二次曲線判斷機

一、設計動機:

我高中時做的科展是有關二次曲線與射影幾何(https://www.ntsec.edu.tw/Science-Content.aspx?cat=12948&a=6821&fld=&key=&isd=1&icop=10&p=1&sid=16488)，而我們主要使用的軟體是 Geogebra(GGB)，透過 GGB 我們可以大大的減輕我們在研究上的困難，但我們發現如果繪製的圖形一多，GGB 就會變得相當卡頓，這時如果還要去求二次曲線的各種性質，又會大大增加電腦的負擔，所以我想要把能夠運算二次曲線各種性質的程式獨立出來，讓 GGB 的負荷量減輕一些。

二、輸入與輸出:

A.輸入:二次曲線一般式 $ax^2+bxy+cy^2+dx+ey+f=0$ 的六個常數

B.輸出:

情形	處理方法
1.一點	算出該點座標
2.兩平行直線	算出兩直線方程式
3.兩重合直線	算出兩直線方程式
4.兩相交直線	算出兩直線方程式
5.圓	算出圓心座標、半徑 r
6.橢圓	算出中心座標、焦點、半長軸長、半短軸長、焦距、逆時針轉幾度會變正(消除 xy 項)
7.拋物線	算出頂點座標、焦點、焦距、逆時針轉幾度會變正(消除 xy 項)
8.雙曲線	算出中心座標、焦點、半貫軸長、半共軛軸長、焦距、逆時針轉幾度會變正(消除 xy 項)
9.空集合	不處理

C.範例:

1.一點

```
C:\Users\蔡弘祥\Desktop\計程\個人報告1\二次曲線判斷.exe
請輸入二次曲線一般式的六個係數abcdef，注意abc不能全部為0，否則不為二次曲線
1
2
2
2
-2
5
判斷結果:
點
座標(-3,2)
--------------------------------
Process exited after 5.416 seconds with return value 0
請按任意鍵繼續 . . .
```

2.兩平行直線

```
C:\Users\蔡弘祥\Desktop\計程\個人報告1\二次曲線判斷.exe
請輸入二次曲線一般式的六個係數abcdef，注意abc不能全部為0，否則不為二次曲線
9
12
4
-9
-6
-10
判斷結果:
兩平行直線
直線1為:x+0.666667y=1.66667
直線2為:x+0.666667y=-0.666667
--------------------------------
Process exited after 10.92 seconds with return value 0
請按任意鍵繼續 . . .
```

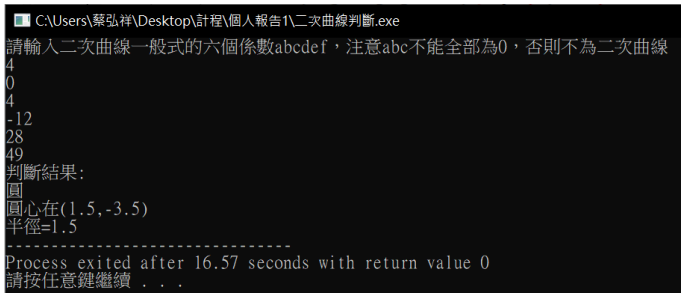
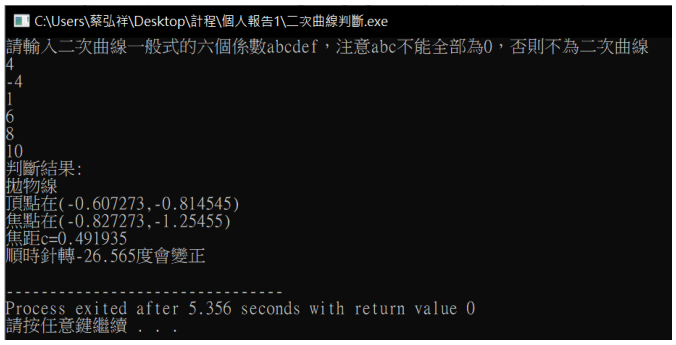
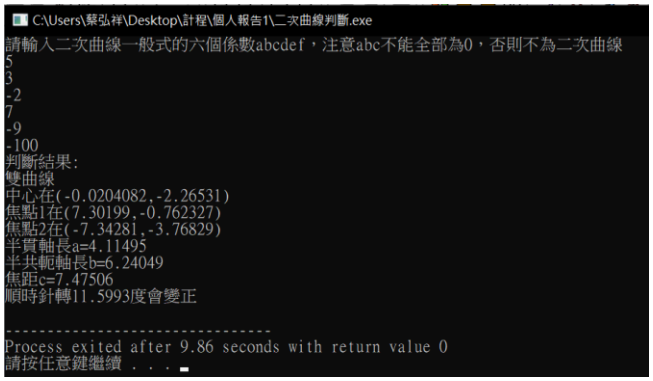
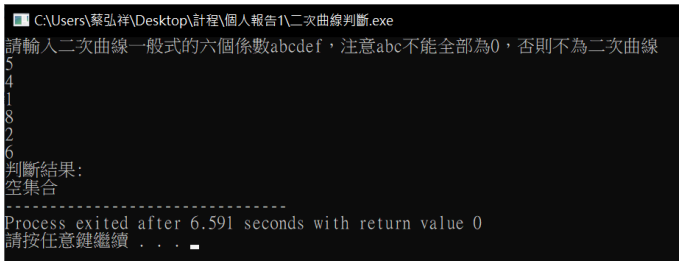
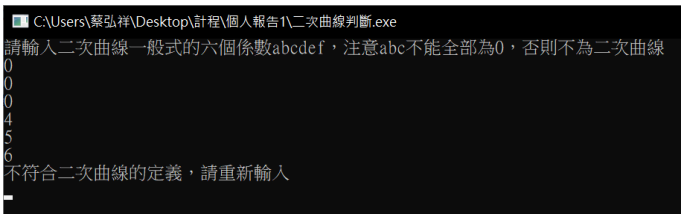
3.兩重合直線

```
C:\Users\蔡弘祥\Desktop\計程\個人報告1\二次曲線判斷.exe
請輸入二次曲線一般式的六個係數abcdef，注意abc不能全部為0，否則不為二次曲線
-4
12
-9
-20
30
-25
判斷結果:
兩重合直線
直線為:x-1.5y=-2.5
--------------------------------
Process exited after 16.19 seconds with return value 0
請按任意鍵繼續 . . .
```

