22      int score1 = -1;//score1是不及格分數最高者
23      int score2 = 101;//score2是及格分數最低者
24      //輸入學生人數(1~20)
25      printf("Enter number of student(1~20): ");
26      scanf("%d", &studentNumber);
27      //判斷學生人數(1~20)
28      while(studentNumber<1 || studentNumber>20){
29          printf("Enter number of student(1~20): ");
30          scanf("%d", &studentNumber);
31      }
32      //輸入學生的分數
33      printf("Enter the student's score: ");
34      for(int i=0; i<studentNumber; i++){
35          scanf("%d", &studentScore[i]);
36          while(studentScore[i]<0 || studentScore[i]>100){
37              printf("Invalid score. Please try again.\n");
38              scanf("%d", &studentScore[i]);
39          }
40      }
41      //小到大排序
```

# 上課內容



接續第四十二行，呼叫bubblesort泡沫排序法排序學生分數。
第四十四～五十一行，則是利用迴圈將學生分數輸出。（-1代表去除空結束）。
第五十二~五十九行同樣利用迴圈的性質找出學生分數。
第五十三、五十四行是找出合格的學生中分數最低的。然後儲存至score1。
第五十六、五十七行是找出不合格的學生中分數最高的。然後儲存score2。

```
42      bubbleSort(studentScore,studentNumber);
43      //輸出第一行
44      for(int i=0; i<studentNumber; i++){
45          if(i == studentNumber - 1){
46              printf("%d", studentScore[i]);
47          }
48          else{
49              printf("%d ", studentScore[i]);
50          }
51      }
52      for(int i=0; i<studentNumber; i++){
53          if(studentScore[i]<PASS && score1<studentScore[i]){
54              score1 = studentScore[i];
55          }
56          if(studentScore[i]>PASS && score2>studentScore[i]){
57              score2 = studentScore[i];
58          }
59      }
```

# 上課內容



接續第六十行，輸出 \n(跳行)。
第六十二行代表沒有人不及格(score1代表不及個最高分，沒有數字替換掉 -1代表所有人都及格)。
然後輸出第六十三行。

第六十五行則是若有數字代替掉 -1成為不及格分數最高的，則輸出此數字。

同理，第六十九行代表沒有及格分數最低(所有人都不及格，因此101的值沒被替換掉)。
然後輸出第七十行。

第七十二行代表若有數字被替換掉，則輸出第七十三行。

```
60      putchar('\n');
61      //輸出第二行
62      if(score1 == -1){
63          printf("best case\n");
64      }
65      else{
66          printf("%d\n", score1);
67      }
68      //輸出第三行
69      if(score2 == 101){
70          printf("worst case\n");
71      }
72      else{
73          printf("%d\n", score2);
74      }
75      return 0;
76  }
```

```
Enter number of student(1~20): 5
Enter the student's score: 12 45 78 96 50
12 45 50 78 96
50
78
```

# 上課心得

4/1

今天的課教了指標和函式的運用，同時也講到了演算法。
其中最讓我感到很有趣也很困難的是演算法，演算法在電腦中是一個非常重要的概念，演算法也算是個耳熟能詳的名詞。
第一次使用演算法的程式，我感到很新奇，覺得演算法真的是一個很神奇的概念，我在自己打演算法的程式時很常狀況百出，運行了好幾次程式仍就沒辦法成功，只不過有老師的協助，我才成功的讓演算法的程式能成功運用。老師在講解演算法的原理時，還利用了課堂上的同學身高作為資料來排序，不得不佩服老師的這種講解方法，這讓我能親自體驗到演算法在排序的過程中所處理的順序，就好像是我就是個正在被處理的資料，而因此我也才更能對於泡沫排序法有著更深刻的印象，也因此我在處理這份簡報時，能盡自己所能的講解有關泡沫排序法的每行程式，雖然在試圖講解演算法的過程中，我也遇到了不少困難，不知道該怎麼說明，或者不知道該程式是如何運行，為此我還去查閱了不少資料，來復的看了好幾次程式，才終於將報告給統整完。

# 上課前言

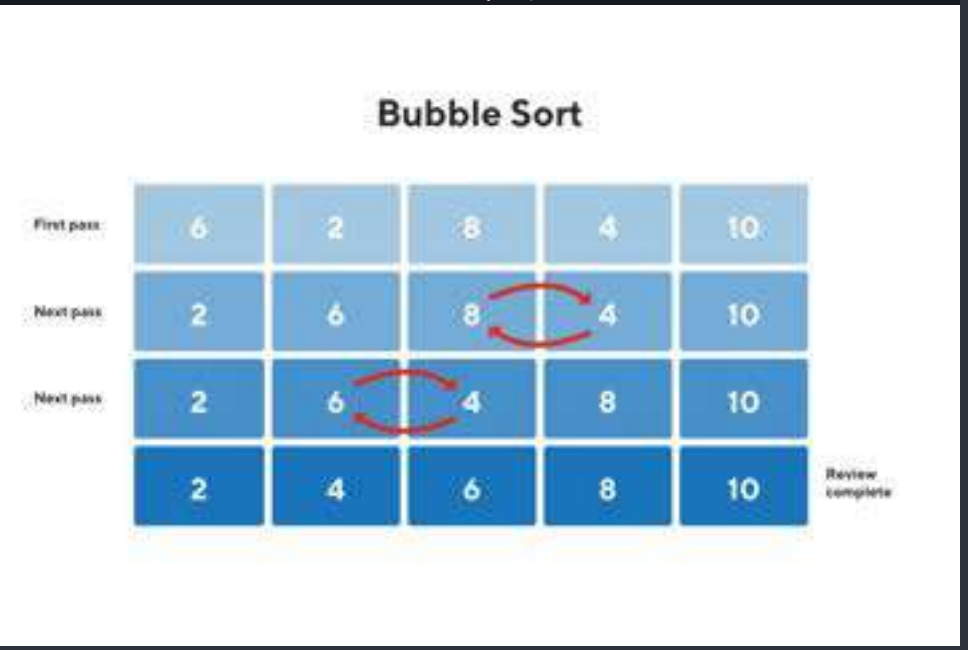


此次的課堂介紹了選擇排序法Selection sort,

後面會進行講解並且與泡沫排序法Bubble sort進行比較。

故後面簡報會先大概整理一下泡沫排序法的用法。

# 上課內容




圖片來源:https://reurl.cc/eD2X5R

此處先大概說明兩者運行模式:
泡沫排序法:相鄰兩數比較、交換。
選擇排序法:挑最大的去比較最後一個。

4/8(1)

圖片來源:https://reurl.cc/4ozQRV

# 上課內容

4/8(1)

```c
#include<stdio.h> /* 定義 printf 和 scanf */
#define SIZE 5 /* 0,1,2,3,4 */
void bubbleSort(int * const array, const int size); /* 定義 bubbleSort() 副程式 */
void swap(int *element1Ptr, int *element2Ptr); /* 定義 swap() 交換 */

/* 主程式 */
int main(){
    int array[SIZE]; /* 整數陣列 */
    printf("Input any %d numbers > ", SIZE);
    for (int i = 0; i < SIZE; ++i){ /* 讀取鍵盤數字並儲存至 array[] */
        scanf("%d", &array[i]);
    }
    bubbleSort(array, SIZE); /* 呼叫 bubbleSort() 副程式 */
    return 0;
} /* 結束主程式 */
```

此為bubbleSort的用法，主要是先儲存要排序的資料。

# 上課內容

```
17  /* 氣泡排序法 */
18  void bubbleSort(int * const array, const int size){
19      int pass; /* 第一層 for 迴圈，pass 計數 */
20      int j; /* 第二層 for 迴圈，記錄位置 */
21      /* for 迴圈控制排序的位置 */
22      for (pass = 0; pass < SIZE - 1; pass++){
23          /* for 迴圈進行比較 */
24          for (j = 0; j < SIZE - 1; j++){
25              /* print 出 array[] 陣列的數值 */
26              putchar('\n');
27              for(int i = 0; i < SIZE; ++i){
28                  printf("%d/", array[i]);
29              } /* 結束 for */
```

此部分是用迴圈性質輸出資料內容。

```
31                  /* 進行交換 */
32                  if(array[j] > array[j + 1]){
33                      swap(&array[j], &array[j + 1]); /* 呼叫 swap() 副程式 */
34                  } /* 結束 if */
35              } /* 結束第二層 for 迴圈 */
36          } /* 結束第一層 for 迴圈 */
37      } /* 結束 bubbleSort() 副程式 */
38
39      /* 交換 element1Ptr 和 element2Ptr 的數值 */
40      void swap(int *element1Ptr, int *element2Ptr){
41          int hold = *element1Ptr; /* 定義臨時的空間存放第一個數值 */
42          *element1Ptr = *element2Ptr; /* 第一個空間存放第二個數值 */
43          *element2Ptr = hold; /* 第二個空間存放第一個數值 */
44      } /* 結束 swap() 交換副程式 */
```

此部分是比較資料大小後，進行交換。

# 上課內容

4/8(2)

選擇排序法Selection Sort
第二行先定義了一個SIZE大小為五。
第三、四行定義了兩個函式。

第六行來到了主程式。
第七行定義矩陣array的大小為五。
第八行為輸入要排序的數字。
第九、十行則是利用迴圈性質讀取資料後儲存。
第十二行呼叫了選擇排序的函式。

```c
#include<stdio.h>
#define SIZE 5 /* 0,1,2,3,4 */
void selectionSort(int * const array, const int size);
void place_largest(int * const array, const int size);

int main(){
    int array[SIZE];
    printf("Input any %d numbers > ", SIZE);
    for (int i = 0; i < SIZE; ++i){
        scanf("%d", &array[i]);
    }
    selectionSort(array, SIZE);
    return 0;
}
```

# 上課內容



接續第十七行。
第十七行主要是定義選擇排序法的函式，用來排序資料，而資料就是括號內的資料，const是用來表示其輸入後的資料不可變（用來提升可讀性和安全性）。
第十八行用來跳格用。
第十九、二十行利用迴圈性質將輸入的資料輸出。

第二十二行size>1代表矩陣內還有元素，就會將此元素放置到最後一個（不會被放在已被忽略的元素後面），然後到了第二十四行，就會將該元素忽略。

```
16  /* 選擇排序法 */
17  void selectionSort(int * const array, const int size){
18      putchar('\n');
19      for (int i = 0; i < SIZE; ++i){
20          printf("%d/", array[i]);
21      }
22      if (size > 1){
23          place_largest(array, size);
24          selectionSort(array, size - 1);
25      } /* 結束 if */
26  }
27
```

# 上課內容



接續第二十九行，將最大值放到後面。
第三十、三十一行定義兩個變數，用於交換數字的暫存用。
第三十二行將最大的數值存入，而size-1代表最後一個位置。所以最大的數會在最後一個位置。
第三十四、三十五行代表若是遇到比size-1這個位置的元素還大的數值時，就會進行交換（size-1代表最後一個位置，用size-2代表從倒數第二個開始找）。

```
28  /* 找出最大值並丟到後面 */
29  void place_largest(int * const array, const int size){
30      int temp;
31      int j;
32      int max_index = size - 1; /* 記錄最大值的位置 */
33      for (j = size - 2; j >= 0; --j){
34          if (array[j] > array[max_index])
35              max_index = j;
36          }
37      }
38      if (max_index != size - 1){
39          temp = array[size - 1]; /* 暫時的空間存放最後一個數值 */
40          array[size - 1] = array[max_index];
41          array[max_index] = temp; /* 原本最大值的空間存放最後一個數值 */
42      } /* 結束 if */
43  } /* 結束 place_largest() 副程式 */
```

