บทที่ 12

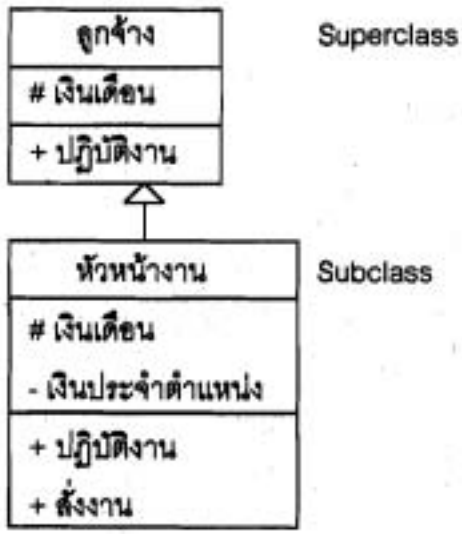
Inheritance

วัตถุประสงค์

1. เพื่อให้นักศึกษาทราบถึงการถ่ายทอดคุณลักษณะจาก base class ไปยัง subclass
2. เพื่อให้นักศึกษาเข้าใจรูปแบบการกำหนดคลาสที่สามารถถ่ายทอดคุณลักษณะไปยังคลาสอื่นๆได้
3. เพื่อให้นักศึกษาเข้าใจถึงกระบวนการถ่ายทอดคุณลักษณะของคลาส
4. เพื่อให้นักศึกษาสามารถประยุกต์เพื่อใช้ออกแบบในการแก้ปัญหาโปรแกรมได้

Inheritance คือการถ่ายทอดคุณสมบัติจากคลาสที่มีอยู่แล้วไปยังคลาสใหม่ ซึ่งทำให้เกิดข้อดีในหลายๆด้าน กล่าวคือทำให้เราสามารถสร้างคลาสใหม่ๆจากการปรับเปลี่ยนคลาสที่มีอยู่ทำให้ลดเวลาและค่าใช้จ่ายในการพัฒนาระบบ เป็นคุณลักษณะที่ดีประการหนึ่งของการออกแบบโปรแกรมที่เรียกว่า Reusable

สำหรับในบทนี้จะกล่าวถึง Generalization Abstraction เป็นกระบวนการพิจารณาคลาสที่มีลักษณะที่คล้ายคลึงกันหรือมีคุณลักษณะอย่างใดอย่างหนึ่งเหมือนกัน นำออกมาจัดเป็นหมวดหมู่เป็นคลาสที่เป็นสามัญ(General) และเมื่อเพิ่มคุณลักษณะพิเศษ(Specialization) ให้กับคลาสสามัญ จะทำให้ได้คลาสใหม่ ที่เราเรียกว่า คลาสย่อย(Subclass) โดยถ่ายทอดคุณลักษณะของคลาสสามัญมายังคลาสย่อยนั่นเองเราเรียกการที่ คลาสย่อยรับคุณสมบัติทุกอย่างมาจาก Superclass ว่า Inheritance สัญลักษณ์ ที่ใช้คือลูกศรหัวรูปสามเหลี่ยมชี้จาก Subclass ไปยัง Superclass



ดังนั้นการกำหนดคลาสย่อย(Subclass) ที่ได้รับถ่ายทอดคุณลักษณะของคลาสสามัญ(Superclass) จึงมีรูปแบบการกำหนดที่เปลี่ยนไปดังรูปแบบดังนี้

Defining a Derived Class
Form : class *derived* : access base { ... }; Example : class circle : public figure { ... };

คำอธิบาย

1. class derived ในที่นี้คือคลาสที่เป็นคลาสย่อย(sub class) ที่ได้รับถ่ายทอดคุณลักษณะมาจากคลาสสามัญ(super class หรือ class base)
2. สำหรับ access specifier อาจเป็น public หรือ private หรือ protected ก็ได้
   - ถ้าเป็น public จะส่งผลให้ public หรือ protected ของ members ของ base class เป็น public หรือ protected ใน derived class.
   - ถ้าเป็น private หรือ protected จะส่งผลให้ public และ protected ของ members ของ base class กลายเป็น private หรือ protected ของ derived class
3. คำว่า protected จะแสดงถึงระดับของการป้องกันระหว่าง public และ private
4. การเรียกใช้ items ที่กำหนดหลังจากการกำหนดคำเฉพาะ public สามารถเข้าถึงหรือเรียกใช้ได้ในทุกๆฟังก์ชัน
5. การเรียกใช้ items หลังจากการกำหนดคำเฉพาะ private จะสามารถเรียกใช้ได้เฉพาะ Member function ของคลาสนั้นเท่านั้น
6. สำหรับ items ที่กำหนดหลังคำเฉพาะ protected สามารถเข้าถึงได้ทั้งสองคลาสทั้งคลาสที่กำหนดและคลาสที่รับการถ่ายทอดคุณลักษณะไป.

