古典密碼學C++實作與音樂密碼

自主學習計畫暨研究成果與心得

作者：陳嘉翎

自主學習計畫

自主學習成果

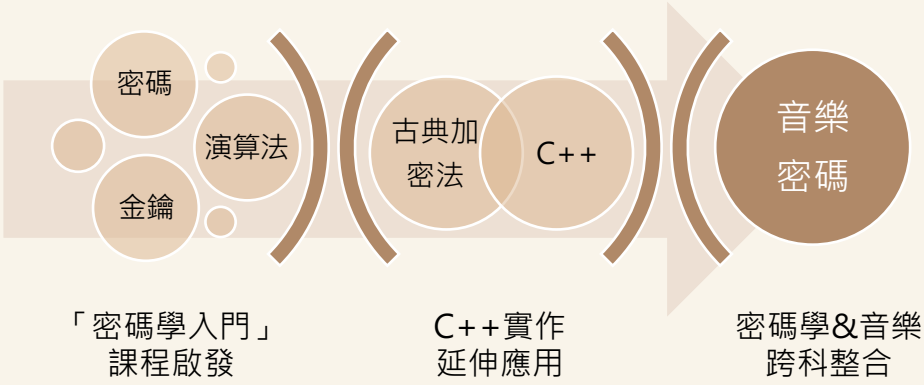
古典密碼學程式實作

- 1 密碼學與古典密碼學
- 2,3 幾何換位法
- 4,5 金鑰式行換位法
- 6,7,9 凱薩加密法
- 10-12 仿射密碼
- 13 頻率分析

音樂密碼

- 14 音樂密碼簡介
- 15,16 音樂密碼代表作曲家
- 17,18 簡單音樂密碼樂句創作

學習脈絡



學習摘要

學習主題	**古典密碼學C++實作與音樂密碼**	
學習單元	古典密碼學 C++實作	音樂密碼 跨科整合
學習時間	1-13週	14-18週
作品屬性	延伸應用	跨科整合（密碼學＋音樂）
學習動機與目的	由「密碼學入門」課程啟發，考慮手動解密時的**費時**情形，決定設計一款能**自行加密與解密**的程式作品。	多元的古典密碼類型使我聯想到音樂課介紹過的**音樂密碼**，運用自身專長，想深入瞭解音樂密碼背後的原理、相關作曲家及其作品。
內容說明	結合**C++程式編寫能力**，透過已知的解密流程，進一步設計成完整的程式作品，**改善密碼破譯效率**。	**連結密碼學與音樂兩大領域**，自主學習音樂密碼原理、相關作曲家與其作品，並利用音符與簡譜對照表，將自行設計的**英文短句轉換為音樂樂句**。

節次	計畫內容
1	了解「密碼學」、「古典密碼學」並製作簡單介紹
2	「幾何換位法」範例與流程圖製作
3	「幾何換位法」C++編寫與實際測試
4	「金鑰式行換位法」範例與流程圖製作
5	「金鑰式行換位法」C++編寫與實際測試
6	「凱撒加密法」範例與流程圖製作
7	「凱撒加密法」C++編寫與實際測試
8	「仿射密碼」範例與流程圖製作
9	「仿射密碼」C++編寫與實際測試
10	「維吉尼爾加密法」範例與流程圖製作
11	「維吉尼爾加密法」C++編寫與實際測試
12	「維吉尼爾加密法」C++編寫與實際測試
13	了解「頻率分析」並製作簡單介紹
14	了解「音樂密碼」並製作簡單介紹
15	了解「音樂密碼代表作曲家與樂曲」並製作簡單介紹
16	了解「音樂密碼代表作曲家與樂曲」並製作簡單介紹
17	運用音樂密碼轉換方法設計簡單樂句
18	運用音樂密碼轉換方法設計簡單樂句

依計畫執行 計畫變更

節次	執行進度	執行日期	完成
1	搜尋密碼學、古典密碼學相關資料並製作簡單介紹	2/12	✓
2	「幾何換位法」定義、範例、加解密流程圖製作	2/15	✓
3	「幾何換位法」C++編寫與範例測試	2/17	✓
4	「金鑰式行換位法」定義、範例、加解密流程圖製作	2/19	✓
5	「金鑰式行換位法」C++編寫與範例測試	2/20	✓
6	「凱撒加密法」由來、範例、加解密流程圖製作	2/21	✓
7	「凱撒加密法」C++編寫與範例測試	2/22	✓
8	複習函數（functions）應用	2/23	✓
9	「凱撒加密法」運用函數優化程式作品	2/25	✓
10	「仿射密碼」定義、範例、加解密流程圖製作	2/26	✓
11	「仿射密碼」C++編寫	2/27	✓
12	「仿射密碼」C++編寫與範例測試	3/1	✓
13	了解「頻率分析」並製作簡單介紹	3/5	✓
14	了解「音樂密碼」並製作簡單介紹	3/6	✓
15	了解1位音樂密碼代表作曲家與樂曲並製作簡單介紹	3/8	✓
16	了解2位音樂密碼代表作曲家與樂曲並製作簡單介紹	3/9	✓
17	尋找合適音符轉換法、簡譜對照表製作	3/11	✓
18	運用音樂密碼轉換方法設計簡單樂句	3/12	✓