4.兩相交直線

```
C:\Users\蔡弘祥\Desktop\計程\個人報告1\二次曲線判斷.exe
請輸入二次曲線一般式的六個係數abcdef，注意abc不能全部為0，否則不為二次曲線
-6
29
-28
-11
6
10
判斷結果:
兩相交直線
直線1為:x-1.33333y=0.666667
直線2為:x-3.5y=-2.5

--------------------------------
Process exited after 13.36 seconds with return value 0
請按任意鍵繼續 . . .
```

5.圓	6.橢圓
	
7.拋物線	8.雙曲線
	
9.空集合	10.錯誤輸入
	

三、運作方式與步驟:

1.先請使用者輸入二次曲線一般式 $ax^2+bxy+cy^2+dx+ey+f=0$ 的六個常數，得到六個常術後先判斷是否符合二次方程式的定義

2.將二次曲線分類成九種不同的情形

3.對各情形進行處理

四、程式碼解說:

步驟 1: 第 6-12 行

```
 6:      cout<<"請輸入二次曲線一般式的六個係數abcdef，注意abc不能全部為0，否則不為二次曲線"<<endl;
 7:      float a=0,b=0,c=0,d=0,e=0,f=0;   //宣告二次曲線一般式的6個係數
 8:      cin>>a>>b>>c>>d>>e>>f;
 9:      while(a==0&&b==0&&c==0){
10:          cout<<"不符合二次曲線的定義，請重新輸入"<<endl;
11:          cin>>a>>b>>c>>d>>e>>f;
12:      }
```

首先要求使用者輸入二次曲線的六個變數，並且使用 while 來判斷輸入的資料是否有符合二元二次方程式的定義

步驟 2: 第 13-52 行

```
13:     float H=0,delta_1=0,delta_2=0,omega=0;   //宣告四種不同的判別式
14:     H=a+c;   //計算判別式H
15:     delta_1=b*b-4*a*c;   //計算判別式delta_1
16:     delta_2=4*a*c*f+b*e*d-c*d*d-a*e*e-b*b*f;      //計算判別式delta_2
17:     omega=d*d+e*e-4*(a+c)*f;      //計算判別式omega
18:     int condition=0;      //二次曲線的狀態變數
```

宣告圓錐曲線判別式，其中 delta1 為 δ、delta2 為 Δ、omega 為 Ω、condition 為狀態變數(方便用來進行步驟 3 的計算)，下表為二次曲線的判斷方法

判別式：$H=a+c$ ；$\delta=b^2-4ac$ ；$\Delta=\frac{1}{2}\begin{vmatrix} 2a & b & d \\ b & 2c & e \\ d & e & 2f \end{vmatrix}$ ；$\Omega=d^2+e^2-4(a+c)f$ 。

	$\delta<0$	$\delta=0$	$\delta>0$
$\Delta=0$	1.一點	2.若 $\Omega>0$ ，則為兩平行直線 3.若 $\Omega=0$ ，則為兩重合直線 9.若 $\Omega<0$ ，則為空集合	4.兩相交直線
$\Delta\neq0$	9. $H\cdot\Delta>0$ 空集合 5,6. $H\cdot\Delta<0$ 圓(5)或橢圓(6)	7.拋物線	8.雙曲線

```
20:     if(delta_1<0){
21:         if(delta_2==0){
22:             condition=1;     //一點
23:         }else{
24:             if(H*delta_2>0){
25:                 condition=9;     //空集合
26:             }else{
27:                 if(a==c&&b==0){
28:                     condition=5;     //圓
29:                 }else{
30:                     condition=6;     //橢圓
31:                 }
32:             }
33:         }
```

先處理 $\delta<0$ ，再處理 $\Delta=0$ 與 $\Delta\neq0$ ，值得注意的地方是我們利用圓的一般式沒有 xy 項且 x^2 和 y^2 的係數相同來分別圓和橢圓

```
34:     }else if(delta_1==0){
35:         if(delta_2==0){
36:             if(omega>0){
37:                 condition=2;     //兩平行直線
38:             }else if(omega==0){
39:                 condition=3;     //兩重合直線
40:             }else if(omega<0){
41:                 condition=9;     //空集合
42:             }
43:         }else{
44:             condition=7;     //拋物線
45:
```

先處理 δ=0 ，再處理 $\Delta=0$ 與 $\Delta\neq0$

```
46:     }else if(delta_1>0){
47:         if(delta_2==0){
48:             condition=4;     //兩相交直線
49:         }else{
50:             condition=8;     //雙曲線
51:         }
52:     }
```

先處理 $\delta>0$ ，再處理 $\Delta=0$ 與 $\Delta\neq 0$

步驟 3: 第 53-305 行

```
54:     cout<<"判斷結果:"<<endl;
55:     switch(condition){
```