ข้อสังเกตุ

1. constuctors ไม่เป็น inherited คือไม่สามารถถ่ายทอดคุณลักษณะไปยังคลาสอื่นๆได้

2. การประกาศออบเจ็กต์ ที่เป็น derived class นั้นจะส่งผลให้มีการปฏิบัติงานของ base class constructor ก่อน derived class constructor.

เพื่อให้เข้าใจได้ดียิ่งขึ้น มาพิจารณาตัวอย่างที่ 12.1 ซึ่งเป็นการสร้าง Subclass หรือ base class ชื่อ indexListBase โดยถ่ายทอดคุณลักษณะไปยังคลาสที่ชื่อว่า indexListSpec

ตัวอย่างที่ 12.1 Class indexListSpec derived from indexListBase parent class

สำหรับแฟ้ม indexListBase.h เป็นต้นแบบของ base class ที่จะถ่ายทอดคุณสมบัติบางประการไปยัง indexListSpec.h มีการระบุรายละเอียดดังต่อไปนี้

```
// File : indexListBase.h
// Index list base class

# ifndef INDEX_LIST_BASE_H
# define INDEX_LIST_BASE_H

template <class T, int maxSize>
class indexListBase
{
public:
        // Constuctor
        indexListBase ( );

        // Add an element of the end of an indexed lise
        bool append( const T& );       // IN : element appended
        // Insert an element at a specified index.
        bool insert (int,                  // IN: Inserttion index
                const T& );                // IN: element inserted
```

```
        // Retrieve an element at a specified index.
        bool retrieve (int,             // IN:  index of element  to retrieve
                       T&)  const;      // OUT: value retrieved

        // Remove  (delete) an element at a specified index.
        bool remove (int);              // IN:  index of element to  be remove

        // Get the current number of data element in the list.
        int getSize (  )  const;

protected:
        T elements [maxSize];           // Storage for data
        int size;                       // Count of number of  elements  in list
};
#endif   // INDEX_LIST_BASE_H
```

สำหรับคลาสนี้เป็นคลาสต้นแบบที่สามารถถ่ายทอดคุณลักษณะไปยังคลาสอื่นๆได้ เป็น Template class ที่ประกอบด้วย member function หลายฟังก์ชัน ได้แก่ append , insert , retrieve , remove และ getSize ซึ่งเป็นฟังก์ชันที่สามารถเรียกใช้จากฟังก์ชันอื่นๆได้ สำหรับตัวแปร elements และ size ได้ถูกกำหนดให้เป็น protected ซึ่งเป็นคุณลักษณะที่สามารถถ่ายทอดไปให้กับ subclass ตัวอื่นๆได้ ดังนี้

```
// File: indexListSpec.h
// Index list class specialized from index list parent
//       includes sum and average attributes and functions
//       Definition and Implementation
```

```
# ifndef   INDEX_LIST_SPEC_H
# define  INDEX_LIST_SPEC_H

template < class T, int maxSize >
class indexListSpec : public indexListBase
{
public:
        // Public member functions inherited form parent class:
        //     append , insert , retrieve , remove , getSize

        // Constructor
        .indexListSpec ( ) ;

        // compute sum of all dadata in the list
        T computeSum ( ) ;

        // Compute average of all data in the list
        T computeAverage ( ) ;

protected:
        // Protected data  inherited from parent class:
        //       list of data elements and the count of elements in list
        T sum;          // sum of the data element in the list
        T ave;          // average of the data elements in the list
} ;
#endif   // INDEX_LIST_SPEC_H
```

สำหรับ subclass นี้มีการถ่ายทอดคุณลักษณะทั้งหมดจาก base class โดย Member function ทั้งหมดที่เป็น public ก็จะถูกถ่ายทอดมาเป็น public ของ subclass นี้ด้วย นอกจากนี้ subclass นี้ยังได้กำหนดคุณลักษณะพิเศษอื่นๆเพิ่มขึ้นไปอีกไม่ว่าเป็นฟังก์ชัน computeSum และ computeAverage รวมทั้งเพิ่มคุณลักษณะของ sum และ ave เข้าไป ซึ่งสามารถถ่ายทอดคุณลักษณะต่างๆเหล่านี้ไปให้แก่ subclass อื่นๆได้อีก

กรณีศึกษา: การหาพื้นที่และความยาวของรูปต่าง ๆ

ปัญหา เราต้องการที่จะหาพื้นที่และความยาวของรูปต่างๆที่มีทรวดทรงแตกต่างกัน อาทิเช่น รูปวงกลม รูปสี่เหลี่ยมผืนผ้า รูปสี่เหลี่ยมด้านเท่า โดยผู้ใช้สามารถป้อนข้อมูลของคุณลักษณะที่จำเป็นของแต่ละรูปให้แก่โปรแกรม โปรแกรมสามารถประมวลผลคำนวณผลลัพธ์ และพิมพ์คุณลักษณะต่างๆของรูปออกมาอย่างถูกต้องทางจอภาพ