```
Input any 5 numbers > 1 5 6 9 8

1/5/6/9/8/
1/5/6/8/9/
1/5/6/8/9/
1/5/6/8/9/
1/5/6/8/9/
```

第三十八行為布林值，如果最大的數值位置不是在最後一個，則會運行第三十九～四十一行。第三十九行用 temp暫存最後一個位置的數值，第四十行在將最後一個位置改成最大的數 值，第四十一行在讓原本最後一個位置的數 值和被移動的數值交換。

此處以5、2、4、3舉例，依序從小排到大。

泡沫排序法：此程式比較時會重複走過多次要排序的數列，一次比較兩個相鄰的數值(元素)，如果排序有錯誤的話，就會交換過來，使得最大或是最小的數值可以排至最末端。

第一輪：先比較5、2，5>2，交換位置，2543
再比較5、4，5>4，交換位置，2453
再比較5、3，5>3，交換位置，2435

第二輪：因為2已經是最小的了(已確定2被排序完)，故從4開始比較
比較4、3，4>3，交換位置，2345，結束排序。

選擇排序法：從未排序的數列當中，選出最大或最小的數值，將其排至最末端，被排序後排在最後的值，就會暫時被忽略，然後在從剩下的數列中選擇再進行排序。

第一輪：先找出最大值(5)，將其排至最後面，2435

第二輪：找到除了5(已被忽略)以外的最大數字(4)，將其放在未排的數值最後面，2345，結束運算。

此處同樣以2、4、3、5舉例。
時間長度：也叫做時間複雜度，表示演算法所需的時間，通常數列長度會影響其 值。
以下用T表示比較次數，S比較交換次數，O表示最糟的情況下，演算法處理所需的時間，i代表每一趟所交換的次數。

泡沫排序法：總共需要比較(4-1)+(4-2)+(4-3)+(4-4)=6次，交換了(2+1+0+0)=3次。
若用n表示數列長度，則可以用公式表示 比較次數 和 交換次數。
比較次數：T(n)=(n-1)+(n-2)+ ... 2+1=n*(n-1)/2。
時間長度：O(n^2)。
交換次數：S(n)=O(n**2)或O(1)，會因數列初始排序而有差別。
選擇排序法，總共需要比較(4-0)+(4-1)+(4-2)+(4-3)=10次，交換了(1+1+1+0)=3次。
比較次數：T(n)=(n-i)+(n-i+1)+ ... +(n-n+i)=(n*n-i)/i。
時間長度=O(n^2)。
交換次數：S(n)=n-i=O(n)。
可以發現到兩個排序法的時間複雜度相同，但選擇排序法的交換次數比泡沫排序法的次數相對來說較少，故總結來說，選擇排序法的效率會稍微勝過泡沫排序法。

# 上課內容



右圖的程式用來判斷三角形。
第二行先定義三角形(三個邊)。

第四行來到主程式。
第五行先定義了三角形的邊。
第七、八行利用迴圈性質輸入後儲存。
第十二行為儲存三邊長的另一個做法。
第十五行～二十一行利用泡沫排序法的用法，將輸入的三個邊依照大小排好(用來後面計算三邊常關係。)
這邊也是利用了temp暫存，然後讓數字能夠互相交換。

```c
#include<stdio.h>
#define TRIANGLE 3

int main(){
    unsigned int triangle[TRIANGLE]; /* unsigned int = 正整數 */

    for (int i = 0; i < TRIANGLE; i++){
        scanf("%d", &triangle[i]);
    }

    /* 第二種 scanf 方法 */
    scanf("%d %d %d\n", &triangle[0], &triangle[1], &triangle[2]);

    /* 氣泡排序法 */
    for (int i = 0; i < TRIANGLE; i++){
        /* for 迴圈進行比較 */
        for (int j = 0; j < TRIANGLE - 1; j++){
            if (triangle[j] > triangle[j+1]){
                int temp = triangle[j]; /* 定義暫時的空間存放第一個數值 */
                triangle[j] = triangle[j + 1]; /* 第一個空間存放第二個數值 */
                triangle[j + 1] = temp; /* 第二個空間存放第一個數值 */
            }
        }
    }
```

```
PS C:\Users\u
3 9 2
2 3 9
No
PS C:\Users\u
4 3 2
2 3 4
Obtuse
PS C:\Users\u
5 3 4
3 4 5
Right
PS C:\Users\u
9 7 8
7 8 9
Acute
```

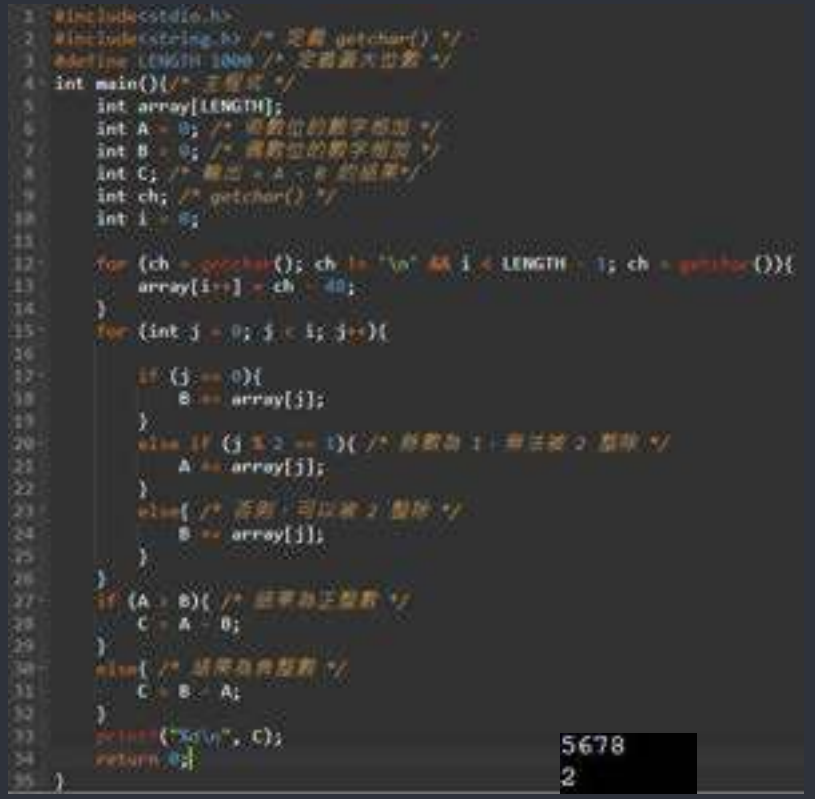
```
    printf("%d %d %d\n", triangle[0], triangle[1], triangle[2]);
    /* 第二行輸出 ，輸出三角形 */
    if (triangle[0] + triangle[1] <= triangle[2]){
        printf("No\n"); /* 無法構成三角形 */
    }
    else if ((triangle[0] * triangle[0]) + (triangle[1] * triangle[1]) < (triangle[2] * triangle[2])){
        printf("Obtuse\n"); /* 鈍角三角形 */
    }
    else if ((triangle[0] * triangle[0]) + (triangle[1] * triangle[1]) == (triangle[2] * triangle[2])){
        printf("Right\n"); /* 直角三角形 */
    }
    else if ((triangle[0] * triangle[0]) + (triangle[1] * triangle[1]) > (triangle[2] * triangle[2])){
        printf("Acute\n"); /* 銳角三角形 */
    }
    return 0;
}
```

第二十六行輸出已經排序好的三角形三個邊。第二十八 ~四十一行的布林值則會根據三個邊長來判斷三角形的種類。
若無法構成三角形，輸出第二十九行。若為鈍角三角形，輸出第三十二行。
若為直角三角形，輸出第三十五行。若為 銳角三角形，輸出第二十八行。

# 上課內容

4/8(4)

右圖為偶數位總和、奇數位總和相減的程式。
第二行是用來定義getchar的函示庫。
第三行先設置一個長度為1000的位數。
第五行定義了一個矩陣array為1000長度。
第六~十行定義了幾個變數。
A=奇數字總和。B=偶數字總和。
C=A-B的結果(還未被初始化)。
ch用來表示getchar輸入讀取的元素的ASCII值。
第十一行為迴圈，若是ch的值不等於跳行符號，
且小於lENGTH-1的長度，則會輸出第十三行，
就會儲存該位置的數值(-48是應為ASCII的編號問題，0~9在其中代表48~57。)
第十五～第二十六行是利用迴圈性質算過每個位置後將其判斷並存入A或B當中。
第十七、十八行是為了避免第0位數字被忽略。
第二十~二十六行用來判斷奇數位、偶數位。
再利用第二十七~三十四行輸出總和後的A、B數值相減後輸出。



# 上課內容

4/8(5)

右圖程式用來判斷輸入事件。
第三行定義三個正整數a b c。
第四行定義一個變數(後續用為儲存c的值。)
第五行定義變數error=0。
第七行輸入後a、b、c的值後儲存。

第九～十行判斷a、b，並進行轉換。
第十二～十四行，若a&(and)b為真，則輸出第十三行，否的話輸出第十四行，並繼續判斷下去。
當output=c，就會輸出其對應內容。
舉a=1 b=1 c=1來說
1and1為真，output=1，c=1。
1or1為真，output=1，c=1。
舉a=0 b=0 c=0來說
0and0為假，output=0，c=0。
0or0為假，output=0，c=0。
0xor0為假，output=0，c=0。
輸出and、or、xor，依此類推。

```
PS C:\Users\user\
1 1 1
AND
OR
PS C:\Users\user\
0 0 0
AND
OR
XOR
PS C:\Users\user\
1 0 1
OR
XOR
PS C:\Users\user\
2 3 4
IMPOSSIBLE
```

```c
#include<stdio.h>
int main(){
    unsigned int a, b, c;
    int output;
    int error = 0;

    scanf("%d %d %d", &a, &b, &c);

    if (a > 0) a = 1;
    if (b > 0) b = 1;

    output = a & b; /* a AND b */
    if (output == c) printf("AND\n");
    else error++;

    output = a | b; /* a OR b */
    if (output == c) printf("OR\n");
    else error++;

    output = a ^ b; /* a XOR b */
    if (output == c) printf("XOR\n");
    else error++;

    if (error == 3) printf("IMPOSSIBLE\n");

    return 0;
}
```

# 上課心得



本次課堂教了有關泡沫排序法(Bubble sort)和選擇排序法(Selection sort)的內容，排序法也就是演算法，在電腦科學上一個很重要的存在，演算法可以幫我們理解問題的結構和本質，並且找到有效率和邏輯的思路提供解決的方法。

這次教的程式似乎都比以往的長，看起來的確很複雜，並且在一個程式中會用到許多過去學到的東西，甚至還有些是數學的公式，所以在理解上要耗更多的時間，我也是花了數個小時去查閱了大量資料才了解，尤其是在演算法的部分。我從以前就很常聽到演算法這個詞，但一直沒認真的去了解這是甚麼，只看到了時間複雜度和一行我看不懂的公式，所以在這堂課聽到老師說會教我們演算法的概念和程式時，我滿懷了許多的期待和驚喜，沒想到自己有一天真的能學習到有關演算法的知識。為此我也好好的把握住了這次的機會好好學習，才發現到原來演算法有分那麼多種類別，其程式看似很長難以理解，但我還是努力的去了解每行程式的功能，雖然老師有提到時間複雜度的概念，但並沒有詳細說明，所以我還自己去網路上查了資料，並加入到了簡報的內容，也因此我才能把過去看不懂的公式，藉由著這次的機會去理解，而為了能詳細理解，我還自己帶了數字去慢慢推敲，其中就會算到演算法執行中的比較次數和交換次數，雖然耗費了無數時間，但我還是很高興自己能理解這些資料並整理到簡報當中。

# 上課內容

4/15(1)

右圖為利用輸入內容判斷事件後並輸出的程式

第二行將const char的類型重名為String。

第四～八行定義了一個結構Account，第五～七行定義了其內容。。

第十、十一行定義了一個函式，其作用式判斷輸入的accNumber後回傳其對應的帳戶。

```
1  #include<stdio.h>
2  typedef const char *String;
3
4  struct Account{
5      String id;
6      String name;
7      double balance;
8  };
9
10 void printAccount(int accNumber, struct Account acc){
11     printf("Account %d (%s, %s, %f)\n", accNumber, acc.id, acc.name, acc.balance);
12 }
13 int main(){
14     struct Account acc1, acc2, acc3;
15     int accNumber;
16
```

第十三行～第十五行來到了主程式。
第十四行是將帳戶編了其對應的數字1、2、3。
第十五行定義了變數accNumber，用來判斷其對應帳號。

# 上課內容



接續第十七行。
第十七行～二十七行分別創建了個別帳戶內的訊息。

第三十行用來輸入帳戶編號並儲存。

第三十一定義了個分支，會利用輸入的數字來輸出其對應的事件。

第三十二～三十五行就會依據輸入的數字去對應事件後輸出有關帳戶的資訊，break是代表執行該行程式後就不會在執行其他事件的程序。

```
17      acc1.id = "123-456-789"; // 第一個帳號的id
18      acc1.name = "Alice";  // 第一個帳號的名字
19      acc1.balance = 1000;  // 第一個帳號的餘額
20
21      acc2.id = "000-123-456"; // 第二個帳號的id
22      acc2.name = "Chrystal";  // 第二個帳號的名字
23      acc2.balance = 666;  // 第二個帳號的餘額
24
25      acc3.id = "000-456-789";  // 第三個帳號的id
26      acc3.name = "Catherine";  // 第三個帳號的名字
27      acc3.balance = 888;  // 第三個帳號的餘額
28
29      // 判斷帳號
30      scanf("%d", &accNumber); // 讀取帳號
31      switch(accNumber){
32          case 1: printAccount(accNumber, acc1); break;
33          case 2: printAccount(accNumber, acc2); break;
34          case 3: printAccount(accNumber, acc3); break;
35          default: printf("Account is not found...\n");
36      }
37      return 0;
38  }
```