先輸出所有情況都需要輸出的”判斷結果” ，接著使用 switch 來進行個情況的處理

```
56:         case 1:      //一點
57:         {
58:             float x=0,y=0;
59:             y=(2*a*e-b*d)/(b*b-4*a*c);
60:             x=(b*y+d)/(-2*a);
61:             cout<<"點"<<endl;
62:             cout<<"座標("<<x<<","<<y<<")";
63:             break;
64:         }
```

先假設算出的點座標為 (x,y)

接著將 $ax^2+bxy+cy^2+dx+ey+f=0$ 看成是 x 的一元二次方程式，即

$ax^2+(by+d)x+(cy^2+ey+f)=0$ ，並對 x 進行求解，又因為 x 重根($x=\frac{by+d}{-2a}$)，所以判別式

$(by+d)^2-4a(cy^2+ey+f)=0$ ，對於這個 y 的一元二次方程式 y 也有重根，可得 $y=\frac{2ae-bd}{b^2-4ac}$ ，

再將 y 代回 x 即可求得座標 (x,y)

```
65:         case 2:     //兩平行直線
66:           {
67:               float a_0=0,b_0=0,c_1=0,c_2=0,x=0,y=0;
68:               if(a>0){
69:                   a_0=sqrt(a);
70:                   b_0=b/(2*a_0);
71:                   x=d/a_0;
72:                   y=f;
73:                   c_1=(x-sqrt(x*x-4*y))/2;
74:                   c_2=(x+sqrt(x*x-4*y))/2;
75:               }else if(a<0){
76:                   a_0=sqrt(-a);
77:                   b_0=-b/(2*a_0);
78:                   x=-d/a_0;
79:                   y=-f;
80:                   c_1=(x-sqrt(x*x-4*y))/2;
81:                   c_2=(x+sqrt(x*x-4*y))/2;
82:               }else{
83:                   c_1=(-e-sqrt(e*e-4*c*f))/(2*c);
84:                   c_2=(-e+sqrt(e*e-4*c*f))/(2*c);
85:               }
86:               //cout<<a_0<<endl<<b_0<<endl<<c_1<<endl<<c_2<<endl;
87:               cout<<"兩平行直線"<<endl;
88:               if(a_0*b_0!=0){
89:                   if(b_0/a_0>0){
90:                       cout<<"直線1為:"<<"x+"<<b_0/a_0<<"y="<<-c_1/a_0<<endl;
91:                       cout<<"直線2為:"<<"x+"<<b_0/a_0<<"y="<<-c_2/a_0;
92:                   }else{
93:                       cout<<"直線1為:"<<"x"<<b_0/a_0<<"y="<<-c_1/a_0<<endl;
94:                       cout<<"直線2為:"<<"x"<<b_0/a_0<<"y="<<-c_2/a_0;
95:                   }
96:               }else if(a==0){
97:                   cout<<"直線1為:"<<"y="<<c_1<<endl;
98:                   cout<<"直線2為:"<<"y="<<c_2;
99:               }else{
100:                  cout<<"直線1為:"<<"x="<<-c_1/a_0<<endl;
101:                  cout<<"直線2為:"<<"x="<<-c_2/a_0;
102:              }
103:
104:              break;
105:          }
```

處理部分:

我希望能夠把 $ax^2+bxy+cy^2+dx+ey+f=0$ 寫成 $(a_0x+b_0y+c_1)(a_0x+b_0y+c_2)=0$ ，為了方便計算

取 $|a|=a_0{}^2$

Case1: $a>0$

接著透過比較係數可得$\begin{cases} a_0=\sqrt{a} \\ b_0=\dfrac{b}{2\sqrt{a}} \\ c_1+c_2=\dfrac{d}{\sqrt{a}} \\ c_1c_2=f \end{cases}$，不失一般性可假設$c_1<c_2$，解出

$\begin{cases} c_1=\dfrac{\frac{d}{\sqrt{a}}-\sqrt{(\frac{d}{\sqrt{a}})^2-4f}}{2} \\ c_2=\dfrac{\frac{d}{\sqrt{a}}+\sqrt{(\frac{d}{\sqrt{a}})^2-4f}}{2} \end{cases}$，程式中先假設$\begin{cases} x=\dfrac{d}{\sqrt{a}} \text{ 以減少電腦重複運算相同的東西} \\ y=f \end{cases}$

Case2: $a<0$

接著透過比較係數可得$\begin{cases} a_0=\sqrt{-a} \\ b_0=\dfrac{-b}{2\sqrt{-a}} \\ c_1+c_2=\dfrac{-d}{\sqrt{-a}} \\ c_1c_2=-f \end{cases}$，不失一般性可假設$c_1<c_2$，解出

$\begin{cases} c_1=\dfrac{\frac{-d}{\sqrt{-a}}-\sqrt{(\frac{-d}{\sqrt{-a}})^2+4f}}{2} \\ c_2=\dfrac{\frac{-d}{\sqrt{-a}}+\sqrt{(\frac{-d}{\sqrt{-a}})^2+4f}}{2} \end{cases}$，程式中先假設$\begin{cases} x=\dfrac{-d}{\sqrt{-a}} \text{ 以減少電腦重複運算相同的東西} \\ y=-f \end{cases}$

Case3: $a=0$

很明顯$a_0=0$也成立，即可將兩直線寫成$(y-k_1)(y-k_2)=0$，可發現原二元二次方程式簡化為一元二次方程式，接著我們只要套用公式解($y=\dfrac{-e\pm\sqrt{e^2-4cf}}{2c}$)即可求得兩直線

輸出部分:

接下來就是要處理輸出的部分為了讓輸出好看一點，為避免輸出$5x+0y=10$這種可以寫成$x=2$的形式，所以將輸出分成三種情形

$\begin{cases} \text{斜直線: } x+py=q \\ \text{鉛直線: } x=q \\ \text{水平線: } y=q \end{cases}$，對應的情形為$\begin{cases} \text{斜直線: } a_0b_0\neq 0 \\ \text{鉛直線: } b_0=0, a_0\neq 0 \\ \text{水平線: } a_0=0, b_0\neq 0 \end{cases}$，接著只要針對三種情形作個別處理即可讓輸出好看一點

```
106:        case 3:      //兩重合直線
107:          {
108:              float a_1=0,b_1=0,c_1=0;
109:              if(a>0){
110:                  a_1=sqrt(a);
111:                  b_1=b/(2*a_1);
112:                  c_1=d/(2*a_1);
113:              }else if(a<0){
114:                  a_1=sqrt(-a);
115:                  b_1=-b/(2*a_1);
116:                  c_1=-d/(2*a_1);
117:              }else if(c>0){
118:                  b_1=sqrt(c);
119:                  c_1=e/(2*b_1);
120:              }else{
121:                  b_1=sqrt(-c);
122:                  c_1=-e/(2*b_1);
123:              }
124:              cout<<"兩重合直線"<<endl;
125:              if(a_1*b_1!=0){
126:                  if(b_1>0){
127:                      cout<<"直線為:"<<"x+"<<b_1/a_1<<"y="<<-c_1/a_1;
128:                  }else{
129:                      cout<<"直線為:"<<"x"<<b_1/a_1<<"y="<<-c_1/a_1;
130:                  }
131:              }else if(a_1==0){
132:                  cout<<"直線為:"<<"y="<<-c_1/b_1<<endl;
133:              }else{
134:                  cout<<"直線為:"<<"x="<<-c_1/a_1<<endl;
135:              }
136:              break;
137:          }
```

處理部分:

我希望能夠把 $ax^2+bxy+cy^2+dx+ey+f=0$ 寫成 $(a_1x+b_1y+c_1)^2=0$ ，為了方便計算取 $|a|=a_1^2$

Case1: $a>0$

接著透過比較係數可得 $\begin{cases} a_1=\sqrt{a} \\ b_1=\dfrac{b}{2\sqrt{a}} \\ c_1=\dfrac{d}{2\sqrt{a}} \end{cases}$

Case2: $a<0$

接著透過比較係數可得 $\begin{cases} a_1 = \sqrt{-a} \\ b_1 = \dfrac{-b}{2\sqrt{-a}} \\ c_1 = \dfrac{-d}{2\sqrt{-a}} \end{cases}$

Case3: $a = 0, c > 0$

很明顯 $a_0 = 0$ 也成立，即可將兩直線寫成 $(b_1 y - c_1)^2 = 0$ ，同樣經由比較係數可得 $\begin{cases} b_1 = \sqrt{c} \\ c_1 = \dfrac{e}{2\sqrt{c}} \end{cases}$

Case4: $a = 0, c < 0$

很明顯 $a_0 = 0$ 也成立，即可將兩直線寫成 $(b_1 y - c_1)^2 = 0$ ，同樣經由比較係數可得 $\begin{cases} b_1 = \sqrt{-c} \\ c_1 = \dfrac{-e}{2\sqrt{-c}} \end{cases}$

輸出部分:

基本上輸出格式會兩平行直線差不多

```
138:        case 4:      //兩相交直線
139:          {
140:              cout<<"兩相交直線"<<endl;
141:              if(a!=0){
142:                  float b_1=0,c_1=0,b_2=0,c_2=0;
143:                  b_1=(b-sqrt(b*b-4*a*c))/(2*a);
144:                  b_2=(b+sqrt(b*b-4*a*c))/(2*a);
145:                  c_1=(d*b_1-e)/(a*(b_1-b_2));
146:                  c_2=(d*b_2-e)/(a*(b_2-b_1));
147:                  if(b_1>0){
148:                      cout<<"直線1為:"<<"x+"<<b_1<<"y="<<-c_1<<endl;
149:                  }else if(b_1==0){
150:                      cout<<"直線1為:"<<"x="<<-c_1<<endl;
151:                  }else{
152:                      cout<<"直線1為:"<<"x"<<b_1<<"y="<<-c_1<<endl;
153:                  }
154:                  if(b_2>=0){
155:                      cout<<"直線2為:"<<"x+"<<b_2<<"y="<<-c_2<<endl;
156:                  }else if(b_1==0){
157:                      cout<<"直線2為:"<<"x="<<-c_2<<endl;
158:                  }else{
159:                      cout<<"直線2為:"<<"x"<<b_2<<"y="<<-c_2<<endl;
160:                  }
161:              }else{
162:                  float b_1=0,c_1=0,b_2=0,c_2=0;
163:                  b_2=b;
164:                  b_1=c/b_2;
165:                  c_2=d;
166:                  c_1=f/c_2;
167:                  if(b_1>=0){
168:                      cout<<"直線1為:"<<"x+"<<b_1<<"y="<<-c_1<<endl;
169:                      cout<<"直線2為:"<<"y="<<-c_2/b_2<<endl;
170:                  }else{
171:                      cout<<"直線1為:"<<"x"<<b_1<<"y="<<-c_1<<endl;
172:                      cout<<"直線2為:"<<"y="<<-c_2/b_2<<endl;
173:                  }
174:              }
175:              break;
176:          }
```

處理部分:

目標是把 $ax^2+bxy+cy^2+dx+ey+f=0$ 寫成 $(a_1x+b_1y+c_1)(a_2x+b_2y+c_2)=0$ ，然而這不好直接求解，所以將其分成以下不同的 Case 進行處理