วิเคราะห์

เราต้องทำการวิเคราะห์ว่ารูปทั้งสามรูปอันได้แก่ รูปวงกลม รูปสี่เหลี่ยมผืนผ้า และรูปสี่เหลี่ยมด้านเท่า มีคุณลักษณะ และ member function อะไรบ้าง จะเห็นได้ว่ารูปทั้งสามรูปนี้มีกิจกรรมในการทำงานเหมือนๆกัน 4 ฟังก์ชัน ได้แก่

1. อ่านข้อมูล (readFigure)
2. คำนวณหาพื้นที่ (computeArea)
3. คำนวณความยาวของรูป (computePerim)
4. พิมพ์คุณลักษณะของรูป (displayFig)

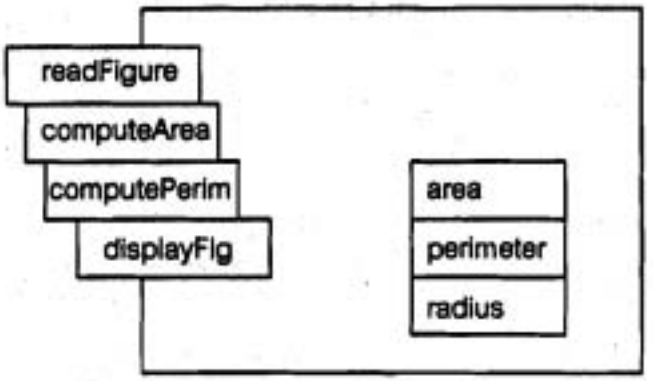
สำหรับคุณลักษณะ (attribute) ที่เหมือนๆกันคือ

1. พื้นที่ (area)
2. ความยาวของรูป (perimeter)

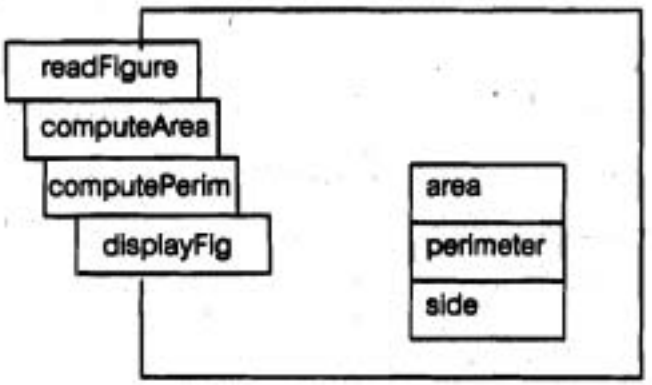
สำหรับคุณลักษณะพิเศษที่แตกต่างกันของรูปต่างๆ ได้แก่

1. ความยาวรัศมี(radius) ซึ่งเป็นคุณลักษณะพิเศษของรูปวงกลม
2. ความยาวด้าน (side) เป็นคุณลักษณะพิเศษของรูปสี่เหลี่ยมด้านเท่า
3. สำหรับคุณลักษณะพิเศษของสี่เหลี่ยมผืนผ้าประกอบด้วย
   - ความกว้าง (width) และ ส่วนสูง (height)

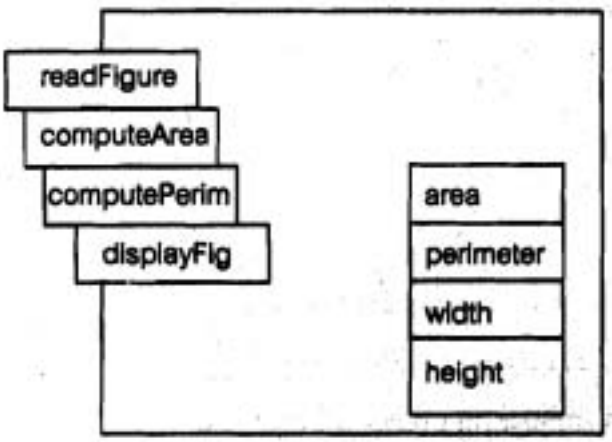
ในที่นี้เราสามารถสร้างคลาสขึ้นมาทั้งหมด 3 คลาสให้ชื่อว่า circle , square และ rectangle ตามรายละเอียดที่ระบุดังภาพต่อไปนี้



circle

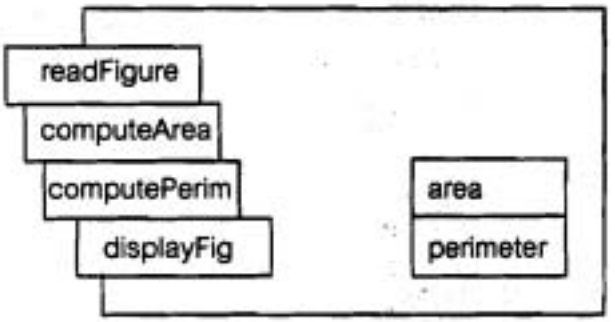


square



rectangle

จะเห็นได้ว่าคุณลักษณะของรูปทั้งสามมีส่วนที่เหมือนกันดังนั้นเราสามารถสร้าง base class หรือ superclass เพื่อเป็นคลาสต้นแบบและให้มีการถ่ายทอดคุณลักษณะไปยังคลาสทั้งสามนี้ได้ดังนี้