```
PS C:\Users\user\OneDrive\桌面\C> cd "c:\Users
1
Account 1 (123-456-789, Alice, 1000.000000)
PS C:\Users\user\OneDrive\桌面\C> cd "c:\Users
2
Account 2 (000-123-456, Chrystal, 666.000000)
PS C:\Users\user\OneDrive\桌面\C> cd "c:\Users
3
Account 3 (000-456-789, Catherine, 888.000000)
PS C:\Users\user\OneDrive\桌面\C>
```

# 上課內容



右圖程式與上一個程式的內容大致相同，但新增了一些功能，下一頁會說明。

第二行同樣定義了字元矩陣char[]為字串。

第四~八行紀錄了Account的內容，也就是其構造。

第十、十一行定義了程式，主要是輸入一個值就能得到其對應數據。

```
1  #include<stdio.h>
2  typedef const char* String;
3  //定義Account的結構
4  struct Account{
5      String id; // 當下Account的id
6      String name; // 當下Account的名字(name)
7      int balance; // 當下Account的餘額(balance)
8  } Account;
9
10 void printAccount(int accNumber, struct Account acc){
11     printf("Account %d (%s, %s, %d)\n", accNumber + 1, acc.id, acc.name, acc.balance);
12 }
13
14 int main(){
15     struct Account acc[] = {
16         {"123-456-789", "Alice", 1000}, // 第一個帳號，acc[]位址 = 0
17         {"000-123-456", "Chrystal", 666}, // 第二個帳號，acc[]位址 = 1
18         {"000-456-789", "Catherine", 888}  // 第三個帳號，acc[]位址 = 2
19         };
20     int accNumber;
```

第十四行來到了主程式，第十五行用帳號矩陣（acc[0]、acc[1]、acc[2]為位置對應到了1、2、3）。
第二十行定義了變數accNumber，用來之後的回傳資料的判斷依據

# 上課內容



接續第二十六行。接下來的程式，
主旨是利用迴圈的性質，將所有的帳戶內容輸出，並且可以再根據所輸入的值再輸出其對應的內容，而後只要輸入0就可以結束程式，其程式與上一個程式的不同就是多了這些功能。

```
Account 1 (123-456-789, Alice, 1000)
Account 2 (000-123-456, Chrystal, 666)
Account 3 (000-456-789, Catherine, 888)

Enter an account to check (0 for EXIT...)> 1
Account 1 (123-456-789, Alice, 1000)

Enter an account to check (0 for EXIT...)> 2
Account 2 (000-123-456, Chrystal, 666)

Enter an account to check (0 for EXIT...)> 3
Account 3 (000-456-789, Catherine, 888)

Enter an account to check (0 for EXIT...)> 0
EXIT... end
```

```
26      //利用迴圈性質 列出所有帳號
27      for(int i = 0; i < 3; i++){
28          printAccount(i, acc[i]);
29      }
30
31      // 判斷帳號，輸入0就會退出
32      printf("\nEnter an account to check (0 for EXIT...)> ");
33      scanf("%d", &accNumber);
34
35      while(accNumber){
36          switch(accNumber){//輸入1，2，3就會對應到其事件
37              case 1: printAccount(accNumber - 1, acc[0]); break;
38              case 2: printAccount(accNumber - 1, acc[1]); break;
39              case 3: printAccount(accNumber - 1, acc[2]); break;
40              default: printf("Account is not found...\n");
41          } // 結束switch
42
43          printf("\nEnter an account to check (0 for EXIT...)> ");
44          scanf("%d", &accNumber);
45      } // 結束while迴圈
46
47      printf("EXIT... end\n");
48
49      return 0;
50  } // 結束主程式
```

# 上課內容



右圖程式主要是根據前兩個程式的雛型，所延伸出來的，有更多的功能可使用。

第一~九行同樣都為定義個字元矩陣
char[]為字串，然後定義帳號內容。

第十二~十六行為存款的程式
第十二行先定義存款的程式
第十三行代表，若是要存款的數字 <0，則輸出第十四行後結束。
若要存款數字>0，則至第十八行
第十八行表示帳號(acc)中的存款(balance)加上存款的數字。

第二十一行為提款的程式，同理，
第二十二行先定義提款的程式。
第二十三行判斷要提款的數字，若是大於帳戶內的存款，則輸出第二十四行後結束。若沒有的話則至第二十八行，將原本的存款數字減去提款的數字。

```
1  #include<stdio.h> // 定義printf()和scanf()
2
3  typedef const char* String; // 定義char[]為字串
4
5  typedef struct { // 定義Account{}
6      String id; // 當下Account的id
7      String name; // 當下Account的name
8      int balance; // 當下Account的balance
9  } Account; // 結束Account{}
10
11 // 定義存款deposit()副程式
12 void deposit(Account *acc, int moneymoneymoney){
13     if(moneymoneymoney <= 0){ // 判斷輸入的金額是否小於0
14         puts("Invalid input...\nExit...\n");
15         return;
16     } // 結束if判斷
17
18     (acc -> balance) += moneymoneymoney;
19 } //結束deposit()副程式
20
21 // 定義提款withdraw()副程式
22 void withdraw(Account *acc, int moneymoneymoney){
23     if(moneymoneymoney > (acc -> balance)){
24         puts("Insufficient balance...\nExit...\n");
25         return;
26     }
27
28     (acc -> balance) -= moneymoneymoney;
29 } // 結束withdraw()副程式
```

# 上課內容



```
30
31  // 定義printAccount()副程式，print出帳號最新的資訊
32  void printAccount(Account *acc){
33      printf("Account(%s, %s, %d)\n", acc->id, acc->name, acc->balance);
34  } // 結束printAccount()副程式
35
36  // 主程式
37  int main(){
38      Account acc = {"1234-5678", "Catherine", 100000};
39      int action; // 輸入 - 1為存款，2為提款，3為顯示帳號最新資訊，0為退出
40      int moneyMoneyMoney; // 輸入 - 存款/提款金額
41
42      printAccount(&acc); // print出帳號最新的資訊
43
44      printf("\nSelect action:\n\tDeposit >> 1\t\tWithdraw >> 2\t\tCheck >> 3\t\tExit >> 0\n > ");
45      scanf("%d", &action);//用來儲存action的輸入值並暫存。
```

接續至第三十行。第三十一行～第三十四行為輸出帳號的最新資訊。
第三十七～四十行則是定義了帳號內的基本資料，第三十九行先定義了變數 action，用來代表退出0，存款1，提款2，顯示帳號最新資訊3，以上的動作，第四十行則是定義變數，代表提款、存款金額。
第四十二行會顯示帳號最新資訊。
第四十四行輸出括號內容，主要是告訴使用者可以用哪些動作。
第四十五行會存入使用者輸入的數字，並且判斷後續該進行那些動作。

```
46
47      while(action){//一種迴圈結構，可以用來重複執行一段程式碼，直到某個條件不成立為止。
48          switch(action){//分支，利用輸入數字去對應其事件。
49              case 1: // 存款
50                  printf("Enter the amount of deposit > ");
51                  scanf("%d", &moneymoneymoney);
52                  deposit(&acc, moneymoneymoney); // 呼叫存款deposit()副程式
53                  printAccount(&acc); // print出帳號最新的資訊
54                  break;//結束此程式，跳至第六十六行。
55              case 2: // 提款
56                  printf("Enter the amount of withdraw > ");
57                  scanf("%d", &moneymoneymoney);
58                  withdraw(&acc, moneymoneymoney); // 呼叫提款withdraw()副程式
59                  printAccount(&acc); // print出帳號最新的資訊
60                  break;//結束此程式，跳至第六十六行。
61              case 3: // 顯示帳號最新的資訊
62                  printAccount(&acc); // print出帳號最新的資訊
63                  break;//結束此程式，跳至第六十六行。
64              default: printf("Constructing...\n");
65          } // 結束switch
66          printf("\nSelect action:\n\tDeposit >> 1\t\tWithdraw >> 2\t\tCheck >> 3\t\tExit >> 0\n > ");
67          scanf("%d", &action);
68      } // 結束while迴圈
69
70      printf("Exit...END\n"); // 離開訊息
71
72      return 0;
73  } // 結束主程式
```

由於圖片過大，這邊以註解方式進行講解。

# 上課內容



```
Select action:
        Deposit >> 1            Withdraw >> 2           Check >> 3              Exit >> 0
 > 1
Enter the amount of deposit > 1000
Account(1234-5678, Catherine, 101000)

Select action:
        Deposit >> 1            Withdraw >> 2           Check >> 3              Exit >> 0
 > 2
Enter the amount of withdraw > 3000
Account(1234-5678, Catherine, 98000)

Select action:
        Deposit >> 1            Withdraw >> 2           Check >> 3              Exit >> 0
 > 3
Account(1234-5678, Catherine, 98000)

Select action:
        Deposit >> 1            Withdraw >> 2           Check >> 3              Exit >> 0
 > 0
Exit...END
```

此為輸入其動作後會輸出的結果，輸入 1就會存款，這邊我存款的數字為 1000，就能看見後續輸出中，帳號的存款從原本的 100000變成了101000。
輸入2後會進行提款的動作，這邊我提款了 3000，能看見後續的輸出中，帳號的存款從101000變為了98000。
輸入3後能顯示帳號最新資訊，可以發現到這和原本前面的存款數字後取款的數字，都能對上。
輸入0後則結束了動作。

# 上課內容



右圖程式為建立一個矩陣後，輸入其值，後續的程式會找出每列最大值並且將其加總。並且會找出每一行的最大值。
第一～三行用來定義函式庫，與使用者需要的行(N)列(M)。

接著第六行來到了主程式。
第七、八行用來輸入行列。
第九行輸入矩陣內的數值。
第十、十一行定義了變數sum總和、temp暫存的臨時空間，用來儲存美行的最大數值。
第十二行用一個數列來存取每行的最大數值。

```
1  #include<stdio.h> /* 定義printf()和scanf() */
2  #define ROW_N 20 /* 列數(row)為20 */
3  #define COLUMN_M 20 /* 行數(column)為20 */
4
5  /* 主程式 */
6  int main(){
7      int N; /* 輸入 - 列數(row) */
8      int M; /* 輸入 - 行數(column) */
9      int array[ROW_N][COLUMN_M]; /* 輸入 - array[][]數值 */
10     int sum = 0; /* 輸出 - 每一列的最大值之和 */
11     int temp = 0; /* 臨時的空間 - 找出每一列的最大值 */
12     int array2[ROW_N]; /* 臨時的空間 - 存放每一列的最大值*/
13
14     /* 輸入array[][]的大小 */
15     printf("please enter the number of ROW and COLUMN:");
16     scanf("%d %d", &N, &M);
17
```

第十五、十六行用來輸入行列數後儲存。

# 上課內容



接續第十八行。
第十九~二十三行利用迴圈性質，來讓輸入的N、M與迴圈的i、j數值相等，變成a[i][j]，並且經過第二十一行輸入後儲存內容。
第二十五用來跳行。
第二十七~四十三行用來判斷矩陣中每行數列的最大數值。
第二十八行利用迴圈性質，將矩陣變為array[i][0]，並且用temp暫存其內容。

第三十二行~三十六行用來判斷矩陣中的每行數列的最大數值。
第三十三、三十四行為布林值，若是array[i][j]的數值大於temp，temp就會等於其數值。
第三十九行定義了一個矩陣array2[i](單行數列來儲存temp的值。)
第四十二行將每行暫存的temp(最大數值)相加。