加解密原理不同，但想同時呈現在同一程式作品

問題

加密和解密都是密碼學重要的部分，且兩者的**流程不相同、有不同的程式碼**。想同時呈現兩者的成果，但又希望能將同一類型密碼製作成**單一份程式作品**。

解決

① 編寫程式時，使用 **if 迴圈**分別寫出加密和解密程式

② 程式開始執行時，先以**二選一問題**選擇想要加密或解密，並經由程式判斷引導到正確的解題流程。

程式碼多有重複內容，想整合、精簡程式碼

問題

到了第8週的凱撒加密法，雖完成程式編寫，但有許多步驟在**執行重複的內容**，因此想**整合、精簡**程式碼。

解決

我想到了函數的運用，因為對函數概念不夠熟悉，編寫遇到困難，決定**找回暑期營隊的課程簡報**，**更改自主學習計畫**並利用一堂課複習，熟悉原理後，透過函數整合重複程式碼，完成**程式優化**。

與音名對應的字母少，限制樂句創作

問題

在撰寫自主學習計畫時，規劃創作簡單並隱含密碼的旋律，但深入研究音樂密碼與代表樂曲後，發現在大多數的作品**僅使用音名創作成簡單的人名或縮寫**，字母通常限制在A~G之間，如此一來便侷限了音樂密碼的創作。

解決

搜尋其他的音樂密碼轉換法，使用**音樂簡譜**Do~Si對應1~7搭配英文字母A~Z對應的1~26，採用尾數1~7的字母，這樣一來就能使用**20個字母**排列組合成英文句子。

連結相關經驗，展現靈活的變通特質

從密碼學聯想到C++，接著又連結音樂密碼，這樣快速結合相關經驗、**充滿想像力與新點子**的過程，展現了我靈活的變通特質，**舉一反三發揮無限創意**。

記取執行經驗，有效安排最適合的學習計畫

高一下開始摸索自主學習，先後嘗試了線上課程、學校微課程，最後都未能完成計畫，因為撰寫計畫時，設定**太緊迫的進度與目標**，導致每星期在短短的一小時內，無法順利完成。經過反思，這次的密碼學程式實作，我**依照平時寫程式的時間，安排每週可行的執行計畫**，也依據每週的狀況適時做**滾動式調整**，例如：新增凱撒密碼程式優化、增加較難的仿射密碼編寫時間，最後雖然捨去了部分內容，但也因此呈現出較好的程式作品，**完成自主學習計畫**。

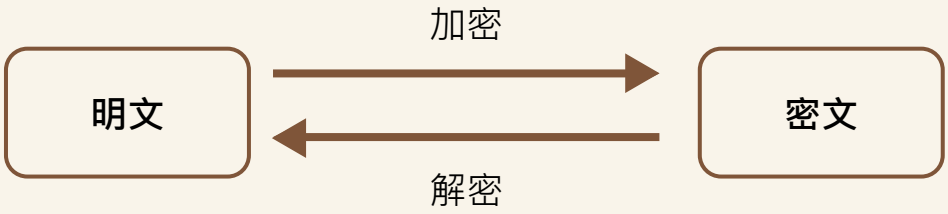
迎面接受挑戰，探索未知領域

從「密碼學入門」課堂遇到的**密碼破譯效率問題**出發，**聯想應用**C++程式編碼增加加解密效率，培養**遇到問題主動尋求解方，結合專長嘗試克服困難**的能力；同時，想到音樂課僅短暫介紹過的音樂密碼，在好奇心的驅使下，藉由自主學習的機會，延伸了解音樂密碼和作曲家，最後，**透過 MuseScore 軟體編寫簡單樂句，完成創作**。自主學習對我來說，是自我探索、**主動尋求答案、結合自身能力精進學習與了解欲探索領域**的過程，即使與平時課堂學習有極大的差異，遇到困難須想辦法自行解決，但我很喜歡這種**探索未知領域、接受挑戰**的學習過程，同時相信自己在未來繼續嘗試自主學習其他領域時，也能克服困難勇往直前。

密碼學簡介

密碼學（Cryptography）可分為古典密碼學和現代密碼學，在英文中，密碼學一詞源於希臘語*kryptós*「**隱藏的**」和*gráphein*「**書寫**」。古典密碼學主要探討資訊的**保密書寫與傳遞**，現代密碼學則進一步關注資訊的完整性驗證、不可抵賴性與**資安**問題。

加解密由演算法及金鑰決定：加密演算法為**明文**（普通資訊）**轉換到密文**（較難理解的資料）的過程，解密演算法則相反；**金鑰**是一個用於加解密演算法的秘密參數，通常只有通訊者擁有，缺乏秘密訊息、演算法與金鑰的人皆無法解讀密碼。



古典密碼學簡介

古典密碼學僅考慮訊息的**機密性**，將可理解資訊轉為困難資訊，並使擁有秘密訊息者能**逆向回覆**（使用相同金鑰可加密也可解密）。古典密碼大致可分為以下兩類：

移轉式密碼：將字母順序重新排列。

實例：斯巴達密碼棒、**幾何換位法**、**金鑰式行換位法**

替換式密碼：有系統地將一組字母換成其他字母或符號。

實例：**凱撒加密法**、**仿射密碼**、維吉尼爾加密法

幾何換位法

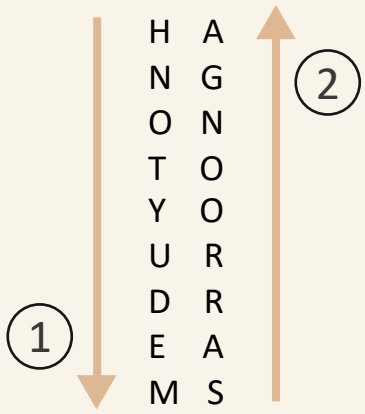
D2(2/15)
D3(2/17)

