

壹、 成果作品

```
命令提示字元 - C:\Users\andre\Desktop\Git資料庫\PublicLibr...
Microsoft Windows [版本 10.0.19042.867]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\andre>C:\Users\andre\Desktop\Git資料庫\PublicLibrary\
MyProjects\排座位\排座位.exe
===== 207-隨機座位 =====

按Enter以啟動程式...

===== 設定視角 =====

請選擇視角：(輸入數字1或2)
(1)看向講台
(2)看向桌椅
回答：1

===== 設定模式 =====

請選擇顯示方式：(輸入數字1或2或3)
(1)同時顯示號碼
(2)逐一顯示號碼
(3)按Enter鍵顯示號碼
回答：1

===== 設定完成 =====

===== 開始執行 =====

                                [講台]

3           31           17           20           4           19           24
23          25          42          39          16          18          14
38           2           35          28           9           34          26
30           22           8           7           6           33          15
40           27           21          11          13          36           1
           29           12          37          32          43

===== 執行完畢 =====

輸入"again"可再次產生新座位
輸入"reset"可重新設定
輸入"end"可結束程式
回答：
```

圖一、隨機換座位程式(執行結果)

圖一是隨機換座位程式的執行結果，隨機換座位是我編寫的第一個小工具程式，靈感來自於班導師利用 Geogebra¹製作的換座位工具，想嘗試利用當時有學過的基本 C++ 語法，自己編寫一個能同樣達到隨機換座位效果的程式。圖一為程式的執行結果，由圖可見裡面有編寫一些小功能，如可以改變視角，控制講台在上方或是下方，也可以改變顯示模式，增加隨機換座位的趣味。

¹ GeoGebra -- 自由的數學工具--全世界超過一億人使用 Url: <https://www.geogebra.org/?lang=zh-TW>

```

206 void fn_SetSeatNum()
207 {
208     for(int i = 1; i <= 39; i++)
209     {
210         int j;
211
212         do
213         {
214             do
215             {
216                 int_SeatNum[i-1] = rand()%43 + 1;
217             }
218             while(int_SeatNum[i-1] == 41 ||
219                 int_SeatNum[i-1] == 10 ||
220                 int_SeatNum[i-1] == 20 ||
221                 int_SeatNum[i-1] == 5);
222
223             for(j = 1; j < i; j++)
224             {
225                 if(int_SeatNum[i-1] == int_SeatNum[j-1]) break;
226             }
227         }
228         while(j != i);
229     }
230
231     return;
232 }

```

圖二、隨機換座位程式(程式碼)

圖二為隨機換座位程式中用來設定隨機座號的函式。圖二中可見，當時使用的方法是利用 do-while 迴圈來產生不重複的隨機座號，再儲存至陣列當中。迎面而來的第一個問題是我們班的座號並不完全連續，因此需要編寫一個巢狀 do-while 迴圈來確保產生的隨機座號存在相對應的學生。

```

232 void fn_FillNum(int int_Num)
233 {
234     int int_Count = 0;
235
236     switch(int_Num)
237     {
238         case 1:
239             for(int i = 0; i < 6; i++)
240             {
241                 for(int j = 0; j < 7; j++)
242                 {
243                     if(i == 0 && j == 3)
244                     {
245                         intarr_V1[0][3] = 20;
246
247                         continue;
248                     }
249
250                     if((i == 5 && j == 0) || (i == 5 && j == 6)) continue;
251
252                     intarr_V1[i][j] = int_SeatNum[int_Count];
253                     int_Count++;
254                 }
255             }
256
257             break;

```

圖三、隨機換座位程式(程式碼)

圖三為將隨機座號儲存至陣列裡的函式，內容是將產生完畢的隨機變數，依照順序，一一儲存至模擬教室座標的二維陣列中。然而編寫這個程式遇到的第二個狀況是20號學生有弱視的問題，因此座位要固定在講桌前面那一格，當我建立二維陣列來模擬班上市椅的橫排與直排時，在講台前那一格需要自行設定座號，如圖三中的 Line 247，而圖三中的 Line 253 則是當該座標沒有擺放桌椅，會直接跳過那一格。

```
C:\Users\andre\AppData\Local\Programs\Python\Python38-32\python.exe
*Choose the number of files to open it
(1) Unit5.txt
(2) WordBox.txt
2
*File import successfully!

*Input a number of the options
(1)Add new words
(2)Remove words
(3)Inspect the file
(4)Clean file
(5)Save contents
(6)Return to lated archive
(7)Vocabulary test
(8)Change file
(9)End the program
7

=====

*Test Start!
1. a_____e n. 青少年時間
adolescence
*Correct!    Score:1
-----

2. m_____e v. 使現代化
modernize
*Correct!    Score:2
-----

3. b_____e a. 離奇的
bizzare
*Wrong answer!
Correct answer:bizarre
Life:9    Score:2
```

圖四、背單字程式(執行結果)