```
18      /* 輸入array[][]的數值 */
19      for(int i = 0; i < N; i++){
20          for(int j = 0; j < M; j++){
21              scanf("%d", &array[i][j]);
22          } /* 結束內層for迴圈 */
23      } /* 結束外層for迴圈 */
24
25      printf("\n"); // 跳行的符號
26
27      /* 第一行輸出的運算 */
28      for(int i = 0; i < N; i++){
29          /* 先定每一列(row)的第一個數值為最大值 */
30          temp = array[i][0];
31
32          for(int j = 1; j < M; j++){
33              if(array[i][j] > temp){
34                  temp = array[i][j]; /* 取代每一列的最大值 */
35              } /* 結束if判斷 */
36          } /* 結束內層for迴圈 */
37
38          /* 將每一列的最大值存放到array2[] */
39          array2[i] = temp;
40
41          /* 將每一列的最大值加總起來，sum = sum + temp */
42          sum += temp;
43      } /* 結束外層for迴圈 */
44
```

# 上課內容



接續第四十五行。
第四十六行為輸出每行最大值的temp總和。
第四十八行將temp定義為-1，是用來判斷後續的程式。

第五十一~六十三行為一個for迴圈。用來判斷已被儲存於array2的數列中是否能整除sum的值。

五十一行利用迴圈性質讓i=1，i=2，i=3…，i=N，使得數列中的每個數都能被運算。

```
45      /* print出第一行結果 */
46      printf("\n%d\n", sum);
47
48      temp = -1;
49
50      /* print出array2[]的數值 */
51      for(int i = 0; i < N; i++){
52          //printf("%d ", array2[i]);
53
54          /* 判斷sum是否可以被array2[]裡面的數值(每一列的最大值)整除 */
55          if((sum % array2[i] == 0) && (i < N - 1)){ /* 可以被整除，而且不是最後一個數值
56              printf("%d ", array2[i]); /* 尾巴有空格 */
57              temp += array2[i];
58          }
59          else if (sum % array2[i] == 0){ /* 可以被整除，而且是最後一個數值 */
60              printf("%d", array2[i]); /* 尾巴沒有空格 */
61              temp += array2[i];
62          }
63      } /* 結束for迴圈 */
```

第五十四行為判斷sum是否可以被array2[]裡面的數值整除。
第五十五~五十八行表示，若是array2中的數值若能整除sum，並且不是最後一個數值(代表後面有空格)則會經過第五十六行輸出並儲存其數字儲存在 temp(代表此時temp不是-1，有數能整除)。
第五十九~六十一行表示，若是array2中的數值若能整除sum，且是最後一個數值(代表後面沒有空格)則會經過第六十行輸出並儲存其數字儲存在 temp(代表此時temp不是-1，有數能整除)。

# 上課內容



接續至第六十五行。
第六十六行為布林值，若是temp=-1，代表沒有任何一個數字能整除sum。

```
65      /* 判斷sum是否無法被array2[]裡面的數值(每一列的最大值)整除 */
66      if(temp == -1){
67          printf("%d", temp);
68      } /* 結束if判斷 */
69
70      return 0;
71  } /* 結束主程式 */
```

```
please enter the number of ROW and COLUMN:3 4
1 2 3 4
5 6 7 8
9 10 11 12


24
4 8 12
```

以左圖來解釋，此為輸入一個 3*4的矩陣後，所運行輸出後的內容。
可知每行的最大值分別是4、8、12(此時他們為array2的內容。)
相加起來4+8+12=24，輸出24。
並且將array2中的內容也就是4、8、12用來判斷，可知三個數皆可整除sum(24)，於是輸出其三個數值。

```
please enter the number of ROW and COLUMN:3 4
1 3 5 7
2 4 6 8
-9 -10 -11 -12


6
-1
```

以左圖來解釋，此為輸入一個 3*4的矩陣後，所運行輸出後的內容。
可知每行的最大值分別是7、8、-9(此時他們為array2的內容。)
相加起來7+8+(-9)=6，輸出6。
並且將array2中的內容7、8、-9來判斷，可知三個數皆無法整除sum(6)，於是此時temp數值仍就是-1，故輸出-1。

# 上課心得



今天的課程主要是將過去學習到的一些用法統整，並且去模擬一些系統。

今天的課程最讓我印象深刻的是存款、取款的內容，原本的雛型就已經有好幾行程式了，雖然看著複雜，但我還是將其分成許多部份，這樣才能更好的理解每個部份所運行的功能。但沒想到後續還將原本的雛型，新增了更多的功能，雖然一開始很難去理解消耗，但我在課後將原本的雛型釐清明白後，再去了解後續新增的功能，就讓我能更快速地去明白最後的全部程式的每個部份當中，其扮演的角色和功能。
再後續的矩陣程式中，我也經過了許多次的輸入內容才判斷的不同程式間所配合的東西，途中我能明白自己能更有效的去判斷這些程式了，能感知道自己比以前進步很多。

除此之外，我在這堂課也學到了一件事情，就是勇敢去做，我在打本次課堂的簡報時，時常會因為程式過於冗長，所以不知道該從何開始下手，但我最後還是硬著頭皮去解釋每行程式，結果漸入佳境，也是在解釋每行程式的過程中才能明白每個程式的功能，還有後續程式的總和，總結來說，這次不僅僅複習了過去的學到的程式並統合整另一種系統，還學習到了在打程式中，就應該要勇敢地先去式，在慢慢地去修改，至少能先踏出第一步。

# 上課內容

4/22(1)

右圖為一個等差數列的總和，其
首項=1，末項=n，公差=1。
若以n=10舉例。
第三行先定義末項n，n=10。
第四行定義變數sum，用來總和數列。

第六行用來輸入並儲存n的值，也就是10。
第八行為布林值，若n>0(10>0)，則會進行第九~十二行的迴圈，而n符合其條件，故會進行迴圈，迴圈i從1開始累加1，並將每次循環的i值相加起來(1+2+3…+n)，若i>n+1(i>10+1)，則停止迴圈，至第十四行輸出sum值後結束程式。

因此若輸入10的話，i值就會開始累加，
i=1,i=2,i=3…i=10
sum會將i值全部加起來
(1+2+3…+10)=55。

```c
#include<stdio.h>
int main(){
    int n; // 輸入 - 10
    int sum = 0; // 輸出 - 加總1 + 2 + ... + 10 = 55

    scanf("%d", &n); // 輸入一個數值n

    if(n > 0){
        for(int i = 1; i < n + 1; i++){
            //printf("%d ", i);
            sum = sum + i;
        } // 結束for

        printf("%d", sum);
    } // 結束if

    return 0;
}
```

```
10
55
```

# 上課內容

4/22(2)

右圖的程式和上一個程式一樣的等差數列，均為首項=1，末項=n，公差=1。

與上一個程式不同於第九~十一行。
上一個程式是利用布林值if(n>0)為條件才開始進行迴圈。
而右圖程式利用while(n>0)來進行判斷，因為兩者判斷條件相同，且不進行迴圈條件均為n<0，故運行的內容和輸出的結果都相同。

```c
#include<stdio.h>
int main(){
    int n; // 輸入 - 10
    int sum = 0; // // 輸出 - 加總1 + 2 + ... + 10 = 45

    scanf("%d", &n); // 輸入一個數值n

    // 第二種完整的寫法，會重複進行判斷
    while(n < 1){
        scanf("%d", &n);
    } // 結束while

    for(int i = 1; i <= n; i++){
        sum += i; // sum = sum + i
    } // 結束for

    printf("%d", sum);

    return 0;
}
```

# 上課內容



第一~十四行為後續程式所要執行的任務。
主旨就是先輸入數列中所要的長度 n,
而後輸入數列內容，再進行排序，而後可以輸入想尋找的元素，就會輸出其位置。

接著第十六行定義了SIZE為100(用來代表數列的長度。)

第十八行定義了bubbleSort的涵式。
第十九行行定義了swap其交換的涵式。

第二十一行定義selectionSort的涵式。
第二十二行定義了最大數值的涵式。

```
/*陣列、排序、尋找
*任務1：輸入一個數字n(最大為100)，輸入n個數值(任何整數)，輸出n個排序好的數值
*任務2：輸入一個數字k，輸出k的位置，否則輸出-1
*********************************
*任務1輸入範例：
*第一行：5(數列有5個數值)
*第二行：-1 99 34 0 4
*
*任務1輸出範例：
*第一行：-1 0 4 34 99(將五個數值排列)
*
*任務2輸入範例1：0        輸出範例1：1(尋找0的位置，其位置在1，輸出1)
*任務2輸入範例2：9        輸出範例2：-1(尋找2的位置，找不到，輸出-1)
*/
#include<stdio.h>
#define SIZE 100 //number[]最大為100個連續格子

void bubbleSort(int * const array, const int size);
void swap(int *element1Ptr, int *element2Ptr);

void selectionSort(int * const array, const int size);
void place_largest(int * const array, const int size);

```

# 上課內容



接續第二十四行，來至了主程式。
第二十五行定義了變數 n
第二十六、二十七行定義了兩個數列 number、number2，其大小為SIZE。
第二十九行輸出括號內容。
第三十行行輸入了n值並儲存。

第三十二行進入到了迴圈，利用迴圈性質將 i+1值至 i=n，第三十四行就會使number、number2兩個數列的長度都等於n。

第三十七行呼叫了bubblerSort排序number。
第三十八行呼叫了selectionSort排序number2。

第四十~四十二行、第四十五~四十七行利用迴圈性質讓排完的數列輸出。



```
24  int main(){
25      int n; //輸入 - 定義number[]的大小
26      int number[SIZE]; //輸入 - 任何的整數 0,1,2,3,4,5,6,7
27      int number2[SIZE];
28
29      printf("please enter the size of number:",&n);
30      scanf("%d", &n); //定義number[]大小
31
32  for(int i = 0; i < n; i++){
33      scanf("%d", &number[i]); //存放number[]的數值
34      number2[i] = number[i];
35  } //結束for
36
37  bubbleSort(number, n); //呼叫bubbleSort()氣泡排序法
38  selectionSort(number2, n); //呼叫selectionSort()選擇排序法
39
40  for(int i = 0; i < n; i++){
41      printf("%d ", number[i]); //print出排序好的number[]
42  } //結束for
43
44  putchar('\n'); //跳行
45  for(int i = 0; i < n; i++){
46      printf("%d ", number2[i]); //print出排序好的number[]
47  } //結束for
48
49
```

# 上課內容



接續至第五十行，定義了變數 k，用來查詢所要的數字的位置。第五十一行定義了變數 index=-1，表示沒找到其數字。

第五十三行用來跳行。
第五十四行輸出括號內容。
第五十五行用來輸入並儲存 k值。

第五十七行利用迴圈性質，使得數列中的數字能跑過一遍，而當找到了某個數=k時(利用布林值if)，就會經過第五十七行輸出位置i，且改變index的數值，讓其不影響後續程式。

第六十四行為布林值，若index=-1(代表在數列中沒到k值，因此index未做更改)就會經過第六十三行輸出 -1。

第六十八行用來換行。
第七十行代表結束此部分的主程式。



```
50  int k; //輸入 - 尋找在number[]的數值
51  int index = -1; //假設沒有找到對應的數值
52
53  putchar('\n'); //跳行
54  printf("please enter the number you want:");
55  scanf("%d", &k); //讀取要尋找number[]的數值
56
57  for(int i = 0; i < n; i++){
58      if(k == number[i]){
59          printf("in the %d\n",i);
60          index = 2; //更正index
61      } //結束if
62  } //結束for
63
64  if(index = -1){ //確定沒有找到對應的數值
65      printf("%d", -1);
66  } //結束if
67
68  putchar('\n'); //跳行
69
70  return 0;
71  }
72
```

# 上課內容



```
73  void bubbleSort(int * const array, const int size){
74      for(int i = 0; i < size; i++){
75          for(int j = 0; j < size - 1; j++){
76              if(array[j] > array[j+1]){//判斷大小
77                  swap(&array[j], &array[j + 1]);//交換數值
78              } //結束if
79          } //結束for
80      } //結束for
81  } //結束bubbleSort()
82  //定義swap的副程式
83  void swap(int *element1Ptr, int *element2Ptr){
84      int hold = *element1Ptr;//定義hold暫存數值
85      *element1Ptr = *element2Ptr;//交換
86      *element2Ptr = hold;//將暫存的數換至被換掉的數字的位置
87  } //結束swap()
```

```
88
89  void selectionSort(int * const array, const int size){
90      if(size > 1){
91          place_largest(array, size); //呼叫place_largest()
92          selectionSort(array, size - 1); //呼叫selectionSort()
93      } //結束if
94  } //結束selectionSort()
95
96  void place_largest(int * const array, const int size){
97      int max_index = size - 1; //暫定最後一個位置為最大值
98      for(int i = size - 2; i >= 0; --i){
99          if(array[i] > array[max_index]){
100             max_index = i; //紀錄目前最大值的位置
101         } //結束if
102     } //結束for
103     if(max_index != size - 1){
104         swap(&array[max_index], &array[size - 1]);
105     } //結束if
106 } //結束place_largest()
```

上面的兩個程式用來定義兩個排序法，由於前面已解釋過，因此這邊用註解大概解釋。

數列排序位置是由 0開始計

