仮想関数とグラフィック処理

以下に示すプログラムにて、GrWin を使ったグラフィックス機能を使い、
仮想関数の便利さを説明するための簡単な例を示す。

以下のプログラムに自分なりの改良を加え、
仮想関数の便利さが分かるようなプログラムを作る。

```
#include <stdio.h>
#include <math.h>
#include <GrWin.h>

// GrWin の呼び出しをコンストラクタとデストラクタ経由で
// 簡単に使えるようにするだけのクラス。
class GrWin {
private:
        int width , height ;    // ウィンドウの幅と高さ
public:
        GrWin( int , int ) ;    // ウィンドウを開いて初期化
        ~GrWin() ;              // ウィンドウを閉じる
} ;

// ウィンドウを生成し、座標系や表示範囲の初期化
GrWin::GrWin( int w , int h )
:       width( w ) ,
        height( h )
{       // 左上が原点の座標系とする。
        GWopen( 0 ) ;
        GWsize( -5 , &width , &height ) ;
        GWsize( -3 , NULL , NULL ) ;
        GWvport( 0.0 , 0.0 , (float)width / (float)height , 1.0 ) ;
        GWindow( 0.0 , (float)height - 1.0 , (float)width - 1.0 , 0.0 ) ;

        GWclear( GWC_WHITE ) ;
        GWsetpen( GWC_BLACK , GWL_SOLID , 1 , GWX_COPYPEN ) ;
}

// ウィンドウを閉じる。
GrWin::~GrWin() {
        GWquit() ;
}
```

```
// 図形の仮想基底クラス
//      描画の draw()メソッドを派生クラスでは記述すること。
class Figure {
public:
        // draw() メソッドは、図形を表示する基点の座標を指定。
        virtual void draw( int , int ) = 0 ;
} ;

class FigureBox : public Figure {
private:
        int     width ;         // 四角形の幅
        int     height ;        // 四角形の高さ
public:
        FigureBox( int w , int h )
        :       width( w ) ,
                height( h )
        {}
        virtual void draw( int , int ) ;
} ;

void FigureBox::draw( int x , int y )
{
        GWline( x , y ,                         x + width , y ) ;
        GWline( x + width , y ,                 x + width , y + height ) ;
        GWline( x + width , y + height ,        x , y + height ) ;
        GWline( x , y + height ,                x , y ) ;
}

class FigureCircle : public Figure {
private:
        int radius ;    // 半径だけ
public:
        FigureCircle( int r )
        :       radius( r )
        {}
        virtual void draw( int x , int y ) ;
} ;

// 指定された座標を中心に円を描く
void FigureCircle::draw( int x , int y )
{       int x0 , y0 , x1 , y1 ;
        x0 = x + radius ;
        y0 = y ;
        for( int th = 0 ; th <= 360 ; th += 3 ) {
                x1 = x + radius * cos( th / 180.0 * 3.1415 ) ;
                y1 = y + radius * sin( th / 180.0 * 3.1415 ) ;
                GWline( x0 , y0 , x1 , y1 ) ;
                x0 = x1 ;
                y0 = y1 ;
        }
}
```

```
// 一つの配列の中に違う図形のデータが混在している。
Figure* array[ 3 ] = {
        new FigureBox( 50 , 50 ) ,
        new FigureCircle( 50 ) ,
        new FigureBox( 200 , 50 ) ,
} ;

int main() {
        // ウィンドウを準備
        GrWin   win( 640 , 480 ) ;

        // 普通のデータ構造の生成と描画
        FigureBox       box( 100 , 150 ) ;      box.draw( 200 , 50 ) ;
        FigureCircle    ring( 100 ) ;           ring.draw( 200 , 200 ) ;

        // 混在した図形を一斉に描画
        for( int i = 0 ; i < 3 ; i++ )
                array[ i ]->draw( 100 + 100 * i , 150 ) ;

        return 0 ;
}
```

第２回課題：専攻科２年「オブジェクト指向プログラミング」

前述のグラフィックスを用いた仮想関数の動作例のプログラムを改良し、仮想関数の使い方を演習を通して理解し、その成果をレポートとして提出せよ。

1. 四角や丸以外に、三角・星・楕円といった異なるデータと要素を持つ図形を仮想導出クラスとして実装し、動作を検証せよ。

2. 図形に対し、色の情報を追加したクラスを作り、それぞれの図形が『色』を持っているとして、正しく描画ができるように改良せよ。
   （ア）　基底クラスFigureが色を持たせる方法
   （イ）　図形に対し、色を追加した導出クラスを作る方法
   と、いろいろな実装方法が考えられる。
   色だけに留まらず、線の太さの情報を持つ場合など、機能を追加する色々なケースを想定せよ。