figure

เรานำคุณลักษณะและฟังก์ชันการทำงานที่มีกิจกรรมเหมือนกันของทั้งสามคลาสมารวมไว้ในคลาส figure ปัญหาที่ตามมาคือกิจกรรมต่างๆเมื่อถูกถ่ายทอดไปยัง subclass ต่างๆจะมีการประมวลผลแตกต่างกันขึ้นอยู่กับรูปร่าง ข้อมูลที่ต้องการก็แตกต่างกัน ภาษา C++ ได้เตรียมการแก้ปัญหานี้โดยให้ผู้เขียนโปรแกรมระบุคำเฉพาะ virtual ไว้ก่อนหน้าฟังก์ชันต่างๆภายในคลาส เพื่อเป็นการแสดงให้ทราบว่าฟังก์ชันต่างๆเหล่านี้เป็นฟังก์ชันพิเศษเป็นฟังก์ชันเสมือน (virtual function) ที่มีการทำงานแตกต่างกันเมื่อถูกปฏิบัติงานใน subclass โดยทั่วไป ฟังก์ชันเสมือนจะถูกเรียกใช้ในขณะที่ปฏิบัติงาน(execute) เท่านั้น โดยระบุจะตัดสินใจหรือวินิจฉัยฟังก์ชันที่จะทำงานจริงๆจาก subclass ที่มีอยู่

รูปแบบการระบุฟังก์ชันพิเศษ

```
virtual  prototype = 0 ;
```

โดยการระบุนี้จะไม่มีการกำหนดคำสั่งในการทำงานแต่อย่างใด การทำงานจะถูกกำหนดในแต่ละ subclass ที่เกี่ยวข้อง เป็นการกำหนดฟังก์ชันเสมือนเพียวๆ (pure virtul function) เรามาดูการกำหนด superclass ที่ชื่อ figure กันดีกว่า

Definition of the class figure

```
// File : figure.h
// abstract class figure contains operators and attributes common
//        to all figures.
#ifndef FIGURE_H
#define FIGURE_H
class figure
{
        // Member functions (common to all figures) ...
        public:
                // Read figure attributes
                virtual void readFigure( ) = 0 ;

                // Compute the area
                virtual void computeArea( ) = 0 ;

                // Compute the perimeter
                virtual void computePerim( ) = 0 ;

                // Compute information about the figure
                virtual void displayFig( );
        // Attributes common to all figures ...
        protected :
                float area;             // The area of the figure
                float perimeter;        // The perimeter of the figure
};
# endif         // FIGURE_H
```

สำหรับ class figure นี้เราถือว่าเป็น abstract base class ซึ่งไม่มีการทำงานแต่อย่างไร เพียงแต่เป็นคลาสต้นแบบสำหรับถ่ายทอดคุณลักษณะอย่างเดียว โดยบรรยายรายละเอียดของคลาสในแง่ของคุณลักษณะและฟังก์ชันต่างๆที่สามารถกระทำได้เท่านั้น

คราวนี้เรามาดูการกำหนด subclass ต่างๆกันบ้างเริ่มจาก class circle

```
// File : circle.h
# ifndef  CIRCLE_H
# define  CIRCLE_H

# include  "figure.h"

// the class circle
class circle : public figure      // a circle is a figure
{
        // Overriding member functions (unique for circle) ...
        public :
                // Read  a  circle
                void  readFigure ( ) ;

                // Compute the area of a rectangle
                void  computeArea ( ) ;

                // Compute the perimeter of a rectangle
                void computePerim ( ) ;

                // Display characteristics unique to rectangles
                void displayFig ( ) ;
```

```
        // Data members ( unique to rectangles ) ...
        private :
                float  radius;
};
# endif          // CIRCLE_H
```

การกำหนดคลาส circle นี้จะมีการดึงคุณลักษณะจาก คลาส figure ในที่นี้คือดึง area และ perimeter มาด้วย ในทำนองเดียวกันเราสามารถสร้างต้นแบบของคลาสรองสี่เหลี่ยมด้านเท่าได้ดังนี้

The class square

```
// File : square.h
# ifndef  SQUARE_H
# define  SQUARE_H

# include  "figure.h"

// The class square
class square : public figure          // square is  a  figure
{
        // Overriding member functions (unique for square) ...
        public :
                // Read a square
                void  readFigure ( );

                // Compute are of e square
                void  computeArea ( );
```

```
        // Compute perimeter of a square
        void computePerim ( );

        // Display characteristics unique to squares
        void displayFig ( ) ;

    // Attributes (unique to squares) ...
    private :
        float side;                    // length of side of square
};
# endif
// SQUARE_H
```