```
please enter the size of number:5
-1 99 34 0 4
-1 0 4 34 99
-1 0 4 34 99
please enter the number you want:34
in the 3
```

```
please enter the size of number:5
-1 99 34 0 4
-1 0 4 34 99
-1 0 4 34 99
please enter the number you want:20
-1
```

# 上課心得



本次課堂老師主要複習了泡沫排序法和選擇排序法的作用，並且也在一個程式中使用到了兩個排序法，經過這次的複習我已經能明白每行程式之間的作用了。

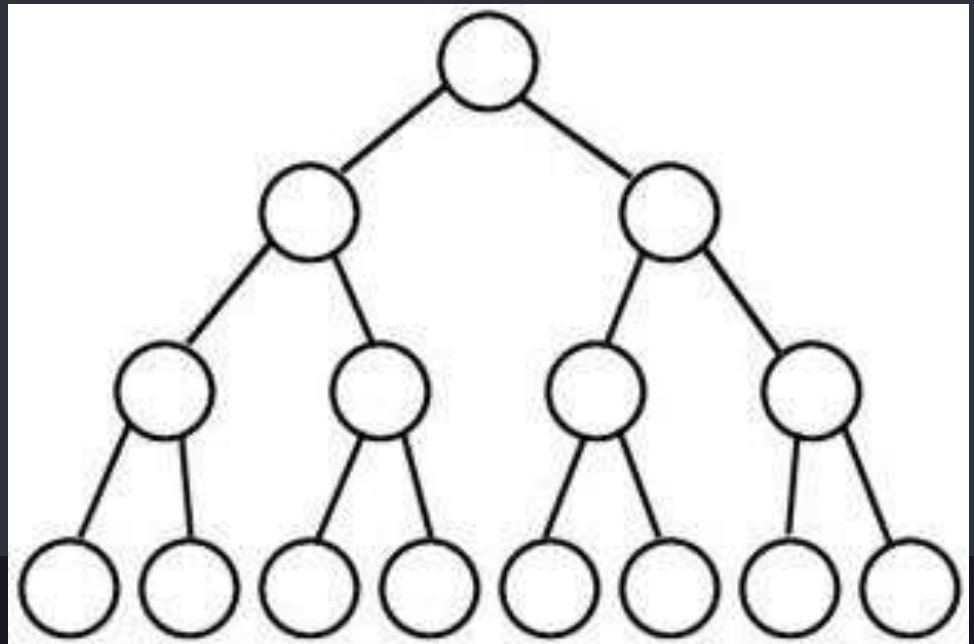
今天的課程中所教的第三個的程式，是第一個打超過破百行的程式，雖然大部分的程式都是註解，但看到自己能打超過破百行的程式，心裡有一種說不出的成就感，但也由於行數太多，我在解釋整個程式時也花了不少苦，尤其是我發現到老師的程式有些錯誤，於是我在之後進行測試的時候花了不少的時間，原本以為輸出的內容是對的，是我看不懂程式，但我在每行的分析之下才發現到是有寫部分打錯了，也因此我花了不少的時間在修正程式和不停地測試，當成功測試出結果後，原本冗長的程式在此刻似乎不再那麼的陌生，我反而很有興致的重複看了好幾遍自己修改過後的程式，所以我在後續解釋程式時，一點也不覺得吃力，解釋中也沒有什麼困難。我認為，或許這就是一種成長，一種只有自己知道的成長的感覺。
總結來說，這堂課又收穫了不少的東西，不僅僅是對程式的了解，更是對心靈上的成長。

# 上課內容



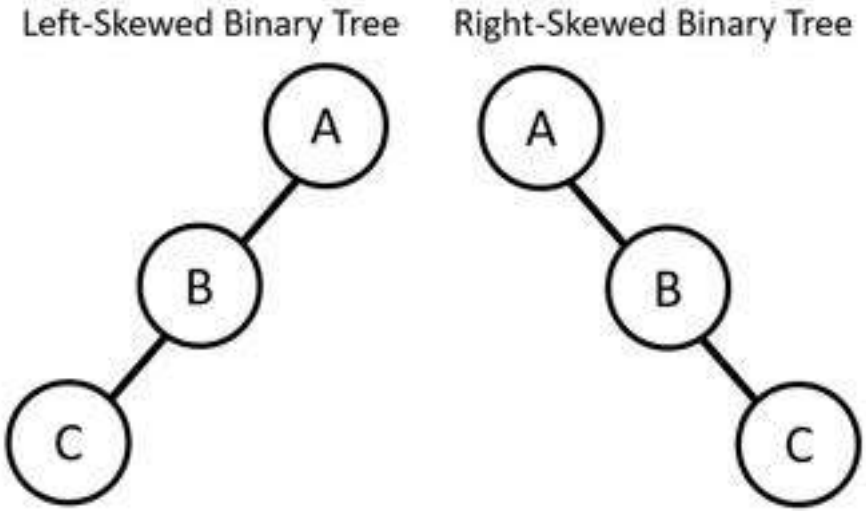
二元樹:一種常用的資料結構，可以去用來儲存、處理資料。頂端會有一個點，每個節點之下又會有兩個節點(此為通常情況下，二元樹有著許多不同種類)，稱為左子樹、右子樹，且左節點的數字(資料)通常較小，而右節點的數字(資料)通常較大。
雖然二元樹有許多同種類，但他們本質都是利用頂端的點而後一個個去推算資料，因此在運行上基本都一樣。

後續將會提到一些有關二元樹的用法。
右圖為其基本架構。

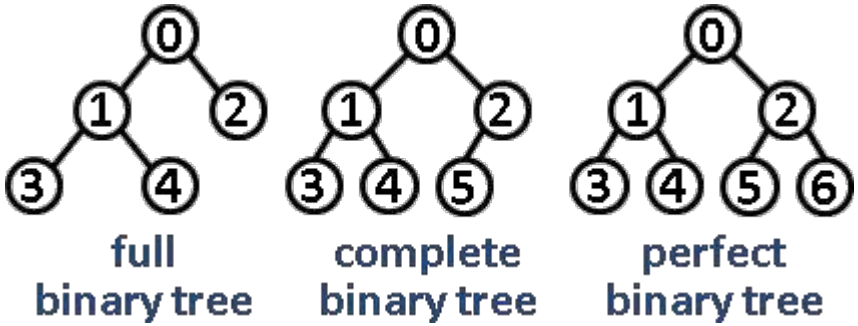


# 上課內容



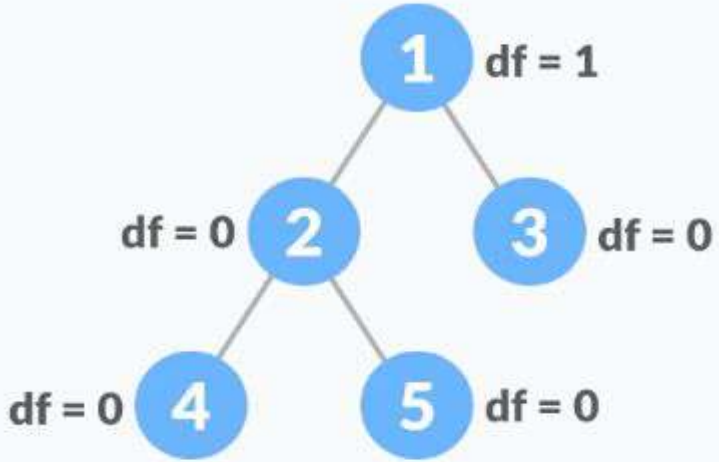



左斜曲二元樹、右斜曲二元樹




滿二元樹full binary tree:代表每個節點都連結另外兩個節點，或者都不連，不會只連一個。
完全二元樹complete binary tree:表示單獨一個的節點會靠左。
完美二元樹perfect binary tree:代表每個節點都連結著另外兩個節點。


退化二元樹，是指左側或者是右側指有一個節點，與斜曲二元樹不同的是，它可以是先左節點再右節點。



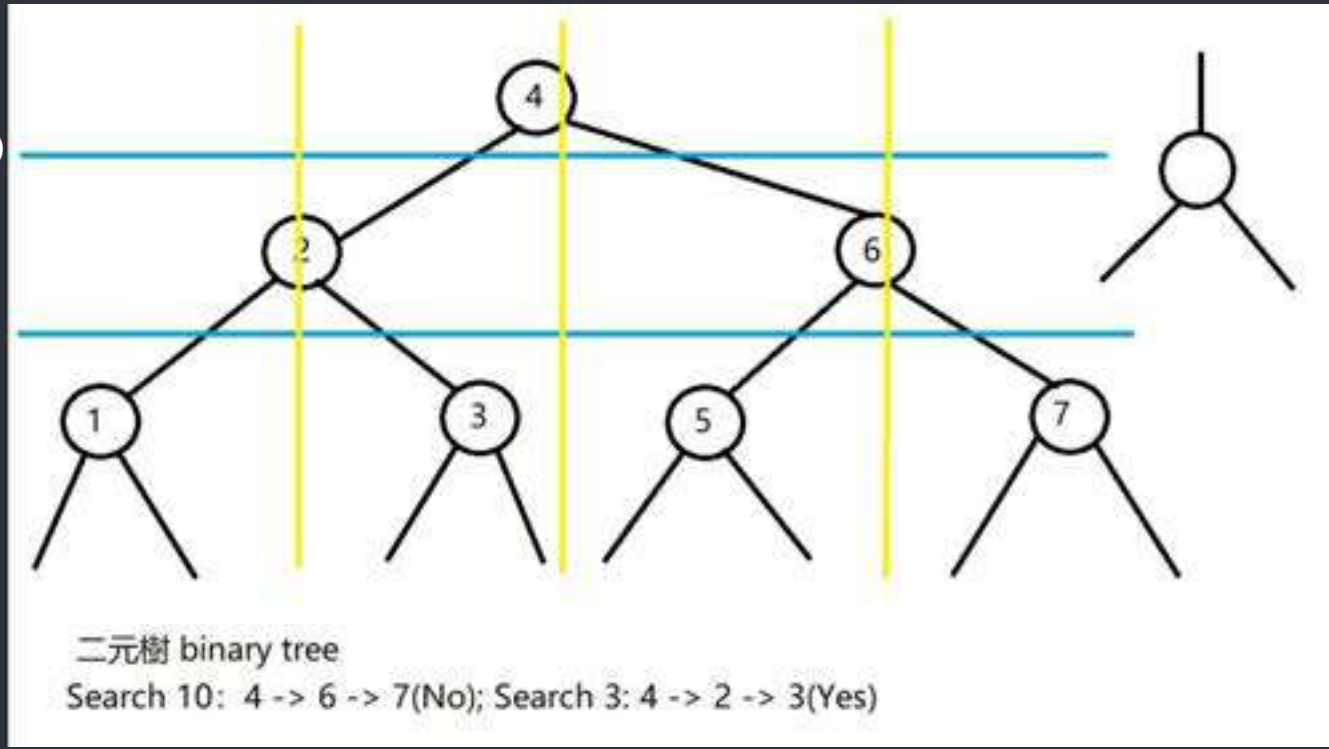

平衡二元樹，代表高度都只有 0或1。
高度代表是，層df。
舉例來說，對於2、3、4、5來說，它們是完整的二元樹，所以df=0。
只不過對1來說，缺少了層(3的子節點)，因此df=1

# 上課內容



右圖為二元樹。
若要找尋10的話，會先從4(頂端)開始尋找，然後找尋至6，再找尋至7，會發現到此二元樹最大結果只有7，因此找不到10，就會找不到而回傳NO。

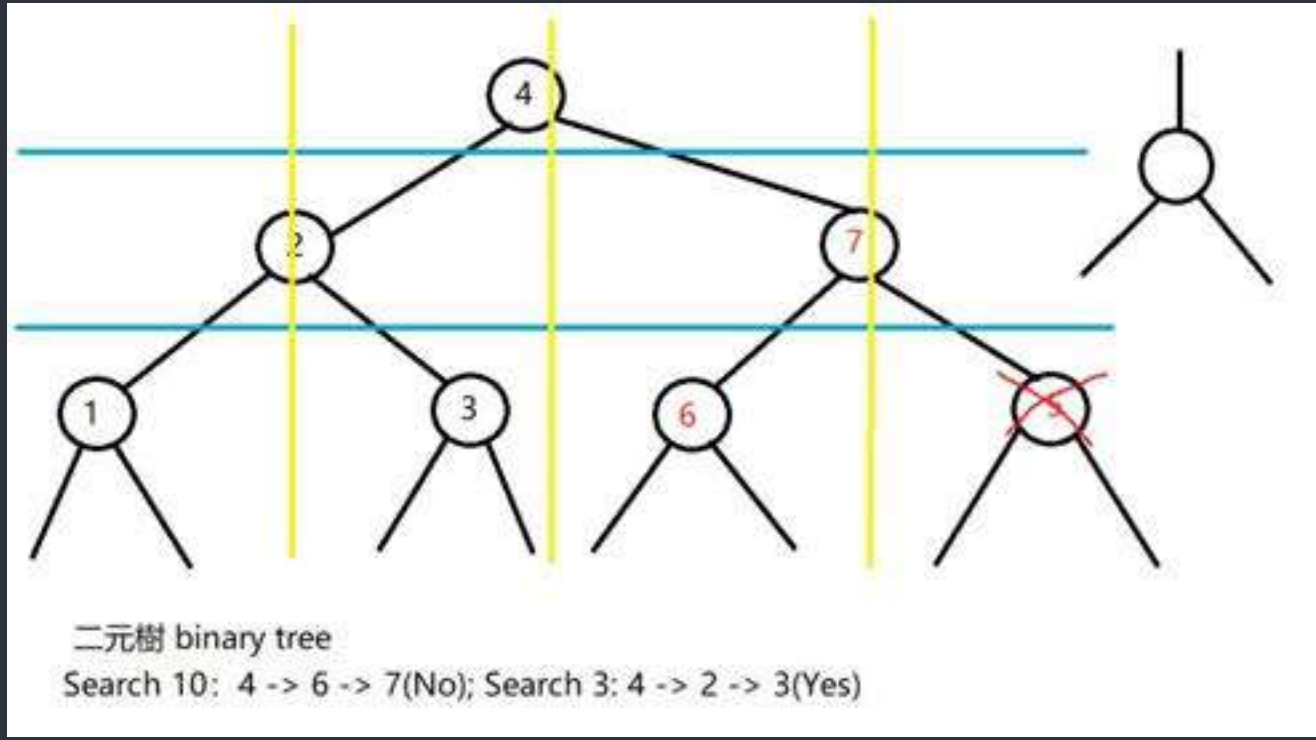
而若是要尋找3，一樣會先從4(頂端)開始尋找，而後來到2，最後就會發現到3的節點，因此找到的就會輸出Yes



# 上課內容



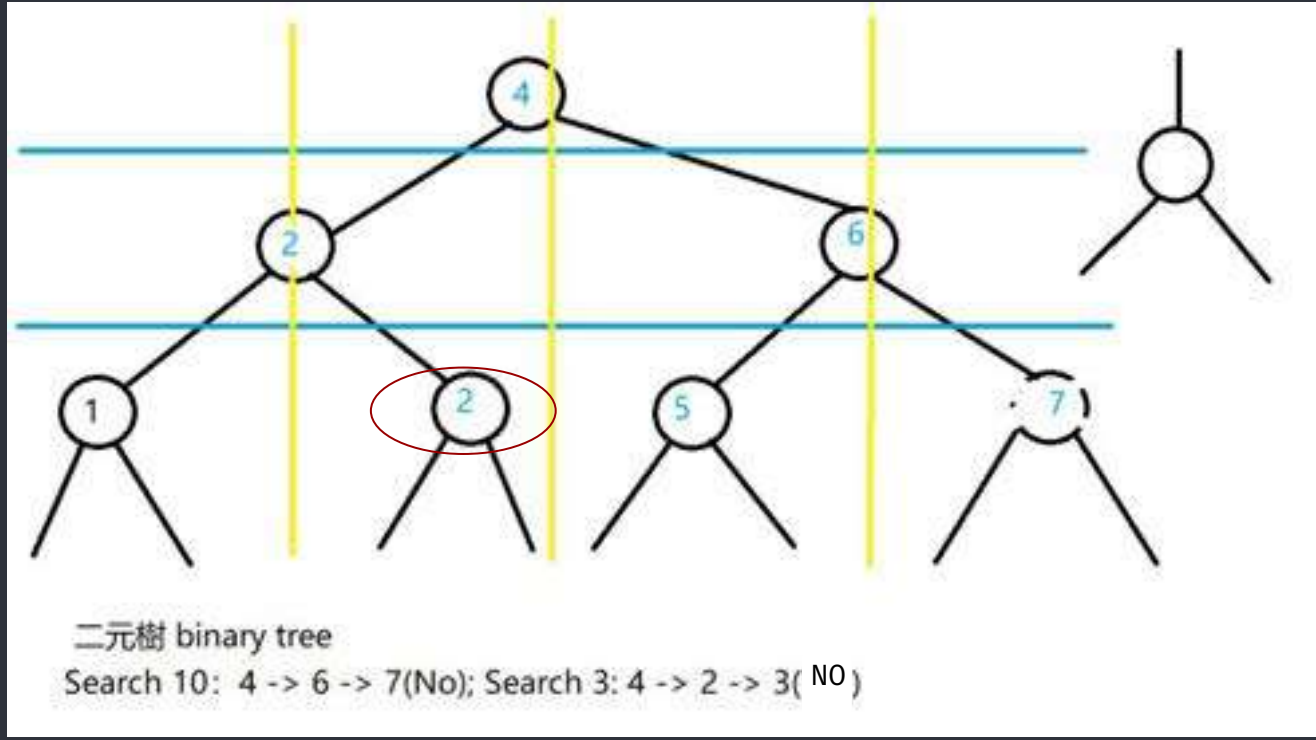
右圖為二元樹的用法。
其中多了移除資料的過程。
若要找尋10的話，會先從4(頂端)開始尋找，然後找尋至6，再找尋至7，會發現到此二元樹最大結果只有7，因此找不到10，會找不到而回傳NO。，只不過因為節點5刪掉不影響其出。
輸出
而若是要尋找3，一樣會先從4(頂端)開始尋找，而後來到2，最後就會發現到3的節點，因此找到的就會輸出Yes(只不過因為節點5刪掉不影響其出。)



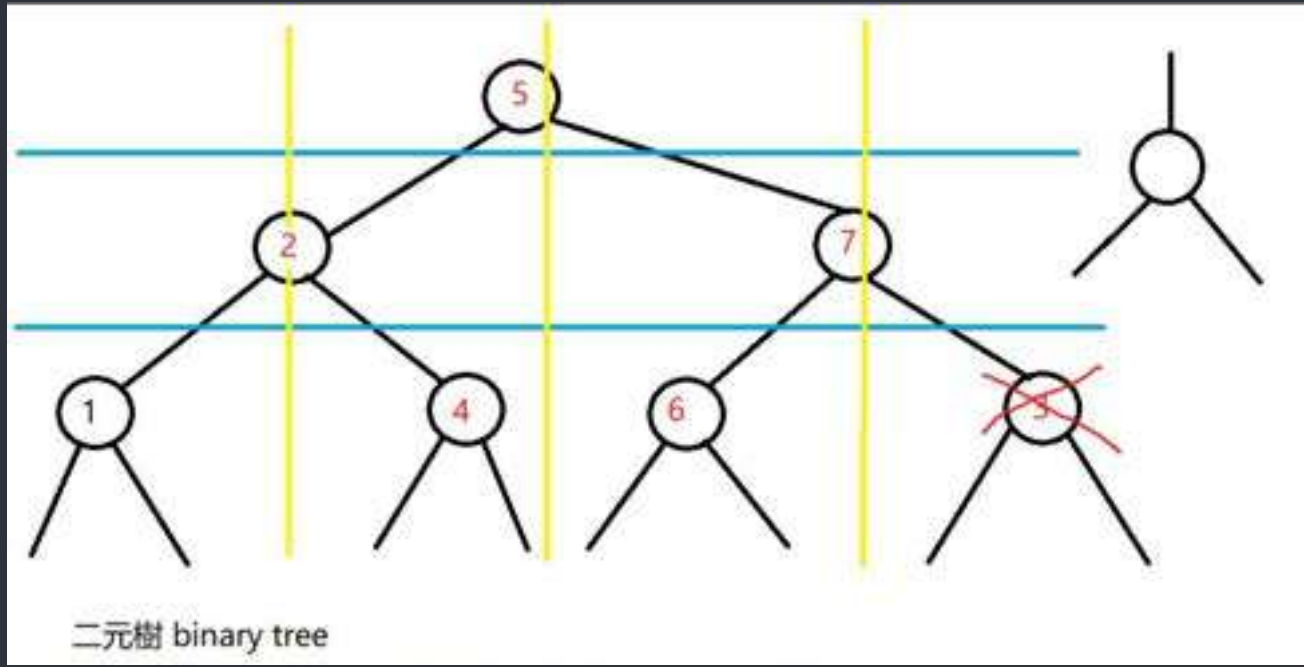
# 上課內容



紅圈的地方原本是3，
但為了要試試看節點之間的關係，引此先暫訂3為2。
若要搜尋10，會先從4(頂端)開始尋找，然後找尋至6，再找尋至7，會發現到此二元樹最大結果只有7，因此找不到10，會找不到而回傳NO。
而若是要尋找3，一樣會先從4(頂端)開始尋找，而後來到2，最後就會發現到找不到3的節點，因此找到的就會輸出NO。



若是將節點的值改變，則其運行規則基本相同(除非改變值後違反規則。)
若是要搜尋10，會先從5(頂端)開始尋找，然後找尋至7，再找尋至6，會發現到此二元樹最大結果只有6，因此找不到10，會找不到而回傳NO。
而若是要尋找3，一樣會先從5(頂端)開始尋找，而後來到7，最後就會發現到找不到3的節點，因此找到的就會輸出NO。




二元樹 binary tree

# 上課內容

4/29(1)

右圖程式為二元樹的運行方式講解。
第一、二行導入了函式庫。
第四~七行定義了一個架構tree，裡面定義了整數size、節點node指標root。並別名為TREE。

第九~十四行定義了架構node，裡面定義了整數date，節點指標left(左節點)、right(右節點)、parent(父節點)。並別名為NODE。

第十六~二十一行定義了NODE指標中的allocNode，其參數為整數x，表示要儲存的資料。
第十七行將NODE型別的指針temp(用來暫存位置。)
第十八行表示使用了malloc函式，並分配了大小為NODE且為NODE型別的指針，如果賦值temp失敗了，則輸出第十九行內容，再經第二十行回傳NULL。

第二十三行初始化了temp中的左、右、父節點為NULL，並且將temp內的data定為參數x，再回傳第二十五行的temp。

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

typedef struct tree{ //定義樹的結構：整棵樹的大小、和節點
    int size;
    struct node *root;
} TREE;

typedef struct node{ //定義節點的結構：節點的data、左邊、右邊、parent
    int data;
    struct node * left;
    struct node * right;
    struct node * parent;
} NODE;

NODE * allocNode(int x){
    NODE *temp;
    if((temp = (NODE*)malloc(sizeof(NODE))) == NULL){
        printf("No memory\n");
        return NULL;
    }

    (temp -> left) = (temp -> right) = (temp -> parent) = NULL;
    temp -> data = x;
    return temp;
}
```

