軟體設計 - Shape Function

M023010011 廖志倫

1. 介紹(Introduction)

這次的 子class有正方形、直角三角形以及圓，先由使用者決定，這3個class的參數 : 寬、高和邊長，在使用者輸入完畢之後，依照這些條件，產生對應的物件，有了這些物件之後，在利用class中的function計算周長，面積，在顯示出結果。

1. 系統分析(analysis)

觀察需求之後，我們至少需要3個class來實作這3種形狀，

而這三種形狀，有些具有寬、長；有些不具寬、長，只有邊長，

而且周長及面積的計算皆有所不同，需要分別定義。

接著，使用者要依序給這3種形狀作設定，之後再依序顯示結果。由這樣條件可以看出，可以利用兩個loop，一個用於輸入參數；一個用於顯示結果。

1. 設計(design)

三種形狀的處理都各有不同，但他們都具有「名稱」、「面積」及「周長」這3種屬性，所以我們可以定義基本的class shape，

包含了「名稱」以及「計算面積」、「計算周長」這兩個function的base class。

面積與周長的class由於需要在各形狀的class中重新定義，所以使用了virtual的做法，且是pure virtual，因為base class中沒有定義，所以一定要由 子class來實作。

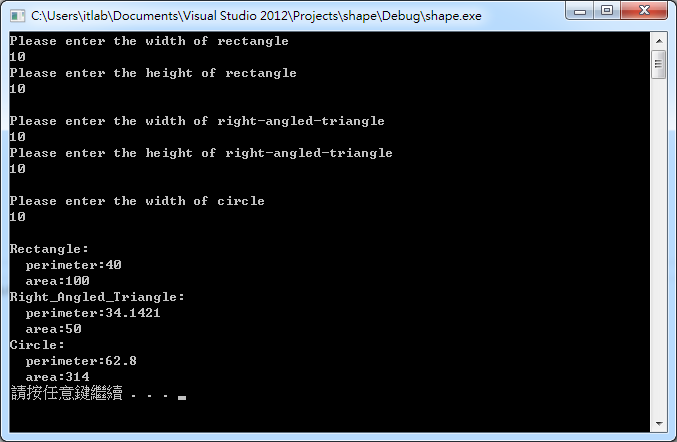
在完成這4個class之後，由於3種形狀都是繼承自base class，套用Polymorphism的概念，我們可以利用一個array來儲存，如此，只要搭配2個for loop就能完成輸入與顯示的工作了。

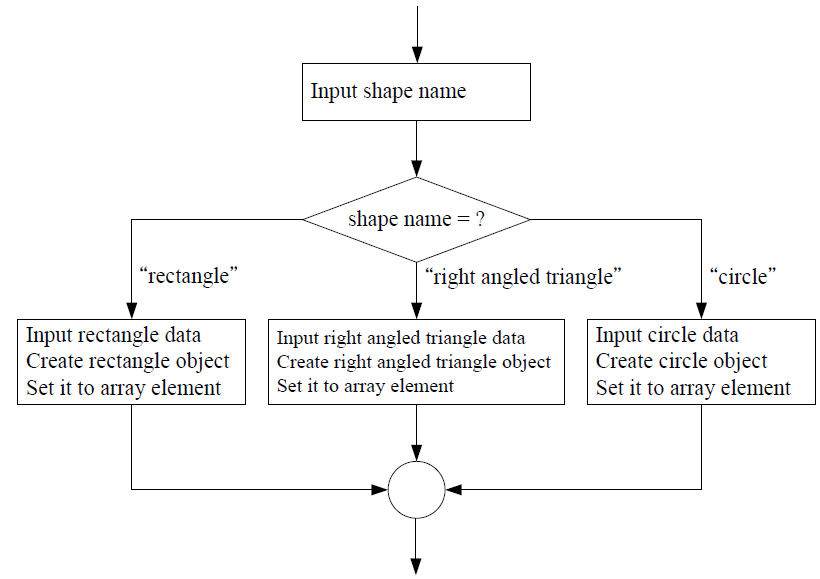
1. 程式規劃(programming)

這部分比較值得注意的是pure virtual的使用，藉由這個指令的幫助，在呼叫「計算面積」及「計算周長」2個function時，我們可以呼叫到 子class(方形、直角三角及圓)的function，而不會呼叫到base class的function，因為base class的function已經被宣告為pure virtual。

又由於「計算面積」及「計算周長」2個function在base class中已經有宣告過，所以之後利用Polymorphism的概念時，我們可以直接用array來指到「計算面積」及「計算周長」這2個function，加上先前提到，這種呼叫方式可以呼叫到 子class，所以我們就能直接得到將這三種形狀存於array中，並直接呼叫function來計算周長與面積。

5. 執行結果(results)





這部分，我大致上是按照上圖的流程，先依序輸入3種形狀的參數，並在輸入完時，產生這個形狀的物件，並加入array的element中。待3種形狀的物件都產生完之後，在計算他們的周長及面積，並顯示結果。