改變字串書寫方向（字母總數為偶數）

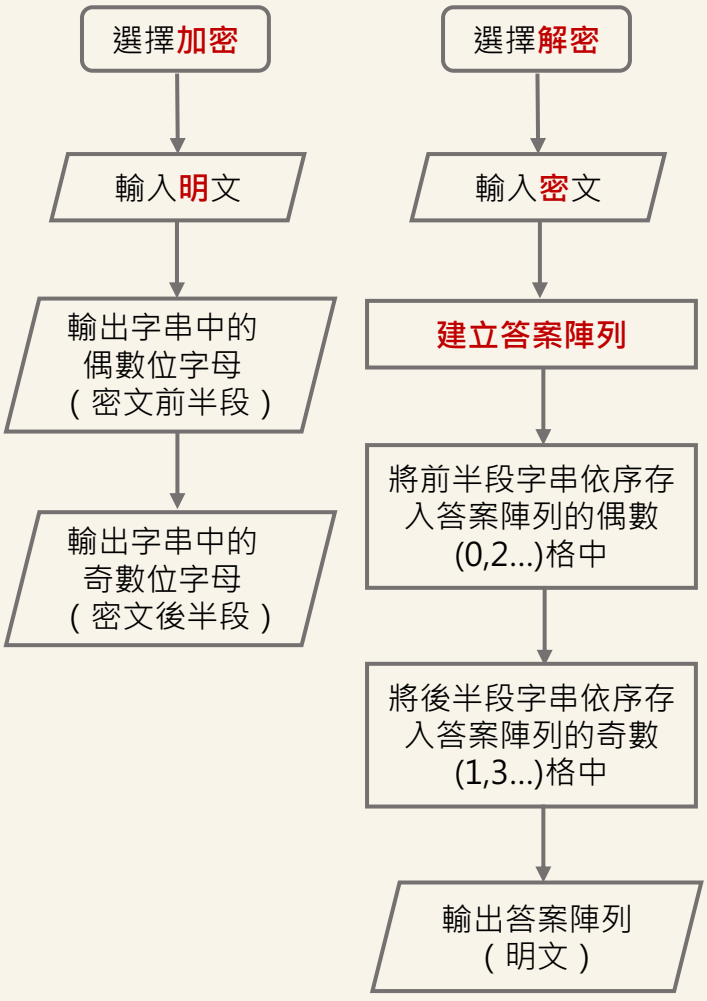
範例

明文：HANG ON TO YOUR DREAMS

密文：HNOTYUDEMAGNOORRAS



我的加/解密流程



我的程式作品

```cpp
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char str[100000]={};
    int model;
    cout<<"請選擇"<<endl<<"1)明文轉密文"<<endl<<"2)密文轉明文"<<endl;
    cin>>model;
    if(model==1){
        cout<<"請輸入明文(不含空格)"<<endl;
        cin>>str;                       //輸入明文
        long long n=strlen(str);        //明文長度
        cout<<"密文：";
        for(long long i=0;i<n;i+=2){ //輸出偶數位字母
            cout<<str[i];
        }
        for(long long i=1;i<n;i+=2){ //輸出奇數位字母
            cout<<str[i];
        }
    }
    else{
        cout<<"請輸入密文(不含空格)"<<endl;
        cin>>str;                       //輸入明文
        long long n=strlen(str);        //明文長度
        char ans2[100000];
        for(long long i=0;i<n;i++){
            if(i<n/2){          //前半段字串存入陣列偶數格
                ans2[2*i]=str[i];
            }
            else{               //後半段字串存入陣列奇數格
                ans2[(i-n/2)*2+1]=str[i];
            }
        }
        cout<<"明文：";
        for(long long i=0;i<n;i++){  //輸出答案陣列（明文）
            cout<<ans2[i];
        }
    }
    cout<<endl;
    return 0;
}
```

金鑰式行換位法

D4(2/19)
D5(2/20)