# 上課內容



接續第二十七行。
第二十八行定義了一個程式 treeInit，其參數為t。
第二十九行將t指向了二元樹的size並賦值為0。
第三十行將t指向了root，並賦值為NULL。

第三十四行定義了一個程式 inorderTreeWalk，其參數為root。
第三十五~三十九行為布林值，如果root不等於NULL，則會將inorderTreeWalk的參數值指向left，後輸出root中的數據，再將其指向 right。

第四十三行定義了一個 NODE中的函數treeSearch，其參數為NODE中的root和整數x。
第四十四行表示，如果 root=NULL且x等於root中的data，則會回傳root。
第四十七行表示，如果 x小於root中的data，則會回傳treeSearch函式的參數， root的左節點和x值。
第五十行表示回傳 treeSearch函式的參數root的右節點和x值。



```
27
28  void treeInit(TREE * t){ //樹的初始化
29      (t -> size) = 0;
30      (t -> root) = NULL;
31  }
32
33  //走遍整棵樹的路徑：先找左腳，return本身，再找右腳
34  void inorderTreeWalk(NODE * root){
35      if(root != NULL){
36          inorderTreeWalk(root -> left);
37          printf("%d\n", root -> data);
38          inorderTreeWalk(root -> right);
39      }
40  }
41
42   //寫法1，搜尋方法都一樣：先比較，小的就往左腳，大的就往右腳
43  NODE * treeSearch(NODE * root, int x){
44      if((root == NULL) || (x == root -> data)){
45          return root;
46      }
47      if(x < (root -> data)){
48          return treeSearch(root -> left, x);
49      }
50      return treeSearch(root -> right, x);
51  }
```