การกำหนดคลาส square นี้จะมีการดึงคุณลักษณะจาก คลาส figure ในที่นี้คือดึง area และ perimeter มาด้วย ในทำนองเดียวกันเราสามารถสร้างต้นแบบของคลาสของสี่เหลี่ยมผืนผ้าได้ดังนี้

The class rectangle

```
// File : reatangle.h
#ifndef  RECTANGLE_H
#define  RECTANGLE_H
#include  "figure.h"

//  the class rectangle
class restangle : public figure     // a rectangle is a figure
{
    // Overriding member functions ( unique for rectangle) ...
```

```
    public :

        // Read a rectangle
        void  readFigure ( );

        // Compute the area of a rectangle
        void computeArea ( );

        // Compute the perimeter of a rectangle
        void computePerim ( );

        // Display characteristics unique to rectangles
        void displayFig ( ) ;

    // Data member ( unique to rectangle ) ...
    private :

        // width of rectangle
        float  width;

        // height of rectangle
        float  height;

};
# endif
// RECTANGLE_H
```

ออกแบบ

เมื่อเราวิเคราห์คลาสต่างๆว่ามีรายละเอียดเป็นอย่างไรได้แล้ว เราสามารถนำสิ่งที่เกี่ยวข้องกันหรือการปฏิบัติการต่างๆมาออกแบบเป็นโปรแกรมเพื่อสามารถแก้ปัญหาตามที่ต้องการได้ โดยมีขั้นตอนการทำงานดังต่อไปนี้

INITIAL ALGORITHM

1. กำหนดรูปร่างของรูป
2. อ่านคุณลักษณะของรูปร่าง (readFigure)
3. คำนวณหาพื้นที่ของรูปร่าง (computeArea)
4. คำนวณความยาวรอบรูป (computePerimeter)
5. แสดงข้อมูลของรูปร่างทางจอภาพ (displayFig)

เขียนโปรแกรม

Figure program main function

```
// File: figuresTest.cpp
// Main program to illustrate the figures class
#include <iostream>
#include <cstddef>
using namespace std;

#include "circle.h"
#include "square.h"
#include "rectangle.h"
#include "figure.h"                    .....1

    // Functions called ...
    // Get the type of figure
    figure* getFigure( ) ;             .....2
```

```
    // Process one figure
    void processFigure
        (figure&) ;              // INPUT:  figure to be processed     .....3

int main()
{
    figure* myFig;               // a pointer to a figure              .....4

    // process a selected figure until no more figures selected
    for ( myFig = getFigure( ) ; myFig ! = 0;  myFig =  getFigure( ) )  .....5
    {
        processFigure(*myFig) ;                                         .....6
        delete myFig;          //delete this figure                     .....7
    }

    return 0 ;
}

// Process one figure
void processFigure
   (figure& fig)               // INOUT: The figure to be  process
{
    fig.readFigure() ;         // get characteristict of figure
    fig.computeArea() ;        // compute its area
    fig.computePerim() ;       // compute its perimeter
    fig.displayFig() ;         // display characteristics
}
```

```
// File: figuresTest.cpp
// Reads the kind of figure
// Pre : none
// Post : returns a pointer to the type of figure
figure* getFigure ( )
{ // local data
   char figChar;          // character indicating figure type
   do
   {
         cout  << " Enter the kind of object " << endl ;
         cout  << " Enter C (Circle) , R (Rectangle) ,or S (Square)"
                 << endl;
         cout  << "Enter X to exit program" << endl;
         cin   >>  figChar;
         switch ( figChar)
         {
               case  ' C ' : case ' C ':
                  return new circle ;
               case ' R' : case ' r' :
                  return new rectangle ;
               case 'S' : case 's' :
                  return new square;
               case 'X' : case 'x' :
                  return 0 ;
         } // end switch
   } while ( true ) ;
} // end getFigure
```

คำอธิบาย

1. เป็นการนำคลาสต่างๆที่ได้ออกแบบไว้นำมารวมไว้ในโปรแกรมเพื่อให้สามารถเรียกใช้ฟังก์ชันต่างๆที่อยู่ในคลาสที่กำหนดได้
2. เป็นต้นแบบของฟังก์ชันที่ทำหน้าที่ในการรับรูปร่างที่ต้องการทำงาน
3. เป็นต้นแบบของฟังก์ชันในการประมวลผลรูปร่างโดยมีการส่งรูปร่างที่ต้องการประมวลผลมาให้แก่ฟังก์ชัน
4. เป็นการประกาศตัวแปรพอยเตอร์ให้เก็บตำแหน่งที่อยู่ของรูปร่าง(figure)
5. เป็นการทำงานวนรอบ โดยตัวแปร myFig จะมีค่าจากผลการทำงานของฟังก์ชัน getFigure โดยแต่ละรอบจะมีค่าที่เกิดจากการทำงานของฟังก์ชัน getFigure โดยการทำงานในฟังก์ชันนี้มีเมนูให้ผู้ใช้เลือกการทำงาน โดยมีเงื่อนไขในการทำงาน 4 ทางเลือกดังนี้
   - ถ้าผู้ใช้เลือกรูปวงกลมหรือกดอักขระ 'C' การทำงานจะสร้างออบเจ็กต์ใหม่ที่อยู่ในคลาส circle โดยให้ตัวแปรพอยเตอร์ myFig ชี้อยู่ที่ออบเจ็กต์นี้
   - ถ้าผู้ใช้เลือกรูปสี่เหลี่ยมผืนผ้าหรือกดอักขระ 'R' โปรแกรมสร้างออบเจ็กต์ใหม่ที่อยู่ในคลาส rectangle โดยให้ตัวแปรพอยเตอร์ myFig ชี้อยู่ที่ตำแหน่งที่อยู่นี้
   - ถ้าผู้ใช้เลือกรูปสี่เหลี่ยม หรือกดอักขระ 'S' โปรแกรมจะสร้างออบเจ็กต์ใหม่ที่อยู่ในคลาส square โดยให้ตัวแปรพอยเตอร์ myfig ชี้อยู่ที่ตำแหน่งนี้
   - ถ้าผู้ใช้เลือกกดอักขระ 'X' โปรแกรมจะส่งค่า 0 กลับไปให้แก่ตัวแปร myFig