// 古典密碼學C++實作與音樂密碼

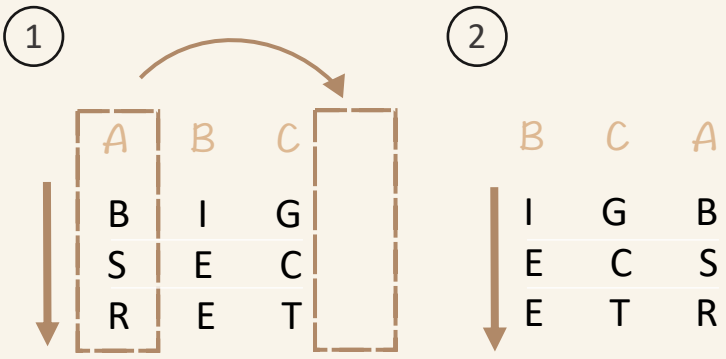
指定明文和密文間行數的轉換關係

範例

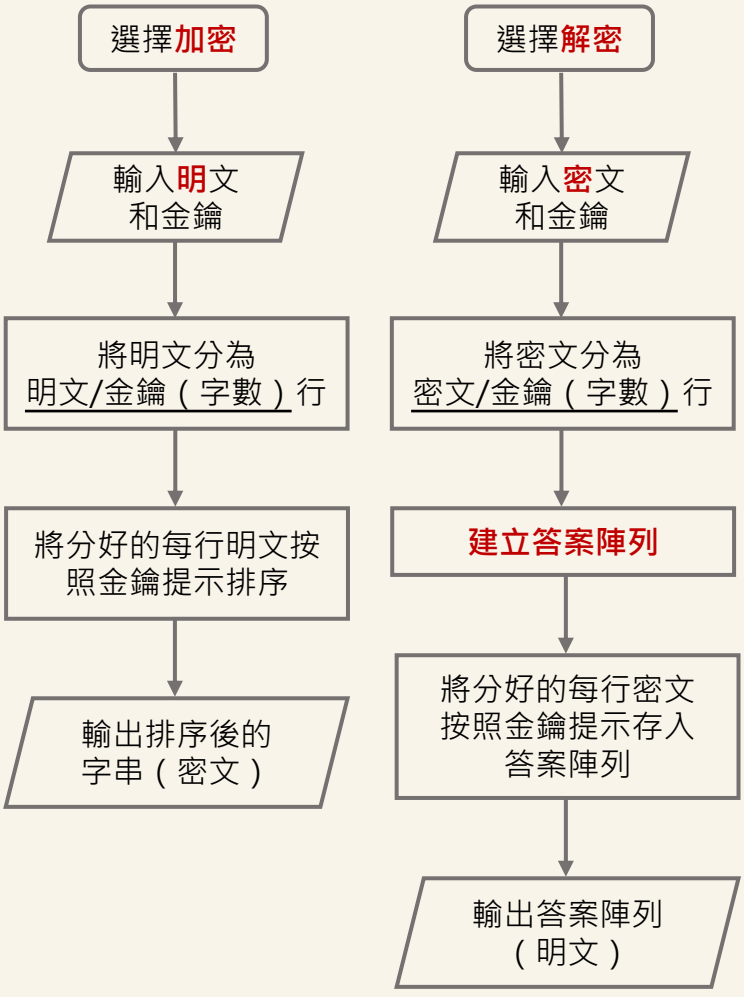
明文：BIGSECRET

金鑰：B C A

密文：IGBECSETR



我的加/解密流程



我的程式作品

```cpp
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char str[100000]={},key[30]={};
    int model,key_order;
    cout<<"請選擇"<<endl<<"1)明文轉密文"<<endl<<"2)密文轉明文"<<endl;
    cin>>model;
    if(model==1){
        cout<<"請輸入明文(不含空格)"<<endl;
        cin>>str;                  //輸入明文
        long long n=strlen(str); //明文長度
        cout<<"請輸入金鑰(大寫)"<<endl;
        cin>>key;                  //輸入金鑰
        long long m=strlen(key); //金鑰長度
        cout<<"密文：";
        for(int i=0;i<n/m;i++){  //按照金鑰提示排序並輸出
            for(int j=0;j<m;j++){
                key_order=key[j]-'A';
                cout<<str[m*i+key_order];
            }
        }
    }
    else{
        cout<<"請輸入密文(不含空格)"<<endl;
        cin>>str;                  //輸入密文
        long long n=strlen(str); //密文長度
        cout<<"請輸入金鑰"<<endl;
        cin>>key;                  //輸入金鑰
        long long m=strlen(key); //金鑰長度
        char ans2[100000]={};
        for(int i=0;i<n/m;i++){  //按照金鑰提示存入答案陣列
            for(int j=0;j<m;j++){
                key_order=key[j]-'A';
                ans2[m*i+key_order]=str[m*i+j];
            }
        }
        cout<<"明文：";
        for(long long i=0;i<n;i++){  //輸出答案陣列（明文）
            cout<<ans2[i];
        }
    }
    cout<<endl;
    return 0;
}
```

凱撒加密法

1. 西元前一世紀，凱撒大帝用來傳遞訊息。
2. 公式：$f(x) = x + k(mod\ 26)$
   - x：該字在字集中一開始的位置
   - k：移動的量

