O-O Programming & Data Structure Lab. 6

6.1 C++ programming with Inheritance, Polymorphism, Virtual Function, ConsolePixel Drawing

(1) class **ConsolePixelDrawing**

```
/* ConsolePixelDrawing.h */
#ifndef PIXEL_DRAWING_H
#define PIXEL_DRAWING_H

#include <iostream>
#include <string>
#include <Windows.h>
#include <conio.h>
#include "Shape.h"
#include "Color.h"

using namespace std;

/* PEN_Styles */
#define PS_SOLID            0
#define PS_DASH             1          // -------
#define PS_DOT              2          // .......
#define PS_DASHDOT          3          // _._._._
#define PS_DASHDOTDOT       4          // _.._.._
#define PS_NULL             5
#define PS_INSIDEFRAME      6
#define PS_USERSTYLE        7
#define PS_ALTERNATE        8

#define MAX_NUM_SHAPES 100

class Shape;

class ConsolePixelFrame
{
public:
        ConsolePixelFrame(int org_x, int org_y);
        ~ConsolePixelFrame();
        void addShape(Shape* new_shape);
        void drawShapes();
        int get_pos_org_x() { return pos_org_x; }
        int get_pos_org_y() { return pos_org_y; }
        HDC getConsole_DC() { return console_DC; }
private:
        HWND console;
        HDC console_DC; // device context
        Shape **pShapes; // Array of Shape Pointers
        int num_shapes;
        int capacity;
        int pos_org_x;
        int pos_org_y;
        bool isValidIndex(int sub);
};
#endif
```

(2) class **Color**

```
/** Color.h */

#ifndef COLOR_H
#define COLOR_H
#include <Windows.h>
#include <iostream>
#include <string>
```

```
#include <iomanip>

using namespace std;

// COLORREF is defined in <Windows.h>
// The COLORREF value is used to specify an RGB color,
//     in hexadecimal form of 0x00bbggrr
const COLORREF RGB_RED = 0x000000FF;
const COLORREF RGB_GREEN = 0x0000FF00;
const COLORREF RGB_BLUE = 0x00FF0000;
const COLORREF RGB_BLACK = 0x00000000;
const COLORREF RGB_ORANGE = 0x0000A5FF;
const COLORREF RGB_YELLOW = 0x0000FFFF;
const COLORREF RGB_MAGENTA = 0x00FF00FF;
const COLORREF RGB_WHITE = 0x00FFFFFF;
ostream& fprintRGB(ostream& ostr, COLORREF color);
// RGB color code chart: https://www.rapidtables.com/web/color/RGB_Color.html
/* Note: RGB(red, green, blue) macro also provides COLORREF data
      . RGB(FF, 00, 00) => 0x000000FF (RGB_RED)
      . RGB(00, FF, 00) => 0x0000FF00 (RGB_GREEN)
      . RGB(00, 00, FF) => 0x00FF0000 (RGB_BLUE)
*/
#endif
```

(3) class **Shape**

```
/* Shape.h */

#ifndef SHAPE_H
#define SHAPE_H

#include <string>
#include <Windows.h>
#include <conio.h>
#include "ConsolePixelDrawing.h"
#include "Color.h"

using namespace std;
#define PI 3.14159

class ConsolePixelFrame;
class Shape
{
        friend ostream& operator<<(ostream &, Shape &);
public:
        Shape(); // default constructor
        Shape(string name);
        Shape(int px, int py, double angle, COLORREF ln_clr, COLORREF br_clr, int pen_thick, string name);
        virtual ~Shape();
        virtual void draw(ConsolePixelFrame cp_frame);
        void fprint(ostream &);
        int get_pos_x() const { return pos_x; }
        int get_pos_y() const { return pos_y; }
        void set_pos_x(int x) { pos_x = x; }
        void set_pos_y(int y) { pos_y = y; }
        string getName() { return name; }
        void setName(string n) { name = n; }
        Shape& operator=(const Shape& s);
protected:
        int pos_x; // position x
        int pos_y; // position y
        double angle;
        string name;
        int pen_thickness;
        COLORREF line_color;
        COLORREF brush_color;
};
#endif
```