   เมื่อตัวแปรพอยเตอร์ myFig มีค่าแล้วจะไปทำงานในคำสั่งที่ 6 และ 7 ต่อไป ต่อจากนั้นจะมีการเรียกฟังก์ชัน getFigure ใหม่อีกครั้งเพื่อทำงานในรอบต่อไป การทำงานวนรอบนี้จะหยุดก็ต่อเมื่อผู้ใช้เลือกกดอักขระ 'X' นั่นเอง
6. เป็นคำสั่งในการเรียกฟังก์ชัน processFigure ทำงาน โดยมีการส่งค่าตำแหน่งที่อยู่ที่ตัวแปรพอยเตอร์ myFig ชี้อยู่ไปให้กับฟังก์ชัน สมมุติว่าตัวแปรพอยเตอร์ myFig ชี้ที่ออบเจ็กต์ที่เป็นสมาชิกของคลาส circle ตำแหน่งที่อยู่ของออบเจ็กต์นี้จะถูกส่งให้แก่ formal parameter ที่ชื่อว่า fig ภายในฟังก์ชัน ดังนั้นค่าที่เก็บในตัวแปร fig มีค่าเท่ากับตำแหน่งที่อยู่ของออบเจ็กต์ที่เป็นสมาชิกของคลาส circle ที่เราอ้างอิงนั่นเอง ส่งผลให้การทำงานในฟังก์ชันมีการเรียกใช้ member function ของคลาส circle ทำงาน เนื่องจากตัวแปรพอยเตอร์ที่กำหนดขึ้นชี้ไปที่ออบเจ็กต์ที่เป็นสมาชิกของคลาส circle นั่นเอง ดังนั้นถ้าตัวแปรพอยเตอร์ชี้ไปที่ออบเจ็กต์ที่

เป็นสมาชิกของคลาส square จะเป็นการเรียกใช้ Member function ของคลาส square ทำงานเช่นกัน

7. เป็นคำสั่งในการลบเนื้อที่ที่ได้จัดสรรขึ้นที่ตัวแปร myFig ชี้อยู่

สำหรับ Member function ของคลาส figure สามารถเขียนคำสั่งภาษา C++ ได้ดังนี้

Implementation of the class figure

```
// File :  figure.cpp
// Implementation of the base class figure

# include " figure .h"
# include < iostream>
using namespace std;

// Functions readFigure , computeArea , computePerim are pure  virtual
// Displays the common characteristics of a figure
// Pre :   none
// Post : common characteristics are displayed
void figure : : displayFig  ( )
{
    cout << " Area  is "  << area << endl ;
    cout << " Perimeter is "  << perimeter << endl ;
}  // end displayFig( )
```

สำหรับ Member function ของคลาส figure นี้ถือว่าจะถูกใช้งานในทุกๆคลาส เนื่องจากมีส่วนของการปฏิบัติงานที่ต้องกระทำเหมือนๆกัน และมีการอ้างอิงถึง attribute ที่มีการถ่ายทอดคุณลักษณะด้วยในที่นี้คือ การพิมพ์ค่าของตัวแปร area และ perimeter ซึ่งเป็นคุณลักษณะที่ subclass ทุกๆ subclass ต้องอ้างอิงถึง ในที่นี้ ไม่ว่าจะเป็น class circle , class square และ calss rectangle ต้องมีการเรียกใช้ ดังรายละเอียดต่อไปนี้