範例 當 $k = 3$ 時

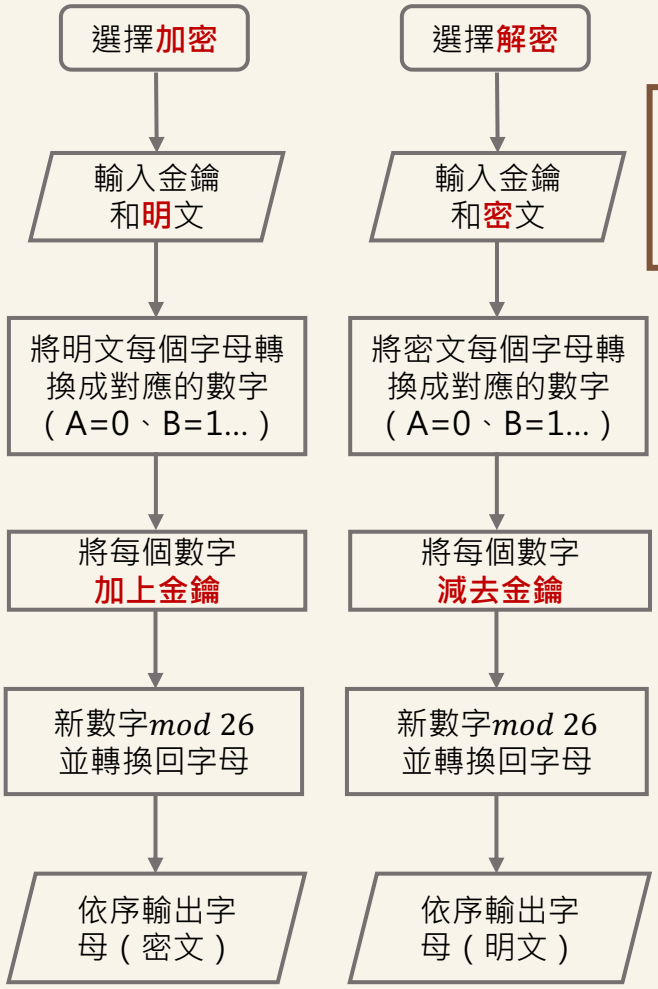
明文	A	B	C	D	E	F	G	H	I	J	K	L	M
x	0	1	2	3	4	5	6	7	8	9	10	11	12
$x+3$	3	4	5	6	7	8	9	10	11	12	13	14	15
$mod\ 26$	3	4	5	6	7	8	9	10	11	12	13	14	15
密文	D	E	F	G	H	I	J	K	L	M	N	O	P

明文	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
x	13	14	15	16	17	18	19	20	21	22	23	24	25
$x+3$	16	17	18	19	20	21	22	23	24	25	26	27	28
$mod\ 26$	16	17	18	19	20	21	22	23	24	25	0	1	2
密文	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

明文：IDONOTCARE

密文：LGRQRWFDUH

我的加/解密流程



優化後程式作品

```cpp
#include<iostream>
#include<string>
using namespace std;

void mod(char str[],long long n,int A){
    for(int i=0;i<n;i++){
        while(str[i]-A<0)    //測試新的字串每個字母是否在A-Z範圍內
            str[i] = str[i]+26; //若新字串超出範圍則將超出字母向前（後）位移26個字元
        while(str[i]-A>25)
            str[i] = str[i]-26;
    }
}

char str[100000]={};
int main(){
    int k,model;
    int A=char('A');
    cout<<"請選擇"<<endl<<"1)明文轉密文"<<endl<<"2)密文轉明文"<<endl;
    cin>>model;
    cout<<"請輸入k"<<endl;
    cin>>k;                         //輸入金鑰
    if(model==1){
        cout<<"請輸入明文(需大寫、不含空格)"<<endl;
        cin>>str;                   //輸入明文
        long long n=strlen(str); //明文長度
        cout<<"密文：";
        for(int i=0;i<n;i++)
            str[i] += k;            //將明文每個字往後移k個字母
        mod(str,n,A);               //移動字串範圍至A-Z
        cout<<str;                  //輸出密文
    }
    else{
        cout<<"請輸入密文(需大寫、不含空格)"<<endl;
        cin>>str;                   //輸入密文
        long long n=strlen(str); //密文長度
        cout<<"明文：";
        for(int i=0;i<n;i++)
            str[i] -= k;            //將明文每個字往前移k個字母
        mod(str,n,A);               //移動字串範圍至A-Z
        cout<<str;                  //輸出明文
    }
    cout<<endl;
    return 0;
}
```

```cpp
            str[i] += k;          //將分好的明文每個字往後移k個字母
            while(str[i]-A<0){    //測試新的字串每個字母是否在A-Z範圍內
                str[i] = str[i]+26; //若新字串超出範圍則將超出字母向前（後）位移26個字元
            }
            while(str[i]-A>25){
                str[i] = str[i]-26;
            }
            cout<<str[i];         //輸出密文
```

```cpp
            str[i] -= k;          //將分好的明文每個字往前移k個字母
            while(str[i]-A<0){    //測試新的字串每個字母是否在A-Z範圍內
                str[i] = str[i]+26; //若新字串超出範圍則將超出字母向前（後）位移26個字元
            }
            while(str[i]-A>25){
                str[i] = str[i]-26;
            }
            cout<<str[i];
```

```cpp
void mod(char str[],long long n,int A){
    for(int i=0;i<n;i++){
        while(str[i]-A<0)    //測試新的字串每個字母是否在A-Z範圍內
            str[i] = str[i]+26; //若新字串超出範圍則將超出字母向前（後）位移26個字元
        while(str[i]-A>25)
            str[i] = str[i]-26;
    }
}
```

優化內容

- 將程式**相同片段整合成函數**，需要使用時呼叫函數。
- 原本程式是每個字單獨輸出，經過函數整合後，將變換後的字元存回字串，**密碼全部破譯後再統一輸出**。