Case1: $a\neq 0$

將 $ax^2+bxy+cy^2+dx+ey+f=0$ 寫成 $(x+b_1y+c_1)(x+b_2y+c_2)=0$，比較係數有 $\begin{cases} b_1+b_2=\dfrac{b}{a} \\ b_1b_2=\dfrac{c}{a} \end{cases}$，不失一般性設 $b_1<b_2$，故可求得 $\begin{cases} b_1=\dfrac{-b-\sqrt{b^2-4ac}}{2a} \\ b_2=\dfrac{-b+\sqrt{b^2-4ac}}{2a} \end{cases}$，而 c_1,c_2 滿足 $\begin{cases} c_1+c_2=\dfrac{d}{a} \\ b_2c_1+b_1c_2=\dfrac{e}{a} \end{cases}$，得 $\begin{cases} c_1=\dfrac{b_1d-e}{a(b_1-b_2)} \\ c_2=\dfrac{b_2d-e}{a(b_2-b_1)} \end{cases}$

Case2: $a=0$

將 $ax^2+bxy+cy^2+dx+ey+f=0$ 必可寫成 $(x+b_1y+c_1)(b_2y+c_2)=0$，比較係數有 $\begin{cases} b_1=\dfrac{c}{b} \\ b_2=b \\ c_1=\dfrac{f}{d} \\ c_2=d \end{cases}$

輸出部分:

基本上輸出格式會兩平行直線差不多

```
177:        case 5:      //圓
178:        {
179:            float h=0,k=0,r=0;  //圓心在(h,k)，半徑為r
180:            h=d/(-2*a);
181:            k=e/(-2*a);
182:            r=sqrt(h*h+k*k-f/a);
183:            cout<<"圓"<<endl;
184:            cout<<"圓心在("<<h<<","<<k<<")"<<endl;
185:            cout<<"半徑="<<r;
186:            break;
187:        }
```

此情況因為 xy 項的係數 $b=0$ 且 x^2 項和 y^2 項的係數相等，滿足 $a=c$，故可將一般式

$ax^2+bxy+cy^2+dx+ey+f=0$ 寫成 $ax^2+ay^2+dx+ey+f=0$，再將 x 和 y 分別配方得

$(x+\dfrac{d}{2a})^2+(y+\dfrac{e}{2a})^2=(\dfrac{d}{2a})^2+(\dfrac{e}{2a})^2-\dfrac{f}{a}$，即可知道圓心座標 $(-\dfrac{d}{2a},-\dfrac{e}{2a})$，另外因為已經確定此情形式圓，所以等號右側 $(\dfrac{d}{2a})^2+(\dfrac{e}{2a})^2-\dfrac{f}{a}>0$，開根號即為半徑

```
188:         case 6:      //橢圓
189:          {
190:              float h=0,k=0;  //中心在(h,k)
191:              h=(b*e-2*c*d)/(4*a*c-b*b);  //詳細推導請看書面報告
192:              k=(b*d-2*a*e)/(4*a*c-b*b);
193:              float theta=0;
194:              theta=atan(b/(a-c))/2;
195:              float a_1=0,c_1=0,f_1=0;
196:              f_1=d*h/2+e*k/2+f;
197:              f_1=-f_1;
198:              a_1=a*pow(cos(theta),2)+b*sin(theta)*cos(theta)+c*pow(sin(theta),2);
199:              c_1=a*pow(sin(theta),2)-b*sin(theta)*cos(theta)+c*pow(cos(theta),2);
200:              float semi_major_axis=0,semi_minor_axis=0;
201:              if(f_1/a_1>=f_1/c_1){
202:                  semi_major_axis=sqrt(f_1/a_1);
203:                  semi_minor_axis=sqrt(f_1/c_1);
204:              }else{
205:                  semi_major_axis=sqrt(f_1/c_1);
206:                  semi_minor_axis=sqrt(f_1/a_1);
207:                  theta+=M_PI/2;
208:              }
209:              float focal_length=0;
210:              focal_length=sqrt(pow(semi_major_axis,2)-pow(semi_minor_axis,2));
211:              float p_1=0,q_1=0,p_2=0,q_2=0;  //(p_1,q_1),(p_2,q_2)為兩焦點
212:              p_1=h+focal_length*cos(theta);
213:              p_2=h-focal_length*cos(theta);
214:              q_1=k+focal_length*sin(theta);
215:              q_2=k-focal_length*sin(theta);
216:              cout<<"橢圓"<<endl;
217:              cout<<"中心在("<<h<<","<<k<<")"<<endl;
218:              cout<<"焦點1在("<<p_1<<","<<q_1<<")"<<endl;
219:              cout<<"焦點2在("<<p_2<<","<<q_2<<")"<<endl;
220:              cout<<"半長軸長a="<<semi_major_axis<<endl;
221:              cout<<"半短軸長b="<<semi_minor_axis<<endl;
222:              cout<<"焦距c="<<focal_length<<endl;
223:              cout<<"順時針轉"<<theta*180/M_PI<<"度會變正"<<endl;
224:              break;
225:          }
```

在求任意橢圓的幾何性質如半長軸長、半短軸長時，最一般的做法是先將其中心平移回原點，再將其轉正(即消除 xy 項)

1.找到中心

中心在原點且非正的橢圓可寫成 $ax^2+bxy+cy^2=f_1$，若將其向右平移 h、向上平移 k，便可將其寫成 $a(x-h)^2+b(x-h)(y-k)+c(y-k)^2=f_1$，展開後再跟 $ax^2+bxy+cy^2+dx+ey+f=0$ 比較係數可知

$$\begin{cases} h=\dfrac{be-2cd}{4ac-b^2} \\ k=\dfrac{bd-2ae}{4ac-b^2} \\ f_1=\dfrac{dh+ek}{2}+f \end{cases}$$