圖四的背單字程式是於高三上學期的時候編寫的程式，同時也是目前唯一一個利用 Python 編寫且具有完整功能的程式，靈感來自各類英文單字軟體，建立專案放入各個自選的英文單字 txt 檔，並利用 Python 的 Read file 與 Write file 功能，以及字串、字元的比對，達成能夠自選單字庫來考試的功能，不過因為沒有特別學習 Read file 和 Write file 的語法，呈現這些單字功能的方式有些簡陋，因此希望在未來學習完整學習 Python 後，能夠將這份專案以更有效率的方式重新編寫。

```

28 #Pack all contents in lines into a list
29 def ScanFile(fname, lstname):
30
31     with open(fname) as f:
32         for line in f:
33             lstname.append(line.strip())
34
35     return lstname
36
37 #Renew the file
38 def RenewFile(fname, lstname):
39
40     with open(fname, 'w') as f:
41         for lst in lstname:
42             f.write(lst + '\n')
43
44 #Pack all contents in words into a list
45 def ScanWord(fname, lstname):
46
47     with open(fname, 'r') as f:
48         for line in f:
49             item = [i for i in line.split()]
50             lstname.append(item)
51
52     return lstname

```

圖五、背單字程式(程式碼)

圖五為背單字程式中，最重要的三個函式。ScanFile 是利用 Read file 功能，以行為單位儲存至陣列中；RenewFile 是利用 Write file 功能，將有變動的陣列各元素，重新逐行寫入檔案中；ScanWord 同樣是利用 Read file 功能，與 ScanFile 不同之處在於 ScanWord 除了逐行讀取之外，還會另外分割單行中的每個元素儲存至陣列中的第二維，如單字、詞性及字義。

```

while True:

    if life == 0:

        print('\n*Test Over!\n Please practice more!\n\n' + 30 * '=' + '\n')

        break

    elif count == len(numlst):

        print('\n*Congratulations!\n You have answered all the questions!'
              + '\n Test Over!\n\n' + 30 * '=' + '\n')

        break

    else:

        rnum = numlst[count]-1

        rword = words2[rnum][0]
        lst = list(rword)

        for i in range(1, len(lst) - 1):
            lst[i] = '_'

        sub = str()

        for i in range(0, len(lst)):
            sub += str(lst[i])

        print(str(count+1) + '. ' + sub + ' ' + words2[rnum][1] + ' ' + words2[rnum][2] + '\n')

        ans = str(input())

        if ans == rword:

            count += 1
            count2 += 1

            print('\n*Correct!   Score:' + str(count2) + '\n\n' + 30 * '-' + '\n')

        else:

            life -= 1
            count += 1

            print('\n*Wrong answer!\n\n Correct answer:' + rword + '\n Life:' + str(life)
                  + '   Score:' + str(count2) + '\n\n' + 30 * '-' + '\n')

```

圖六、背單字程式(程式碼)

圖六為單字測驗模式的程式碼。利用圖五的 ScanFile 函式儲存每行的資料，再透過遮蔽單字中的一些字母，作為題目輸出至畫面中，使測驗者可以進行判斷並輸入答案，再利用 ScanWord 函式，可以直接與輸入的答案進行字串的比對，即可達到單字測驗的效果。

```
下午2:22 4G+ 60%
× Terminal
deg f(x) = 2

積分上值(分子): 2
積分上值(分母): 1

積分下限(分子): 5
積分下限(分母): 1

x 2次方的系數(分子): 1
x 2次方的系數(分母): 1

x 1次方的系數(分子): -7
x 1次方的系數(分母): 1

x 0次方的系數(分子): 10
x 0次方的系數(分母): 1

Ans: 9/2
Process finished.
```

圖七、定積分計算機(執行結果)

圖七為定積分計算機的輸入與輸出，輸入分為三個部分，第一部分是積函數的最高次數；第二部分是積分的上、下限；第三部分是積函數的各項係數，為了增加實用性，因而設計使用者可以輸入分數係數，得到的答案也會以分數來表示。

```

8  struct Coefficient // 係數結構
9  {
10     int int_Numer = 0;    //分子
11     int int_Denom = 1;    //分母
12
13     void operator-= (Coefficient &coef_Num) //定義此結構的複合指定減號運算子
14     {
15         int int_TempNumer = coef_Num.int_Numer;
16         int int_TempDenom = coef_Num.int_Denom;
17
18         int int_LCM = fn_LCM(this->int_Denom, coef_Num.int_Denom);
19
20         this->int_Numer *= int_LCM / this->int_Denom;    // 通分過程
21         this->int_Denom = int_LCM;
22         coef_Num.int_Numer *= int_LCM / coef_Num.int_Denom;
23         coef_Num.int_Denom = int_LCM;
24
25         this->int_Numer -= coef_Num.int_Numer; // 分數相減
26
27         int int_GCD = fn_GCD(this->int_Numer, this->int_Denom);
28
29         this->int_Numer /= int_GCD;    // 約分過程
30         this->int_Denom /= int_GCD;
31
32         coef_Num.int_Numer = int_TempNumer;
33         coef_Num.int_Denom = int_TempDenom;
34     }
35 };