# 上課內容



接續第五十二行。
第五十三~六十二行的主要作用和第四十三 ~五十一行一樣，但為不同的表示方法。
第五十三行定義了函式 iterativeTreeSearch，其參數為root、x，且賦於了NODE類型指標。

第五十四行為while迴圈，當root不等於NULL並且X不等於data，則進入至第五十五 ~六十一行
第五十五行表示，如果 x小於data，則進行第五十六行，將root賦值為left。
第五十八行，如果沒有的話，則會將 root賦值為right。其作用是用來比較、交換大小位置的。
第六十三行回傳root。
第六十六行定義了一個函式 treeMinimum，其參數為root。
第六十七行，當left不等於0時(表示有東西)，則會進入第五十八行，將 root賦值left。
第七十一行回傳root。

```
52  //寫法2，搜尋方法都一樣：先比較，小的就在左邊，大的就在右邊
53  NODE * iterativeTreeSearch(NODE * root, int x){
54      while((root != NULL) && (x != root -> data)){
55          if(x < (root -> data)){
56              root = root -> left;
57          }
58          else{
59              root = root -> right;
60          }
61      }
62
63      return root;
64  }
65  //尋找整棵樹的最小值，即是整棵樹的最左邊，提示：左邊的數值比較小
66  NODE * treeMinimum(NODE * root){
67      while((root -> left) != NULL){ //當左邊還有東西
68          root = root -> left;
69      }
70
71      return root;
72  }
73
```

# 上課內容



接續第七十四行，定義了函式 treeSuccessor，其參數為root。第七十五將y賦予了NODE類型的指標。

第七十八~八十四行為布林值
第七十二行，若是root等於NULL，則回傳NULL。
第八十二行，若是right不等於NULL(還有東西)，則回傳第八十三行。

第八十六行，將y賦值parent。

第八十九行，當y不等於NULL且root等於right，經過第九十、九十一行將root賦值y，再將y賦值給parent。

第九十四~九十六行，如果y等於NULL，則回傳root。

第九十七行，回傳root。



```
74  NODE * treeSuccessor(NODE * root){ //尋找接班人
75      NODE * y;
76
77  //如果root是NULL，則沒有接班人，返回NULL
78      if(root == NULL){
79          return NULL;
80      }
81
82      if((root -> right) != NULL){ //有右子樹，找右邊最小的
83          return treeMinimum(root -> right);
84      }
85  //沒有右子樹，只好往上找，y設定為root的父親
86      y = (root -> parent);
87
88  //若在父親的左邊，則父親就是下一個
89      while((y != NULL) && (root == (y -> right))){
90          root = y;
91          y = root -> parent;
92      }
93
94      if(y == NULL){
95          return root;
96      }
97      return y;
98  }
99
```

# 上課內容



接續第一百行，定義函式treeDelete，其參數為t，z。
第107行，如果z指向的left等於NULL且z指向的right也等於NULL，則賦值y等於z。
第110~112行，若否，則y賦值treeDelete的參數z。

第114行，如果y指向的left不等於NULL，則賦值x等於y指向的left。
第117~119行，若否，則賦值x為y指向的right。

第121~123行，若x不等於NULL，則x所指向的parent賦值y所指向的parent。

```
100  NODE * treeDelete(TREE * t, NODE * z){ //刪除z節點，傳回實際刪除的節點
101      NODE * y, * x, * p, * q; //p是z的父節點，q是z的左右指針
102
103      //如果z只有0或1個子樹，則可把z的子樹往上接，
104      //不然只好找z的下一個y，在z右側子樹的情況下y保證沒有左子樹，再把z的內容換成y的
105      //因此變數y定義為實際上要刪除的節點，變數x為y的子節點
106
107      if((z -> left) == NULL) || ((z -> right) == NULL)){
108          y = z;
109      }
110      else{
111          y = treeSuccessor(z);
112      }
113
114      if((y -> left) != NULL){ //找出y下面的子樹(充一開頭的註解，最多1個)
115          x = (y -> left);
116      }
117      else{
118          x = (y -> right);
119      }
120
121      if(x != NULL){ //y有子樹x，把x接上來
122          (x -> parent) = (y -> parent);
123      }
124
```

# 上課內容



接續第125行，p賦值y指向的parent。
第126~128行，如果y等於y所指向的parent中，所指向的left，則q賦值y所指向的parent中，所指向的left。
第129~131行，若否，則q賦值y所指向的parent中，所指向的right。

第133~135行，若y所指向的parnt等於NULL，則t所指向的root賦值為x。
第136~143行，若y等於y所指向的parent中，所指向的left，則將left賦與x的值，若否，則將y所指向的parent中，所指向的right賦於x的值。
第145~154行，若y不等於z，則z所指向的data等於y所指向的data，p等於z所指向的parent。而如果z等於z所指向的parent當中所指向的left，則將q賦予其值。
若否，則q等於z所指向的parent當中所指向的right。

```
125     p = (y -> parent); //記錄y的父節點
126     if(y == (y -> parent -> left)){ //記錄y是父節點的左邊還右
127         q = (y -> parent -> left);
128     }
129     else{
130         q = (y -> parent -> right);
131     }
132
133     if(y -> parent == NULL){ //找到根節點(root，最頂端)
134         (t -> root) = x;
135     }
136     else{ //把y的父節點的子節點設為x
137         if(y == (y -> parent -> left)){
138             (y -> parent -> left) = x;
139         }
140         else{
141             (y -> parent -> right) = x;
142         }
143     }
144
145     if(y != z){ //如果是用successor取代的方式，記得把y的資料複製到z
146         (z -> data) = (y -> data);
147         p = (z -> parent); //更新p和q為z的父節點和左右指針
148         if(z == (z -> parent -> left)){
149             q = (z -> parent -> left);
150         }
151         else{
152             q = (z -> parent -> right);
153         }
154     }
155
```

# 上課內容



第156行，將已被刪除的y節點從記憶體當中釋放。

第157行，t所指向的size-1(更新二元樹大小。)

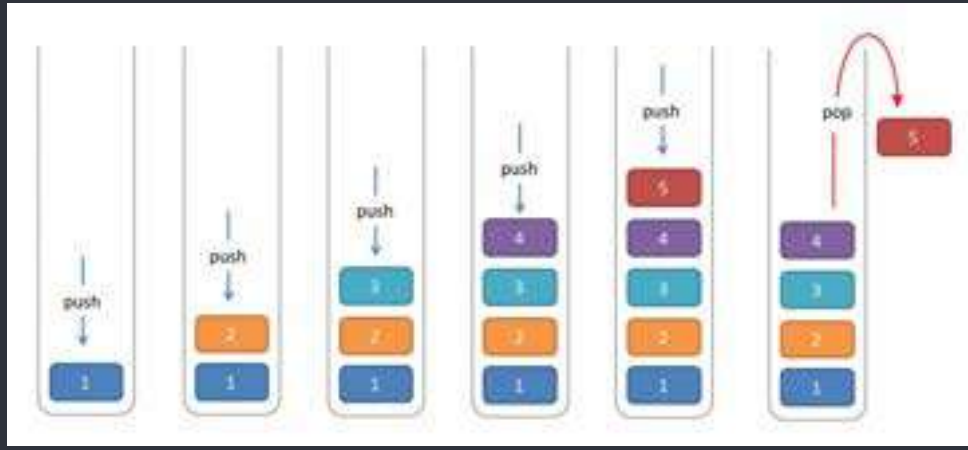
第159行，如果p不等於NULL，則進行第160~166行，而如果q等於p所指向的left，則將p所指向的left賦值為z。
若否，則將p所指向的right賦值為z。

第170、171行，來到了主程式，不過因為此程式主要是用來表示二元樹的運行過程，所以沒有可帶入的數值去執行，故這邊直接回傳 0。

```
156     free(y);
157     t -> size--;
158
159     if(p != NULL){ //如果有父節點，則把p的子節點重新連接為z
160         if(q == (p -> left)){
161             (p -> left) = z;
162         }
163         else{
164             (p -> right) = z;
165         }
166     }
167 }
168
169
170 int main(){
171     return 0;
172 }
```

# 堆疊stack

右圖中的內容為程式語言或者電腦科學當中的一種概念，叫做堆疊stack。
堆疊的概念就是將數張紙慢慢疊起來形成一疊紙，再從那一疊紙中拿取第一張，而此時就可以發現到，最晚放的那一張紙是最早被拿走的。




圖片網址:https://reurl.cc/Yea9AX

因此以資料來說，當有很多筆的資料堆疊再一起，如果使用的是堆疊方法，電腦會先處理最上層的資料，而最上層的資料也是最晚被放進的資料，總結來說電腦會先依序處理最上層的資料直至下層。所以堆疊的特性就是後進先出;先進後出。

# 上課內容



右圖為堆疊stack程式的用法。
第二行先定義了MAXLEN為100，用於後續的數列長度。

第四行定義了字元類型為 Data。
第五~八行定義了stack的結構，內容為top(最頂)、資料長度。

第十行定義了函式 stackInit，其參數為Stack的指標s。
第十一行使得stack(s)中的top為0(清空堆疊的數據。)

第十四行定義了數據Data中的動作，stackpop是指移除一個資料。第十五行表示會回傳 stack中data的top值並回傳被拿掉的數值。

第十八行定義了數據Data中的動作，stackPush是指新增一個資料，參數值為Stack中的數據x。
第十九行表示stack中的數據top被拿掉的值為x。

第二十行回傳x的值。

```
1  #include<stdio.h> //定義pritnf()
2  #define MAXLEN 100 //定義堆疊stack的大小
3
4  typedef char Data;
5  typedef struct _stack{ //定義堆疊stack
6      int top; //記錄當前的位置
7      Data data[MAXLEN];
8  } Stack;
9
10 void stackInit(Stack *s){ //初始化堆疊stack
11     s -> top = 0; //清空堆疊stack裡面的東西
12 }
13
14 Data stackPop(Stack *s){
15     return s -> data[--s -> top]; // s -> top
16 }
17
18 Data stackPush(Stack *s, Data x){
19     s -> data[s -> top++] = x;
20     return x;
21 }
22
```

# 上課內容



接續至第二十三行，來到了主程式。
第二十四行定義了變數i。
第二十五行呼叫了Stack s。
第二十六行定義了stackInit中的資料s，

第二十七、二十八行放入了a、1，兩筆數據。
地二十九、三十行移除了兩筆數據，但根據 stack的性質，先進後出；後進先出的性質，可以知道會先移除 1這筆資料，再移除a。

第三十二~三十五行放入了b、2、c、3這四筆資料。

第三十七~四十行移除了四筆資料，依上述同理，根據stack的性質，先進後出；後進先出的性質，可以知道會先移除b這筆資料，再移除2，再移除c，而後3。

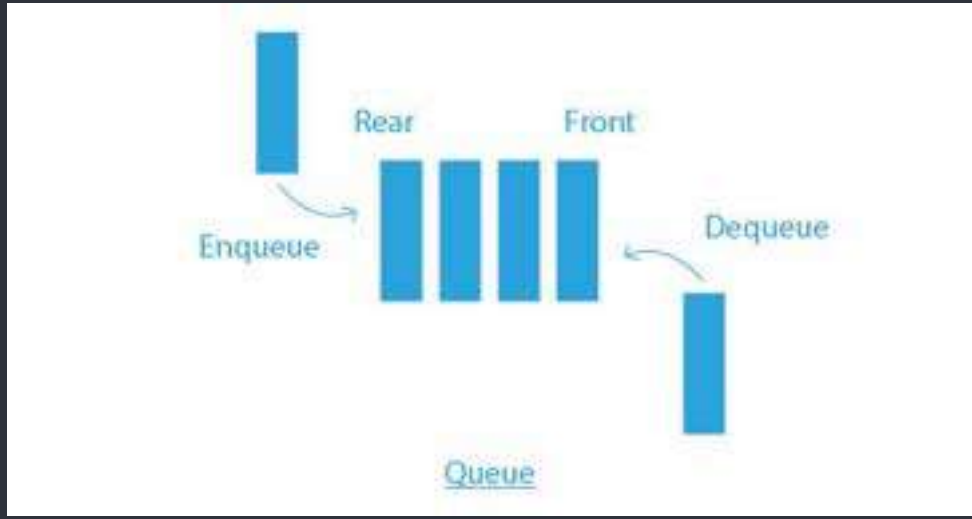
```
push a
push 1
pop 1
pop a
push b
push 2
push c
push 3
pop 3
pop c
pop 2
pop b
```

```c
int main(){ //主程式
    int i;
    Stack s;
    stackInit(&s);
    printf("push %c\n", stackPush(&s, 'a'));
    printf("push %c\n", stackPush(&s, '1'));
    printf("pop %c\n", stackPop(&s));
    printf("pop %c\n", stackPop(&s));

    printf("push %c\n", stackPush(&s, 'b'));
    printf("push %c\n", stackPush(&s, '2'));
    printf("push %c\n", stackPush(&s, 'c'));
    printf("push %c\n", stackPush(&s, '3'));

    printf("pop %c\n", stackPop(&s));
    printf("pop %c\n", stackPop(&s));
    printf("pop %c\n", stackPop(&s));
    printf("pop %c\n", stackPop(&s));
    return 0;
} //結束主程式
```