2.轉正

正的橢圓可寫成 $a_1x^2+c_1y^2=f_1$，若將其以原點為中心，逆時針轉 θ 可得

$a_1(x\cos\theta-y\sin\theta)^2+c_1(x\sin\theta+y\cos\theta)^2=f_1$，和 $ax^2+bxy+cy^2=f_1$ 比較係數可知

$$\begin{cases}\theta=\dfrac{\tan^{-1}\dfrac{b}{a-c}}{2}\\ a_1=a\cos^2\theta+b\sin\theta\cos\theta+c\sin^2\theta\\ c_1=a\sin^2\theta-b\sin\theta\cos\theta+c\cos^2\theta\end{cases}$$

3.寫成標準式並算出半長軸長、半短軸長、焦距

橢圓標準式為 $\dfrac{x^2}{a^2}+\dfrac{y^2}{b^2}=1$(此式子的 a,b 與輸入值無關，而較大者為半長軸長，較小者為半短軸長)。將步驟 2 得到的 $a_1x^2+c_1y^2=f_1$ 寫成上述形式為 $\dfrac{x^2}{(\sqrt{\frac{f_1}{a_1}})^2}+\dfrac{y^2}{(\sqrt{\frac{f_1}{c_1}})^2}=1$，則半長軸與半短軸為

$$\begin{cases}\text{半長軸}=\sqrt{\dfrac{f_1}{a_1}}\\ \text{半短軸}=\sqrt{\dfrac{f_1}{c_1}}\end{cases},if\ \sqrt{\frac{f_1}{a_1}}>\sqrt{\frac{f_1}{c_1}}\ 、\ \begin{cases}\text{半長軸}=\sqrt{\dfrac{f_1}{c_1}}\\ \text{半短軸}=\sqrt{\dfrac{f_1}{a_1}}\end{cases},if\ \sqrt{\frac{f_1}{c_1}}>\sqrt{\frac{f_1}{a_1}}$$

，焦距為半長軸平方減半短軸平方後開根號，即 $\sqrt{\left|\dfrac{f_1}{a_1}-\dfrac{f_1}{c_1}\right|}$

4.計算橢圓的兩個焦點

想法為從橢圓中心加上長度為焦距且平行長軸的向量來得到兩焦點，即兩焦點的位置向量可寫成

$$\begin{cases}\overrightarrow{OF_1}=\overrightarrow{OC}+\dfrac{\sqrt{\left|\frac{f_1}{a_1}-\frac{f_1}{c_1}\right|}}{\left|\overrightarrow{v}\right|}\overrightarrow{v}\\ \overrightarrow{OF_2}=\overrightarrow{OC}-\dfrac{\sqrt{\left|\frac{f_1}{a_1}-\frac{f_1}{c_1}\right|}}{\left|\overrightarrow{v}\right|}\overrightarrow{v}\end{cases}$$

，其中 O 為原點、C 為橢圓中心、$\overrightarrow{v}$ 為長軸的方向向量且其 x 分量為正。(若半長軸 $=\sqrt{\dfrac{f_1}{a_1}}$，則 $\overrightarrow{v}=(\cos\theta,\sin\theta)$，若半長軸 $=\sqrt{\dfrac{f_1}{c_1}}$，$\overrightarrow{v}=(\cos(\theta+\frac{\pi}{2}),\sin(\theta+\frac{\pi}{2}))$)

```
226:        case 7:      //拋物線
227:          {
228:              float theta=0;
229:              theta=atan(b/(a-c))/2;
230:              float a_1=0,c_1=0,d_1=0,e_1=0;
231:              a_1=a*pow(cos(theta),2)+b*sin(theta)*cos(theta)+c*pow(sin(theta),2);
232:              c_1=a*pow(sin(theta),2)-b*sin(theta)*cos(theta)+c*pow(cos(theta),2);
233:              d_1=d*cos(theta)+e*sin(theta);
234:              e_1=-d*sin(theta)+e*cos(theta);
235:              float focal_length=0;
236:              float h_1=0,k_1=0;   //宣告旋轉後的頂點座標
237:              float h=0,k=0;   //(h,k)為頂點
238:              float p=0,q=0;   //(p,q)為焦點
239:              if(abs(a_1)>abs(c_1)){
240:                  focal_length=e_1/(-4*a_1);
241:                  k_1=(f/a_1-(d_1*d_1)/(4*a_1*a_1))/(4*focal_length);
242:                  h_1=d_1/(-2*a_1);
243:                  h=h_1*cos(theta)-k_1*sin(theta);
244:                  k=h_1*sin(theta)+k_1*cos(theta);
245:                  p=h+focal_length*cos(theta+M_PI/2);
246:                  q=k+focal_length*sin(theta+M_PI/2);
247:              }else{
248:                  focal_length=d_1/(-4*c_1);
249:                  h_1=(f/c_1-(e_1*e_1)/(4*c_1*c_1))/(4*focal_length);
250:                  k_1=e_1/(-2*c_1);
251:                  h=h_1*cos(theta)-k_1*sin(theta);
252:                  k=h_1*sin(theta)+k_1*cos(theta);
253:                  p=h+focal_length*cos(theta);
254:                  q=k+focal_length*sin(theta);
255:              }
256:              cout<<"拋物線"<<endl;
257:              cout<<"頂點在("<<h<<","<<k<<")"<<endl;
258:              cout<<"焦點在("<<p<<","<<q<<")"<<endl;
259:              cout<<"焦距c="<<abs(focal_length)<<endl;
260:              cout<<"順時針轉"<<theta*180/M_PI<<"度會變正"<<endl;
261:              break;
262:          }
```