```

圖八、定積分計算機(程式碼)

圖八的內容為這個定積分計算程式中最重要的部分，目的是為了可以輸入分數的係數，畢竟在求反導函數的時候，常常出現係數是分數的狀況，因此編寫了這個係數結構，內含分子及分母兩個整數欄位，在計算時則將分子以及分母分開運算，通分時運用 `fn_LCM` 求最小公倍數；約分時運用 `fn_GCD` 求最大公因數，則可以保證每次進行運算後，各個係數都會化簡至最簡分數。


```

45  int main()
46  {
47      std::cout<< "deg f(x) = ";
48      std::cin>> int_Deg;
49
50      Coefficient* coefptr_Max = new Coefficient;    // 上值
51      Coefficient* coefptr_Min = new Coefficient;    // 下值
52      Coefficient* coefptr_Arr = new Coefficient[int_Deg + 2];    // 各係數
53
54      fn_Preset(coefptr_Max, coefptr_Min, coefptr_Arr);
55      fn_Antiderivative(coefptr_Arr);
56      fn_Calculate(coefptr_Max, coefptr_Arr, &coef_MaxRes);
57      fn_Calculate(coefptr_Min, coefptr_Arr, &coef_MinRes);
58
59      delete coefptr_Max;
60      delete coefptr_Min;
61      delete [] coefptr_Arr;
62
63      coef_MaxRes -= coef_MinRes; // 上函數值 - 下函數值
64
65      if(coef_MaxRes.int_Denom == 1) std::cout<< "Ans: " << coef_MaxRes.int_Numer;
66      else std::cout<< "Ans: " << coef_MaxRes.int_Numer << '/' << coef_MaxRes.int_Denom;
67
68      return 0;
69  }

```

圖九、定積分計算機(程式碼)

圖九為定積分計算機的主函式程式碼，其中我練習利用動態記憶體來實作這個定積分計算機，並利用函式進行輸入積分各項目數值、將積函數轉為反導函數，最後將上、下限的值代入反導函數，得出計算結果，若是分數則以分數形式輸出。

class 類別

是一種自定義資料型別

主要著重在方法設計與延伸

包含資料成員與成員函式

類別定義必須利用存取標籤 (access label) `public` 或 `private` 將成員的權限歸類

- | | | |
|-------------------------------------|---|---|
| • <code>Public</code>
可隨意類別外部做存取 | • <code>Private</code> (C++ 預設)
只能在類別內部做存取 | • <code>Protected</code>
允許在子類別內部做存取 |
|-------------------------------------|---|---|

this 指標

- `this->`
- 用來識別參數名稱與資料成員名稱

💡 參數名稱與資料成員名稱相同時，要明確使用 `this` 指標 ->

Scope Resolution Operator 範疇解析運算子

- 範疇解析運算子 `::`
- 用來表示 `bmi()` 屬於 `class person` 的方法

Encapsulation 封裝

將資料成員與成員函式依功能劃分為公有與私有

包裝在一個類別內

保護私有成員

使它不會直接受到外部的存取

圖十、Notion 筆記內容(一)

圖十、圖十一是我利用 Notion²這個筆記軟體整理的筆記，選擇 Notion 來整理筆記的原因是它的操作便利性，Notion 的文字方塊可以任意拖曳移動，能夠快速進行排版，第二個原因是 Notion 擁有特殊的文字方塊，可以寫下多種語言的程式碼，方便直接寫下語法的範例。而這些筆記的內容主要為 C++ 的語法、易錯觀念以及 ZeroJudge³實作題目的解題思維，這些語法多數是在國立臺北科技大學的程式先修課程學到的重點，例如指標與動態記憶體配置、物件導向程式設計以及泛型設計等。其中又以指標與動態記憶體配置最為複雜，在筆記中有很大的篇幅是著重於這個部分，因為只有 C/C++ 允許使用者直接對記憶體進行操控，是這個程式語言的特色，同時也是 C/C++ 編譯速度最快的原因之一。

² Notion – The all-in-one workspace for your notes, tasks, wikis, and databases. Url: <https://www.notion.so/>

³ ZeroJudge – 高中生程式解題系統 Url: <https://zerojudge.tw/>

函式的多載

- 解決執行內容相同，但引數不同的問題

💡 引數的不同可以是**型態不同**或**個數不同**

- 有**相容匹配問題**(請避免發生)

參數預設值

- 不能使用分離式宣告**
- 有順序問題
P(0, , 4)這樣呼叫**會報錯**

💡 必須將要初始化的參數都**擺後面**

inline 內行展開函式

- 用於加速**簡短函式**的**執行速度**
- 將簡短函式**展開**至 main函式，取代**頻繁的換手過程**

```
inline int fn_Add(int int_Num1, int int_Num2)
{
    return int_Num1 + int_Num2;
}
```

💡 編譯器會**自動判斷**是否執行內行展開的動作，若不執行則視為普通函式，**不影響輸出結果**

Generic Programming 泛型設計

- 開發一堆相同的多載程式

圖十一、Notion 筆記內容(二)