1. 以一對一函數關係進行加密。
2. 公式：$f(x) = ax + b(mod\ 26)$
   - a, b為正整數
   - a 和 26 互質

範例 當 a = 3 , b = 2 時

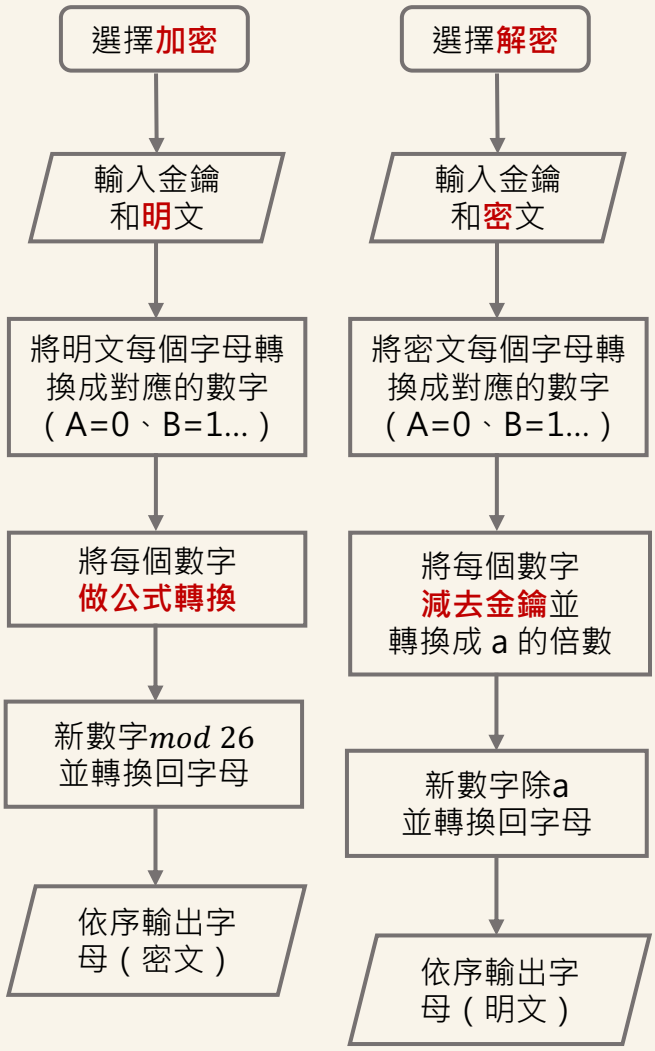
明文	A	B	C	D	E	F	G	H	I	J	K	L	M
x	0	1	2	3	4	5	6	7	8	9	10	11	12
$3x + 2$	2	5	8	11	14	17	20	23	26	29	32	35	38
$mod\ 26$	2	5	8	11	14	17	20	23	0	3	6	9	12
密文	C	F	I	L	O	R	U	X	A	D	G	J	M

明文	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
x	13	14	15	16	17	18	19	20	21	22	23	24	25
$3x + 2$	41	44	47	50	53	56	59	62	65	68	71	74	77
$mod\ 26$	15	18	21	24	1	4	7	10	13	16	19	22	25
密文	P	S	V	Y	B	E	H	K	N	Q	T	W	Z