在求任意拋物線的幾何性質如頂點、焦點時，最一般的做法是先將轉正(即消除 xy 項)，計算出轉後的頂點和焦距再將其轉回原頂點

1.轉正

對於二元二次方程式 $ax^2+bxy+cy^2+dx+ey+f=0$ 要消除 xy 項的做法就是以原點為中心順時針轉 θ，其中 $\theta=\dfrac{\tan^{-1}\dfrac{b}{a-c}}{2}$，將上式轉正後可寫成 $a_1x^2+c_1y^2+d_1x+e_1y+f=0$ 且 a_1,c_1 其中一項是 0，比較係數後得 $\begin{cases}a_1=a\cos^2\theta+b\sin\theta\cos\theta+c\sin^2\theta\\c_1=a\sin^2\theta-b\sin\theta\cos\theta+c\cos^2\theta\\d_1=d\cos\theta+e\sin\theta\\e_1=-d\sin\theta+e\cos\theta\end{cases}$。然而因為電腦無法精確運算出正確數值，所以 a_1,c_1 其中一項會非常接近 0，即可利用 $|a_1|,|c_1|$ 的大小來判斷誰是 0

2.計算頂點、焦點、焦距

這邊我們要分兩個 Case 進行討論，分別是水平開口的拋物線和鉛直開口的拋物線

Case1: $c_1 = 0$

這個情形為鉛直開口的拋物線，因為鉛直拋物線可寫成 $(x-h)^2 = 4C(y-k)$ ，展開後再乘上某個倍數可寫成 $a_1x^2 + d_1x + e_1y + f = 0$ ，由此可得 $\begin{cases} C = -\dfrac{e_1}{4a_1} \\ h_1 = -\dfrac{d_1}{2a_1} \\ k_1 = \dfrac{\dfrac{f}{a_1} - \dfrac{{d_1}^2}{4{a_1}^2}}{4C} \end{cases}$ ，其中 C 為有向焦距(在程式中為 focal_length)、 (h_1, k_1) 為轉後頂點，再將 (h_1, k_1) 以原點為中心逆時針轉 θ 得 (h, k) ，其中 $\begin{cases} h = h_1\cos\theta - k_1\sin\theta \\ k = h_1\sin\theta + k_1\cos\theta \end{cases}$ ，接著要計算焦點，想法為從拋物線頂點加上長度為焦距且平行軸的向量來得到焦點，即焦點的位置向量可寫成 $\overrightarrow{OF} = \overrightarrow{OV} + \overrightarrow{v}$ ，其中 O 為原點、 V 為拋物線頂點、 $\overrightarrow{v}$ 為軸的方向向量，然而軸的方向向兩有兩個方向，必須先找出要往哪邊，經計算發現在鉛直開口的拋物線其焦點的位置向量算法為 $\overrightarrow{OF} = \overrightarrow{OV} + C(\cos(\theta + \frac{\pi}{2}), \sin(\theta + \frac{\pi}{2}))$

Case2: $a_1 = 0$

這個情形為水平開口的拋物線，因為水平拋物線可寫成 $(y-k)^2 = 4C(x-h)$ ，展開後再乘上某個倍數可寫成 $c_1y^2 + d_1x + e_1y + f = 0$ ，由此可得 $\begin{cases} C = -\dfrac{d_1}{4c_1} \\ h_1 = \dfrac{\dfrac{f}{c_1} - \dfrac{{e_1}^2}{4{c_1}^2}}{4C} \\ k_1 = -\dfrac{e_1}{2c_1} \end{cases}$ ，其中 C 為有向焦距(在程式中為 focal_length)、 (h_1, k_1) 為轉後頂點，再將 (h_1, k_1) 以原點為中心逆時針轉 θ 得 (h, k) ，其中 $\begin{cases} h = h_1\cos\theta - k_1\sin\theta \\ k = h_1\sin\theta + k_1\cos\theta \end{cases}$ ，接著要計算焦點，想法為從拋物線頂點加上長度為焦距且平行軸的向量來得到焦點，即焦點的位置向量可寫成 $\overrightarrow{OF} = \overrightarrow{OV} + \overrightarrow{v}$ ，其中 O 為原點、 V 為拋物線頂點、 $\overrightarrow{v}$ 為軸的方向向量，然而軸的方向向兩有兩個方向，必須先找出要往哪邊，經計算發現在水平開口的拋物線其焦點的位置向量算法為 $\overrightarrow{OF} = \overrightarrow{OV} + C(\cos\theta, \sin\theta)$

```
263:        case 8:      //雙曲線
264:          {
265:              float h=0,k=0;  //中心在(h,k)
266:              h=(b*e-2*c*d)/(4*a*c-b*b);  //詳細推導請看書面報告
267:              k=(b*d-2*a*e)/(4*a*c-b*b);
268:              float theta=0;
269:              theta=atan(b/(a-c))/2;
270:              float a_1=0,c_1=0,f_1=0;
271:              f_1=d*h/2+e*k/2+f;
272:              f_1=-f_1;
273:              a_1=a*pow(cos(theta),2)+b*sin(theta)*cos(theta)+c*pow(sin(theta),2);
274:              c_1=a*pow(sin(theta),2)-b*sin(theta)*cos(theta)+c*pow(cos(theta),2);
275:              float semi_transverse_axis=0,semi_conjucate_axis=0;
276:              if(f_1/a_1>=0){
277:                  semi_transverse_axis=sqrt(f_1/a_1);
278:                  semi_conjucate_axis=sqrt(-f_1/c_1);
279:              }else{
280:                  semi_transverse_axis=sqrt(f_1/c_1);
281:                  semi_conjucate_axis=sqrt(-f_1/a_1);
282:                  theta+=M_PI/2;  //此時theta會變共軛軸與x軸夾角，故須加90度
283:              }
284:              float focal_length=0;
285:              focal_length=sqrt(pow(semi_transverse_axis,2)+pow(semi_conjucate_axis,2));
286:              float p_1=0,q_1=0,p_2=0,q_2=0;  //(p_1,q_1),(p_2,q_2)為兩焦點
287:              p_1=h+focal_length*cos(theta);
288:              p_2=h-focal_length*cos(theta);
289:              q_1=k+focal_length*sin(theta);
290:              q_2=k-focal_length*sin(theta);
291:              cout<<"雙曲線"<<endl;
292:              cout<<"中心在("<<h<<","<<k<<")"<<endl;
293:              cout<<"焦點1在("<<p_1<<","<<q_1<<")"<<endl;
294:              cout<<"焦點2在("<<p_2<<","<<q_2<<")"<<endl;
295:              cout<<"半貫軸長a="<<semi_transverse_axis<<endl;
296:              cout<<"半共軛軸長b="<<semi_conjucate_axis<<endl;
297:              cout<<"焦距c="<<focal_length<<endl;
298:              cout<<"順時針轉"<<theta*180/M_PI<<"度會變正"
299:              break;
300:          }
```