สำหรับ Member function ของคลาส circle สามารถเขียนคำสั่งภาษา C++ ได้ดังนี้

Implementation of the class circle

```
// File : circle .cpp
// Implementation of the class circle
# include < iostream >
# include "circle.h"

const float pi = 3.1415927;

// Read data unique to a circle
void circle : : readFigure ( )
{
        cout << " Enter radius > " ;
        cin  >> radius ;
}

// Compute the perimeter ( circumference ) of a circle
void circle : : computePerim ( )
{
        perimeter = 2.0 * pi * radius ;
}

// Compute the area of a circle
void circle : : computeArea ( )
{
        area = pi * radius * radius ;
}
```

```
// Display the characteristics of a circle
void circle : : displayFig ( )
{
        // Display the type of figure and its radius.
        cout << " Figure Shape is Circle " << endl ;
        cout << " Radius is " << radius << endl ;
        // Call the base function to display common characteristics.
        figure : : displayFig ( ) ;
}
```

เห็นได้ว่า Member function ของการอ่านข้อมูล (readFigure) การคำนวณหาพื้นที่ (computeArea) หรือแม้กระทั่งการหาความยาวของเส้นรอบวงกลม(computePerim)นี้ ไม่มีส่วนใดๆ ที่เกี่ยวข้องกับคลาส figure เลย แต่สำหรับการแสดงผลของรูปร่าง(displayFig) มีส่วนที่เกี่ยวข้องโดยมีการเรียกใช้ฟังก์ชันของ figure ทำงานหลังจากที่แสดงผลการทำงานซึ่งถือว่าเป็นคุณลักษณะพิเศษของคลาสนี้แล้ว จากคำสั่ง figure::displayFig()

ในทำนองเดียวกัน สำหรับ Member function ของคลาส square เราสามารถสร้างได้ดังนี้

Implementation of the class square

```
// File : square .cpp
// Implementation of the class squre
# include < iostream >
# include " square.h"
// Read data unique to a square
void square : : readFigure ( )
{
        cout << " Enter side > ";
        cin >> side ;
}
```

```
// Compute the perimeter of a square
void square : : computePerim ( )
{
        perimeter  = 4.0 * side ;
}

// Compute the area of a square
void square : : computeArea ( )
{
        area = side * side ;
}

// Display the characteristics of a square
void square : : displayFig ( )
{
        // Display the type of figure and its size.
        cout  <<  " Figure shape is Square "  << endl ;
        cout  <<  " Side is "  <<  side  << endl ;
        // Call the base function to display common characteristics.
        figure : : displayFig ( ) ;
}
```

สำหรับ Member function ของคลาส rectangle **สามารถเขียนคำสั่งภาษา** C++ **ได้ดังนี้**

implementation of the class rectangle

```
// File : rectangle.cpp
// implementation of the rectangle class
# include  < iostream >
```

```
using namespace std;
# include " rectangle.h"
// Read data unique to a rectangle
void rectangle : : readFigure ( )
{
        cout << " Enter width : " ;
        cin >> width ;
        cout << " Enter height : " ;
        cin >> height ;
}
// Compute the perimeter of a rectangle
void rectangle: :computePerim( )
{
    perimeter = 2.0 * (width + height) ;
}

// compute the area of a rectangle
void rectangle: :computeArea( )
{
    area = width * height;
}

// Display the characteristics of a rectangle
void rectangle: :displayFig( )
{
    // Display the type of figure and its height and width.
    cout << "Figure sharp is Rectangle" << endl;
```

```
    cout <<  "Height is" << height << endl;
    cout <<  "Width is" << width << endl;
     // call the base function to display common characteristics.
    figure: :displayFig( );
}
```

เห็นได้ว่า class square และclass rectangle จะมีฟังก์ชันในการอ่านข้อมูลรูปร่าง (readFigure) การคำนวณหาพื้นที่(computeArea) หรือแม้กระทั่งการหาความยาวของเส้นรอบวงกลม(computePerim)นี้ แตกต่างกัน แต่สำหรับการแสดงผลรองรูปร่าง(displayFig) หลังจากที่มีการแสดงคุณสมบัติพิเศษของแต่ละคลาสแล้ว ต้องมีการเรียกใช้ฟังก์ชันของ figure::displayFig() ซึ่งเป็นฟังก์ชันการทำงานที่เป็น super class เพื่อพิมพ์ค่าของพื้นที่และความยาวของเส้นรอบวงออกมาทางจอภาพ

สำหรับในตัวอย่างนี้ถ้าเราออกแบบโดยใช้ตัวแปรพอยเตอร์ให้มีการชี้หรือเก็บตำแหน่งที่อยู่ของออบเจ็กต์ที่สามารถมีรูปร่างได้แตกต่างกันถึง 3 รูปร่าง การเขียนโปรแกรมหรือโครงสร้างโปรแกรมนี้เราสามารถเขียนได้อีกในหลายๆลักษณะ สมมุติกำหนดให้ ตัวแปร figShape มีชนิดข้อมูลเป็น char มีการปรับเปลี่ยนคำสั่งภาษา C++ ของฟังก์ชันหลัก(main function) โดยปรับปรุงเป็นดังนี้

```
switch ( figShape)
{
      case  'c'  : case 'C'  :
            circle.readFigure ( );
            circle.computArea ( );
            circle.computPerim ( );
            circle.displayFig ( ); break;
          case 'r'   : case 'R' :
             rectangle.readFigure ( );
             rectangle.computeArea ( );
             rectangle.computePerim ( );
```

```
            rectangle.displayFig ( );  break;
        case 's'   : case 'S'  :
            square.readFigure ( );
            square.computeArea ( );
            square.computePerim ( );
            square.displayFig ( );   break;
        default :
            cerr << "Incorrect character for shapes
                    designation."
                 << "Re-run program." << endl;
}   // end  switch
```