明文：LOUISACOFFEE

密文：JSKAECISRROO

我的加/解密流程



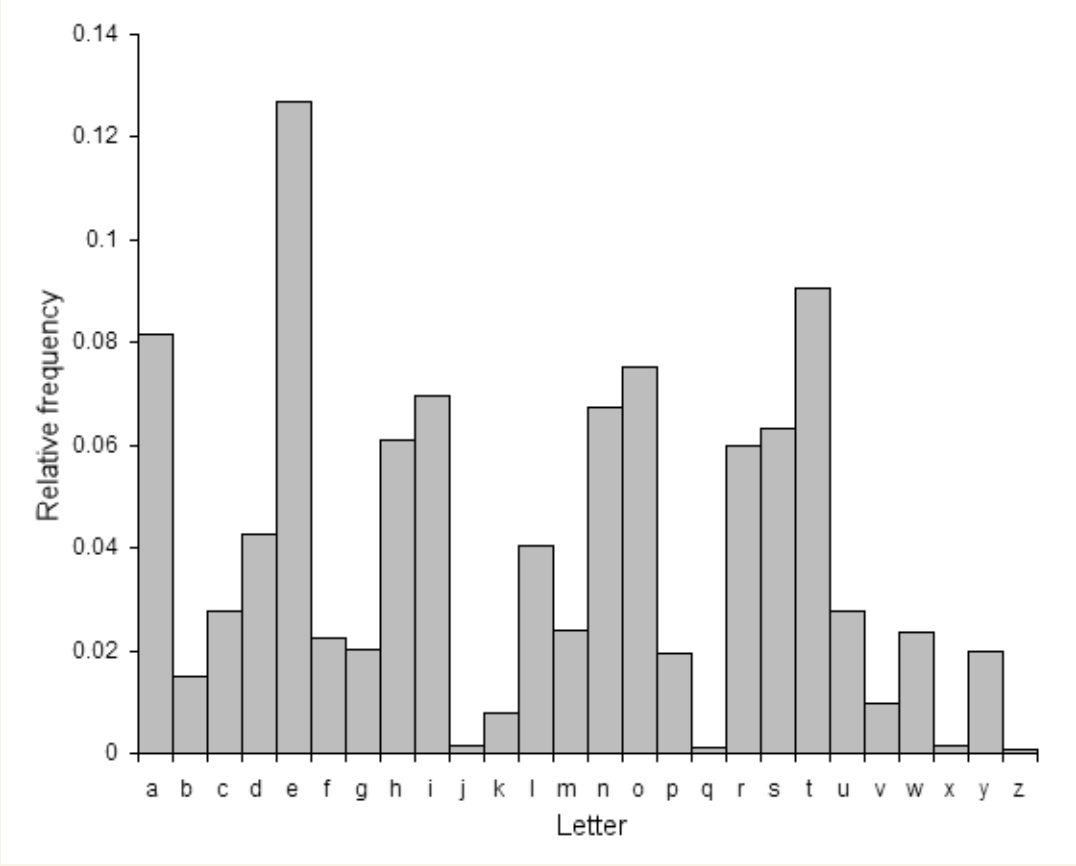
我的程式作品

```cpp
#include<iostream>
#include<string>
using namespace std;

int num_mod(int x){
    while(x<0)                      //轉換為正數
        x += 26;
    x = x % 26;                     //mod26
    return x;
}

int A=char('A');
char str[100000]={};
int main(){
    int a=0,b=0,model;
    int A=char('A');
    cout<<"請選擇"<<endl<<"1)明文轉密文"<<endl<<"2)密文轉明文"<<endl;
    cin>>model;
    cout<<"a="<<endl;
    cin>>a;                         //輸入金鑰
    cout<<"b="<<endl;
    cin>>b;                         //輸入金鑰
    if(model==1){
        cout<<"請輸入明文(需大寫、不含空格)"<<endl;
        cin>>str;                   //輸入明文
        long long n=strlen(str);    //明文長度
        cout<<"密文：";
        for(int i=0;i<n;i++){
            int x=0;
            x = int(str[i])-A;      //轉成數字0-25
            x = x*a+b;              //數字縮放a後+b
            x = num_mod(x);         //數字mod26
            str[i] = char(A+x);     //將0-25數字轉回字母
        }
        cout<<str;                  //輸出密文
    }
    else{
        cout<<"請輸入密文(需大寫、不含空格)"<<endl;
        cin>>str;                   //輸入密文
        long long n=strlen(str);    //密文長度
        cout<<"明文：";
        for(int i=0;i<n;i++){
            int x=0;
            x = int(str[i])-A;      //轉成數字0-25
            x = num_mod(x-b);       //數字mod26
            while(x % a != 0)       //轉成a的倍數
                x += 26;
            x = x/a;                //除以a
            str[i] = char(A+x);     //將0-25數字轉回字母
        }
        cout<<str;                  //輸出密文
    }
    cout<<endl;
    return 0;
}
```

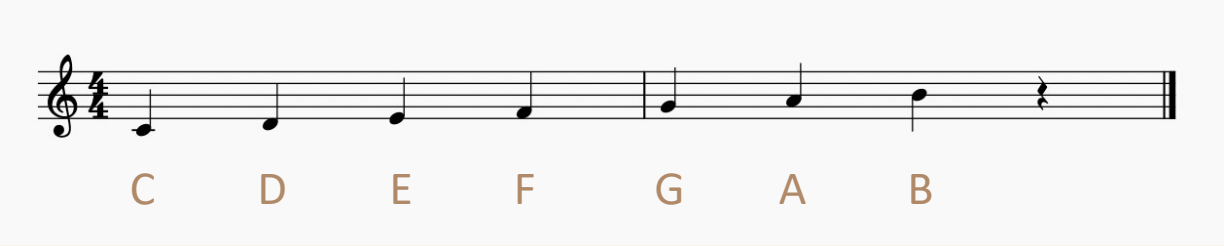
在密碼學中，頻率分析指是一種針對**字母在文本中出現頻率**的研究分析，不同語言會有不同的分析結果，但**同一語言會出現大致相同的字母特徵分佈**。在英文字母中（如圖一），字母「e」最常出現，其次是「t」，而「q」、「z」則較少出現。因此，除了經由程式作品破解密碼，在大部分古典密碼中，運用頻率分析也可以**快速破解密文**。然而這種方便的破解方法也可能導致這些密碼**不需知道金鑰與演算法**，就能迅速被破解的情形。



圖一

音樂密碼源於**17-18世紀**，相較於利用信件或文本傳遞密碼，使用樂曲傳遞訊息**難以被察覺**，也較不會被懷疑。但如此強大的加密法至今卻仍未發現用於傳遞情報或機密訊息，大多數作曲家僅透過此方法將自己或朋友的**名字隱藏在樂句中**，讓樂曲留下自己的蹤跡。

大部分的音樂加密使用音符的**音名**（如圖二所示），然而這樣的密碼**限制了音符的變化**，僅能表達A～G，因此在不同的作品上，音樂密碼的定義也會出現些微的不同。



圖二

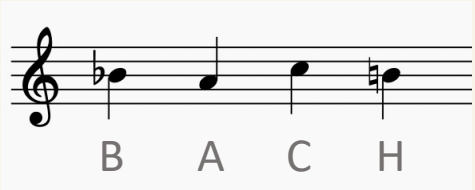
音樂密碼代表作曲家

W15(3/8), W16(3/9)