在求任意雙曲線的幾何性質如半貫軸長、半共軛軸長時，最一般的做法是先將其中心平移回原點，再將其轉正(即消除 xy 項)

1.找到中心

中心在原點且非正的雙曲線可寫成 $ax^2+bxy+cy^2=f_1$，若將其向右平移 h、向上平移 k，便可將其寫成 $a(x-h)^2+b(x-h)(y-k)+c(y-k)^2=f_1$，展開後再跟 $ax^2+bxy+cy^2+dx+ey+f=0$ 比較係數可知

$$\begin{cases} h=\dfrac{be-2cd}{4ac-b^2} \\ k=\dfrac{bd-2ae}{4ac-b^2} \\ f_1=\dfrac{dh+ek}{2}+f \end{cases}$$

2.轉正

正的雙曲線可寫成 $a_1x^2+c_1y^2=f_1$，若將其以原點為中心，逆時針轉 θ 可得

$a_1(x\cos\theta - y\sin\theta)^2 + c_1(x\sin\theta + y\cos\theta)^2 = f_1$，和 $ax^2 + bxy + cy^2 = f_1$ 比較係數可知

$$\begin{cases} \theta = \dfrac{\tan^{-1}\dfrac{b}{a-c}}{2} \\ a_1 = a\cos^2\theta + b\sin\theta\cos\theta + c\sin^2\theta \\ c_1 = a\sin^2\theta - b\sin\theta\cos\theta + c\cos^2\theta \end{cases}$$

3.寫成標準式並算出半貫軸長、半共軛軸長、焦距

雙曲線標準式為 $\dfrac{x^2}{a^2} - \dfrac{y^2}{b^2} = 1$ 或 $\dfrac{y^2}{a^2} - \dfrac{x^2}{b^2} = 1$ (此式子的 a,b 與輸入值無關，而 a 為半貫軸長， b 為半共軛軸長)。將步驟 2 得到的 $a_1x^2 + c_1y^2 = f_1$ 寫成上述形式，則半長軸與半短軸為

$$\begin{cases} \text{半貫軸} = \sqrt{\dfrac{f_1}{a_1}} \\ \text{半共軛軸} = \sqrt{-\dfrac{f_1}{c_1}} \end{cases}, if\ \frac{f_1}{a_1} > 0 \text{、} \begin{cases} \text{半貫軸} = \sqrt{\dfrac{f_1}{c_1}} \\ \text{半共軛軸} = \sqrt{-\dfrac{f_1}{a_1}} \end{cases}, if\ \frac{f_1}{a_1} < 0$$ ，焦距為半長軸平方加半短軸平方後開根號，即 $\sqrt{\dfrac{f_1}{a_1} + \dfrac{f_1}{c_1}}$

4.計算雙曲線的兩個焦點

想法為從雙曲線中心加上長度為焦距且平行長軸的向量來得到兩焦點，即兩焦點的位置向量可寫成

$$\begin{cases} \overrightarrow{OF_1} = \overrightarrow{OC} + \dfrac{\sqrt{\left|\dfrac{f_1}{a_1} - \dfrac{f_1}{c_1}\right|}}{\left|\overrightarrow{v}\right|}\overrightarrow{v} \\ \overrightarrow{OF_2} = \overrightarrow{OC} - \dfrac{\sqrt{\left|\dfrac{f_1}{a_1} - \dfrac{f_1}{c_1}\right|}}{\left|\overrightarrow{v}\right|}\overrightarrow{v} \end{cases}$$

，其中 O 為原點、 C 為雙曲線中心、 $\overrightarrow{v}$ 為貫軸的方向向量且其 x 分量為正。(若半貫軸 $= \sqrt{\dfrac{f_1}{a_1}}$ ，則 $\overrightarrow{v} = (\cos\theta, \sin\theta)$ ，若半貫軸 $= \sqrt{\dfrac{f_1}{c_1}}$ ， $\overrightarrow{v} = (\cos(\theta + \frac{\pi}{2}), \sin(\theta + \frac{\pi}{2}))$)

```
301:         case 9:      //空集合
302:         {
303:             cout<<"空集合";
304:             break;
305:         }
```

直接輸出”空集合”