การเขียนคำสั่งในลักษณะนี้จะทำให้เข้าใจได้ดียิ่งขึ้นเพราะเห็นการทำงานได้ชัดเจน การทำงานมีการอ้างถึงชื่อคลาสและMember function ของคลาสโดยตรง ส่งผลให้การทำงานจะมีการเรียกใช้ Member function ที่แตกต่างกันขึ้นอยู่กับรูปร่างที่ต้องการทำงาน

สรุป

1. Inheritance คือการถ่ายทอดคุณสมบัติจากคลาสที่มีอยู่แล้วไปยังคลาสใหม่ ทำให้สามารถสร้างคลาสใหม่ๆจากการปรับเปลี่ยนคลาสที่มีอยู่ทำให้ลดเวลาและค่าใช้จ่ายในภวรพัฒนาระบบ เป็นคุณลักษณะที่ดีประการหนึ่งของการออกแบบโปรแกรมที่เรียกว่า Reusable
2. การกำหนดคลาสย่อย(Subclass) ที่ได้รับถ่ายทอดคุณลักษณะของคลาสสามัญ(Superclass) มีรูปแบบดังนี้

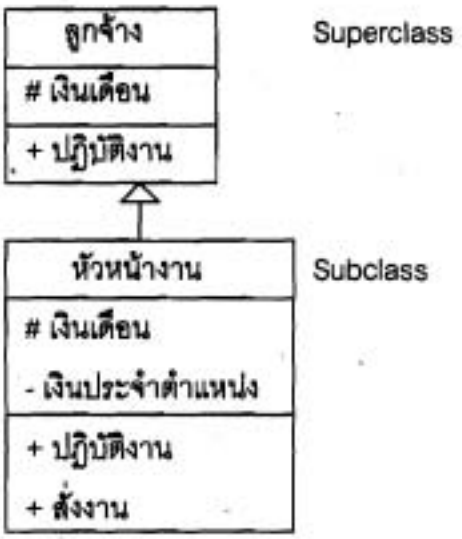
   Form : class *derived* : access base

   { ... };
3. class derived ในที่นี้คือคลาสที่เป็นคลาสย่อย(sub class) ที่ได้รับถ่ายทอดคุณลักษณะมาจากคลาสสามัญ(super class หรือ class base)
4. access specifier อาจเป็น public หรือ private หรือ protected ก็ได้
   - ถ้าเป็น public จะส่งผลให้ public หรือ protected ของ members ของ base class เป็น public หรือ protected ใน derived class.
   - ถ้าเป็น private หรือ protected จะส่งผลให้ public และ protected ของ members ของ base class กลายเป็น private หรือ protected ของ derived class
   - protected จะแสดงถึงระดับของการป้องกันระหว่าง public และ private
5. การเรียกใช้ items ที่กำหนดหลังจากการกำหนดคำเฉพาะ public สามารถเข้าถึงหรือเรียกใช้ได้ในทุกๆฟังก์ชัน
6. การเรียกใช้ items หลังจากการกำหนดคำเฉพาะ private จะสามารถเรียกใช้ได้เฉพาะ Member function ของคลาสนั้นเท่านั้น
7. สำหรับ items ที่กำหนดหลังคำเฉพาะ protected สามารถเข้าถึงได้ทั้งสองคลาสทั้งคลาสที่กำหนดและคลาสที่รับการถ่ายทอดคุณลักษณะไป.
8. constuctors ไม่เป็น inherited คือไม่สามารถถ่ายทอดคุณลักษณะไปยังคลาสอื่นๆได้ การประกาศออบเจ็กต์ ที่เป็น derived class นั้นจะส่งผลให้มีการปฏิบัติงานของ base class constructor ก่อน derived class constructor.

แบบฝึกหัด

1. จงสร้างคลาสจากแผนภาพความสัมพันธ์ต่อไปนี้ โดยออกแบบตามความเหมาะสม

2. พิจารณาแผนภาพต่อไปนี้ จงออกแบบคลาสตามรายละเอียดที่กำหนดให้



3. จงสร้าง class triangle เป็น subclass ของ figure ซึ่งเป็นการหาค่าพื้นที่และความยาวของเส้นรอบรูปของสามเหลี่ยม ซึ่งมี attribute ประกอบด้วยความยาวด้านทั้งสามของสามเหลี่ยมสำหรับ member function จะสามารถกระทำได้เหมือนกับ class circle และคลาสอื่นๆที่เป็นคลาสย่อยของคลาส figure