巴赫（Johann Sebastian Bach）

作曲家簡介

- 出生地：德國
- 時期：巴洛克時期
- 別稱：音樂之父



圖三

音樂密碼代表樂曲

《賦格的藝術》賦格曲Contrapunctus XIV

代表樂曲密碼說明

不同於我們熟悉的音名用法，德國的字母「B」代表音名B♭、字母「H」則代表音名B，因此若將 **B♭、A、C、B♮** 四個音連用（如圖三）並轉換成音名，正好就是巴赫的名字**BACH**。而這樣的密碼稱為「**B-A-C-H動機**」，曾出現在巴赫的許多賦格作品中。

布拉姆斯（Johannes Brahms）

作曲家簡介

- 出生地：德國
- 時期：浪漫主義

音樂密碼代表樂曲

- 《FAE奏鳴曲》
- 《G大調弦樂六重奏》

代表樂曲密碼說明

《FAE奏鳴曲》：這首曲子的**FAE**源自於**座右銘「Frei aber Einsam」**（自由卻孤獨），以此為動機發展整首樂曲。

《G大調弦樂六重奏》：亦被稱為《阿嘉德六重奏曲》，相傳是為**舊情人**阿嘉德（Agathe）所作，曲中以**A G A H E**（省去T）奏出阿嘉德名字，透過樂曲昇華自身情感。

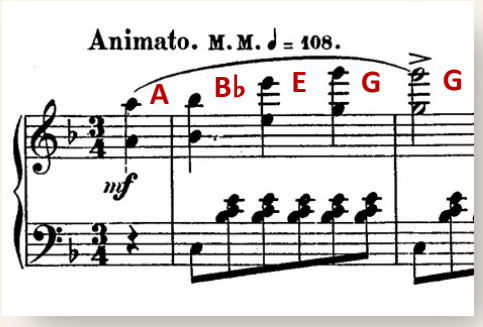
舒曼（Robert Alexander Schumann ）

作曲家簡介

- 出生地：德國
- 時期：浪漫主義
- 別稱：音樂詩人

音樂密碼代表樂曲

- 《阿貝格變奏曲》
- 《狂歡節》



《阿貝格變奏曲》片段

代表樂曲密碼說明

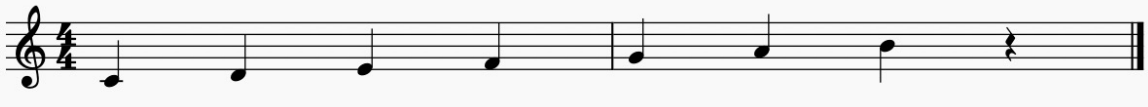
《阿貝格變奏曲》：世人對曲中的阿貝格（Abegg）有兩種猜想，可能是舒曼的**虛構朋友**，或是曾經的愛慕對象。曲中連用**A-B♭-E-G-G**，以**ABEGG**為主題動機譜出優美樂曲。

《狂歡節》：為鋼琴組曲，因為在德文中「Es」代表音名E♭（Es發音為s），若將**A-E♭-C-B**連用，可代表ASCH，正好是**舒曼未婚妻名字**。此外，德文的**狂歡節**（Fasching）也巧妙隱藏了這四個字母，以此寓意作為動機貫穿全曲。

音樂密碼樂句創作

依照音樂的簡譜記法，音符Do～Si可分別用數字 1~7 表示，為了使音樂密碼創造出更完整且更多變化的樂句，我結合了音符與簡譜的對照法，將自行設計的簡單英文句子轉換為樂句。

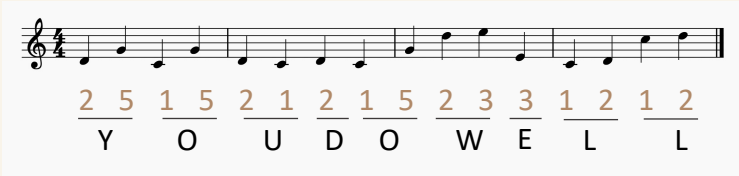
音符與簡譜對照



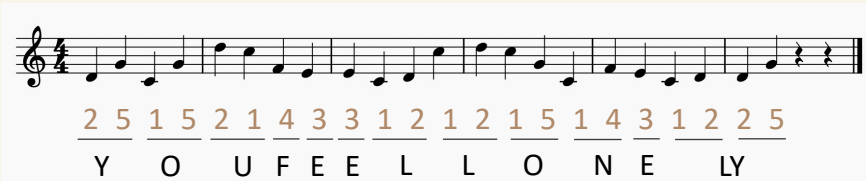
字母	A	B	C	D	E	F	G	H	I	J
對應數字	1	2	3	4	5	6	7	8	9	10
字母	K	L	M	N	O	P	Q	R	S	T
對應數字	11	12	13	14	15	16	17	18	19	20
字母	U	V	W	X	Y	Z				
對應數字	21	22	23	24	25	26				

自創樂句（課後自主創作）

1. YOU DO WELL.



2. YOU FELL LONELY.



在真正使用音樂密碼創作時，我才發覺這種加密方式**比想像中複雜**，如何在少去6字母的限制下，運用**有邏輯的句子**，創作出好聽的音樂旋律，成了活用音樂密碼最具挑戰性的目標。