# 佇列queue




圖片網址:https://reurl.cc/WGmk2e

右圖中的內容同樣為程式語言或者電腦科學當中的一種概念，叫做佇列queue。
佇列的概念，就像是排隊進入餐廳的人一樣，有個先來後到的概念，也就是先進來的人會先進到餐廳，後到的人就比較晚進餐廳。就以資料來說，當今天有數筆的資料被丟進處理器，資料就會根據先來後到的概念進行佇列，電腦就會根據排隊的順序由前至後的處理資料。總結來說，其特性為先進先出；後進後出。

Front:前端，表示佇列中第一筆(最前面)資料。Rear:後端，表示佇列中最後一筆資料。
Dequeue:表示取出佇列中的第一筆資料。Enqueue:表示在佇列中的最後面再加一筆資料。

# 上課內容



右圖的程式為queue佇列的用法。
第二行定義了MAXLEN的值，用來表示佇列queue的大小。

第四行定義了一個結構QUEUE，內容為：
第五行的count，用來表示數列有幾個值。
第六行的head，表示最前面的資料。
第七行為tail，表示最後面的資料。

第十一行定義了一個函式queueInit，內容為QUEUE指標q。
第十二行表示數列中的head、tail、count都為0。
第十五行定義了變數queueIsFull，用來判斷佇列是否已滿，會根據q中的count與MAXLEN比較，若滿回傳1，否則0。

第十九行定義了函式queueAdd，表示要插入的資料。
第二十、二十一行會判斷佇列大小是否超過 MAXLEN。

```c
#include<stdio.h> //定義printf()
#define MAXLEN 100 //定義佇列queue的大小

typedef struct{
    int count;
    int head;
    int tail;
    int data[MAXLEN];
} QUEUE;

void queueInit(QUEUE *q){ //初始化佇列queue
    (q -> head) = (q -> tail) = (q -> count) = 0;
}

int queueIsFull(QUEUE *q){
    return ((q -> count) >= MAXLEN);
}

int queueAdd(QUEUE *q, int x){
    if((q -> count) >= MAXLEN){
        printf("Queue is full\n");
        //exit(0);
        return -1;
    }
```

# 上課內容



接續第三十一行。
三十案行定義了變數queueIsEmpty，QUEUE *q表示其參數，判斷佇列是否為空，是的話回傳 1，否則0。

第三十六行為變數queueRemove，QUEUE *q表示其參數，會將其值暫存在temp。

第三十九~四十三行為布林值，用來判斷函式是否為空，如果q中的count<0，則會輸出第四十行。後回傳-1。

第四十五行使得q中的count減1，表示取走一個資料。
第四十六行暫存被取走的資料，並且存在 q的數列data當中，而q中的head會加1，讓其能運行下一個資料。
第四十八行則是head除以MAXLEN的餘數為0後，就會設head為0(主要是讓其進行下一筆資料。)

第四十八行回傳暫存的值。

```
31
32  int queueIsEmpty(QUEUE *q){
33      return ((q -> count) <= 0);
34  }
35
36  int queueRemove(QUEUE *q){
37      int temp;
38
39      if((q -> count) <= 0){
40          printf("Queue is empty\n");
41          //exit(0);
42          return -1;
43      }
44
45      q -> count--;
46      temp = (q -> data[(q -> head++)]); //拿出head資料
47      (q -> head) %= MAXLEN; //如果head超過MAXLEN，要設為0
48      return temp;
49  }
50
```

# 上課內容



接續第五十一行，來到了主程式。
第五十二行定義了QUEUE類型的變數q，用來儲存或使用佇列中的元素。

第五十三行呼叫了queueInit初始化q。
第五十四、五十五行在佇列中新增了 123，456兩筆資料。
第五十六、五十七行移除了佇列中的資料，由於先輸入的資料是123，故先刪除此資料，再刪除 456。

第五十九~六十一行新增了三筆資料
1234,5566,1314。

第六十二、六十四行刪除了三筆資料
因為1234是最先輸入的，根據佇列性質，先進先出可推知，1234會先被移除，而後是5566，再來是1314。

```
51  int main(){
52      QUEUE q;
53      queueInit(&q);
54      printf("add queue %d\n", queueAdd(&q, 123));
55      printf("add queue %d\n", queueAdd(&q, 456));
56      printf("remove queue %d\n", queueRemove(&q));
57      printf("remove queue %d\n", queueRemove(&q));
58
59      printf("add queue %d\n", queueAdd(&q, 1234));
60      printf("add queue %d\n", queueAdd(&q, 5566));
61      printf("add queue %d\n", queueAdd(&q, 1314));
62      printf("remove queue %d\n", queueRemove(&q));
63      printf("remove queue %d\n", queueRemove(&q));
64      printf("remove queue %d\n", queueRemove(&q));
65
66      return 0;
67  }
```

```
add queue 123
add queue 456
remove queue 123
remove queue 456
add queue 1234
add queue 5566
add queue 1314
remove queue 1234
remove queue 5566
remove queue 1314
```

# 課堂心得



今日課程教了二元樹Binary tree、堆積stack、佇列queue，這次雖然教了許多比較抽象的概念，但老師有教要怎麼將其轉換成程序用法。

在今天的課程中，我遇到了比較多的問題點，在二元樹的部分，一直看不懂二元樹的用法，我找了很多網路上的影片，也試著想從程式中找出一點端睨，試圖去更了解的程式，但最後還是很困惑，我自己疑惑了兩、三天也想不通，只不過我最後去問了老師，才獲得了答案，我也因此才弄明白。但不只是二元樹，我在堆積、佇列的程式中也遇到了不少困難，雖然明白他們的運作原理，但是在寫程式的時候還是遇到了許多不理解，尤其是將某些排序數值改變的部分，那部分的程式我花了好幾個小時的時間都沒弄明白。只不過最後還是成功的做出簡報了，雖然在文字解釋上花了不小的功夫，但至少我感覺我對程式越來越能明白了。

很可惜今天是最後一堂課，過幾個月就要考APCS，老師有問我要不要去試著報名，而我也選擇了去報名看看，雖然在剛學習完這門課程後，對程式還是有點生疏，但還是試著去報名看看了。

# 自我心得

*註:這幾篇心得是在處理報告時所想出來的，故日期不同。

(4/29)

我認為APCS考試是一個很好的機會，讓我可以展現我的程式設計能力，也讓我可以有更多的選擇去申請我想要就讀的大學校系。我知道有些大學校系有開設APCS組，他們會參考APCS的成績來篩選學生，而且APCS組的學測級分要求比一般組低很多。這對於像我這樣對資工資管有興趣但是學測級分不夠高的學生來說，是一個很大的優勢。我希望我可以通過APCS考試，並且進入我的理想大學。

我覺得這個課程非常有幫助，讓我對程式設計有了更深入的了解和興趣，也讓我更有信心去面對APCS考試。

# 自我心得

(6/4)
在考完APCS後，雖然知道自己的成績不會到很好，但至少我也勇敢的踏出了這一步了，雖然主要是自己對於程式上的造詣還沒到很熟練，但在考試之前我也遇到了不少阻因，剛好鄰近考試，學校老師也開始在強調學測大考，因此我那時把重心從程式設計放到了學校的考試，因此我這篇簡報在中間也停止了一段時間處理。

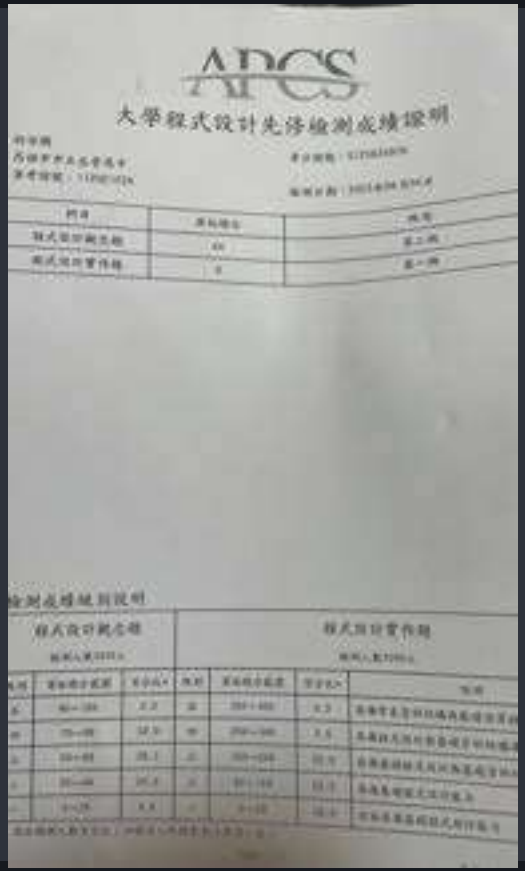
(7/2)
只不過又因為我發現到，自己是對程式充滿了興趣，尤其是近日ai的科技日異月薪，又重新燃回了自己對程式的熱愛，而且現在也是暑假的時間，我花了不少時間去聽了外面的研習講座，也在網路上看了許多有關人工智慧的發展，像是openai，我親自去試用過後，真的是大受震驚，沒想到人工智慧真的能厲害到這種地步，不僅僅有許多問題都可以問它，甚至在一些數據報告中，它的邏輯思維和運算都能處理許多複雜的問題，其程度甚至在美國的某些頂尖大學內的考試順利通過，因此這讓我用對人工智慧產生了更多興趣，讓我更堅定了未來走向資工系的道路。

# 自我心得

(8/8)
今日是將簡報整個整理完的日子，故目前的心得也只寫至今日。

在統整完簡報後，我明顯地感覺到自己在程式方面有所提升，不僅僅是能輕易解決一些原本看不懂的程式，更是將一些抽象概念終於釐清，最大的心得就是完成了這份長達百頁的報告，因為最重要的一點，我原本做這份簡報是帶著迷茫的心情去做的，而後我現在已經沒有了迷茫的念頭，反而都是堅毅，我希望未來自己能在這條路上繼續走。
我已經能清楚的明白自己對於程式的熱愛，再加上最近的ChatGPT、BingAi…等等的許多的人工智慧對話機器人，讓我更迫不及待的能踏進這個領域了，因此我打算在之後做一篇有關人工智慧的簡報，另外，在寫簡報的過程中，我也遇到了許多的有趣程式邏輯問題，像是皇后問題、騎士巡迴，或者是一些概念，像是爬蟲、演算法，我對這些東西非常的感到興趣，因此我希望自己之後也能在這些方面小有成就。
總結來說，非常感謝有這門課，一路走來的心路歷程都是我心境上的變化與成長，更多的是面對自己心靈中的堅毅，這門課不僅學到了許多有趣的程式，更多的是心路的歷程。

# 自我心得


APCS

大學程式設計先修檢測成績證明

比較遺憾的是，因為課程上完後，緊隨而來的就是考試和其他的學校作業要面對，因此在那段時間我暫時沒去接觸程式，只能在空閒之餘學習程式，而那時候我這份簡報也才做不到二十幾頁，沒有完全複習到上課的內容，也因此這次的成績不是這麼的理想。

只不過在之後我將老師上課的內容都整理出來，並做成了這份簡報，為了更快提升對程式的理解，我將所學到的程式都進行了剖析和統整。回首過去，才發現到原本對自己來說很難的題目，似乎變得不怎麼有難度了，也因此我才知道自己在程式已經有許多的進步了，但還是有更多的進步空間，也因此我希望自己能在程式這條路上繼續走下去，獲取更好的成就，並且能在往後的APCS中取得更好的成績。


研習證書

二年十四班

柯安聰 同學

於民國 112 年 02 月 25 日至 04 月 29 日期間，參加本校主辦之 112 年度「APCS 程式語言課程(檢測衝刺班)」，共計 56 小時，特此證明。

校長 張簡錦增

中華民國 112 年 04 月 29 日

# Thank you!