寬高及半徑都設為10，方便確認答案是否正確，圓周率設為3.14。

6. 結論(conclusion)

按照作業上的要求，並不需要使用到set與get，但我在作業完成之餘，嘗試了一下，發現雖然我可以呼叫base class中已經存在的「計算面積」及「計算周長」2個function，但set與get因為是子class中新增的function，所以Polymorphism的array無法直接呼叫，必須要直接使用產生的物件的function來呼叫，我的程式中，在產生物件時是使用new，直接存於array，如果要呼叫的話，就不能直接存進array，而是要先產生物件，利用物件來呼叫set與get，至於array的部分，可以用&rectangle之類的方式，將物件的位址存入array中。或是將set與get也用virtual寫到base class中，但這不可行，因為圓與其他形狀的set、get不同，所以只能用第一種方法--->先產生物件，再將位址給array，只是如此一來會多出額外的步驟。

附錄 程式碼(Appendix Code Listing)

#include <iostream>

#include <string>

using namespace std;

//-----------------class-----------------//

class shape

{

private:

string shape\_instance\_name;

public:

shape();

shape(string inst\_name);

void set\_name(string inst\_name){shape\_instance\_name=inst\_name;}

string get\_name(){return shape\_instance\_name;}

virtual double calc\_perimeter()=0; //pure virtual function

virtual double calc\_area()=0;

};

class rectangle : public shape

{

private:

double width,height;

public:

rectangle();

rectangle(string inst\_name,double w,double h);

void set\_width(double w){width=w;}

double get\_width(){return width;}

void set\_height(double h){height=h;}

double get\_height(){return height;}

double calc\_perimeter(){return (2\*(width+height));} //concrete function

double calc\_area(){return (width\*height);}

};

class right\_angled\_triangle : public shape

{

private:

double width,height;

public:

right\_angled\_triangle();

right\_angled\_triangle(string inst\_name,double w,double h);

void set\_width(double w){width=w;}

double get\_width(){return width;}

void set\_height(double h){height=h;}

double get\_height(){return height;}

double get\_hypotenuse(){return (sqrt((width\*width)+(height\*height)));}

double calc\_perimeter(){return (width+height+get\_hypotenuse());} //concrete function

double calc\_area(){return ((width\*height)\*0.5);}

};

class circle : public shape

{

private:

double radius;

public:

circle();

circle(string inst\_name,double r);

void set\_radius(double r){radius=r;}

double get\_radius(){return radius;}

double calc\_perimeter(){return (2\*3.14\*radius);} //concrete function

double calc\_area(){return (3.14\*radius\*radius);}

};

//-----------------constructor-------------//

shape::shape():shape\_instance\_name("un-declare")

{}

shape::shape(string inst\_name):shape\_instance\_name(inst\_name)

{}

rectangle::rectangle():shape(),width(0),height(0) //call the constructor that define in class:shape

{}

rectangle::rectangle(string inst\_name,double w,double h):shape(inst\_name),width(w),height(h)

{}

right\_angled\_triangle::right\_angled\_triangle():shape(),width(0),height(0)

{}

right\_angled\_triangle::right\_angled\_triangle(string inst\_name,double w,double h):shape(inst\_name),width(w),height(h)

{}

circle::circle():shape(),radius(0)

{}

circle::circle(string inst\_name,double r):shape(inst\_name),radius(r)

{}

//------------declare---------------//

shape \*shape\_array[2];

//----------------------------------//

int main()

{

double w,h,r;

for(int i=0;i<3;i++) //Enter the parameter in the order of rectangle,right-angled-triangle and circle

{

if(i==0)

{

cout<<"Please enter the width of rectangle"<<endl;

cin>>w;

cout<<"Please enter the height of rectangle"<<endl;

cin>>h;

cout<<endl;

shape\_array[i]=new rectangle("Rectangle",w,h);

}

else if(i==1)

{

cout<<"Please enter the width of right-angled-triangle"<<endl;

cin>>w;

cout<<"Please enter the height of right-angled-triangle"<<endl;

cin>>h;

cout<<endl;

shape\_array[i]=new right\_angled\_triangle("Right\_Angled\_Triangle",h,w);

}

else

{

cout<<"Please enter the width of circle"<<endl;

cin>>r;

cout<<endl;

shape\_array[i]=new circle("Circle",r);

}

}

for(int i=0;i<3;i++) //show the result

{

if(i<2)

{

cout<<shape\_array[i]->get\_name()<<":"<<endl

<<" perimeter:"

<<shape\_array[i]->calc\_perimeter()<<endl

<<" area:"

<<shape\_array[i]->calc\_area()<<endl;

}

else //circle doesn't have width and height,but it have radius

{

cout<<shape\_array[i]->get\_name()<<":"<<endl

<<" perimeter:"

<<shape\_array[i]->calc\_perimeter()<<endl

<<" area:"

<<shape\_array[i]->calc\_area()<<endl;

}

}

system("pause");

return 0;

}