(4) class **Circle**

```
/** Circle.h */
#ifndef Circle_H
#define Circle_H

#include <string>
#include "Shape.h"
using namespace std;

#define PI 3.14159

class Circle : public Shape
{
	friend ostream& operator<<(ostream &, Circle &);
public:
	Circle();
	Circle(string name);
	Circle(int px, int py, int r, double ang, COLORREF ln_clr, COLORREF br_clr, int pen_thick,
	    string name);
	//Circle(Circle &tr);
	~Circle();
	double getArea();
	virtual void draw(ConsolePixelFrame cp_frame);
	void fprint(ostream &);
	int getRadius() const { return radius; }
	void setRadius(int r) { radius = r; }
	Circle& operator=(const Circle& cir);

protected:
	int radius;
};
#endif
```

(5) class **Triangle**

```
/** Triangle.h */
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <string>
#include "ConsolePixelDrawing.h"
#include "Shape.h"
using namespace std;

class Triangle : public Shape
{
	//friend ostream& operator<<(ostream &, Triangle &);
public:
	Triangle();
	Triangle(string name);
	Triangle(int px, int py, int b, int h, double ang, COLORREF ln_clr, COLORREF br_clr,
	    int pen_thick, string name);
	~Triangle();
	double getArea();
	virtual void draw(ConsolePixelFrame cp_frame);
	void fprint(ostream &);
	int getBase() { return base; }
	int getHeight() { return tri_height; }
	Triangle& operator=(const Triangle& tri);
protected:
	int base;
	int tri_height;
};
#endif
```

(6) class **Rectang**

```
/** Rectang.h */

#ifndef Rectang_H
#define Rectang_H

#include <string>
#include "ConsolePixelDrawing.h"
#include "Shape.h"
using namespace std;

class Rectang : public Shape
{
        //friend ostream& operator<<(ostream &, Rectangle &);
public:
        Rectang();
        Rectang(string name);
        Rectang(int px, int py, int w, int l, double ang, COLORREF ln_clr, COLORREF br_clr, int pen_thick,
            string name);
        ~Rectang();
        double getArea();
        virtual void draw(ConsolePixelFrame cp_frame);
        void fprint(ostream &);
        int getWidth() { return width; }
        int getLength() { return length; }
        Rectang& operator=(Rectang& rec);

protected:
        int width;
        int length;
};

#endif
```

(7) class **PolyGon**

```
/** Polygon.h */

#ifndef PolyGon_H
#define PolyGon_H

#include <string>
#include "ConsolePixelDrawing.h"
#include "Shape.h"
using namespace std;

class PolyGon : public Shape
{
        //friend ostream& operator<<(ostream &, PolyGonle &);
public:
        PolyGon();
        PolyGon(string name);
        PolyGon(int px, int py, int radius, int num_poly, double ang, COLORREF ln_clr, COLORREF br_clr,
            int pen_thick, string name);
        ~PolyGon();
        virtual void draw(ConsolePixelFrame cp_frame);
        void fprint(ostream &);
        int getRadius() { return radius; }
        int getNumPoly() { return num_poly; }
        PolyGon& operator=(PolyGon& pg);

protected:
        int radius;
        int num_poly;
};

#endif
```

(8) class **Star**

```
/* Star.h */
#ifndef Star_H
#define Star_H

#include <string>
#include "ConsolePixelDrawing.h"
#include "Shape.h"
using namespace std;

class Star : public Shape
{
		//friend ostream& operator<<(ostream &, PolyGonle &);
public:
		Star();
		Star(string name);
		Star(int px, int py, int radius, int num_vertices, double ang, COLORREF ln_clr,
			COLORREF br_clr, int pen_thick, string name);
		//PolyGonle(PolyGonle &pg);
		~Star();
		//double getArea();
		virtual void draw(ConsolePixelFrame cp_frame);
		virtual void draw(); // // used for testing of late binding
		void fprint(ostream &);
		int getRadius() { return radius; }
		int getNumPoly() { return num_vertices; }
		Star& operator=(Star& pg);

protected:
		int radius;
		int num_vertices;
};

#endif
```

(9) class **AngledArc**

```
/* AngleArc.h */
#ifndef ANGLE_ARC_H
#define ANGLE_ARC_H

#include <string>
#include "Shape.h"
using namespace std;

class AngledArc : public Shape
{
    friend ostream& operator<<(ostream&, const AngledArc&);
public:
    AngledArc();
    AngledArc(string name);
    AngledArc(int px, int py, int r, int ang, int start_ang, int sweep_ang, COLORREF ln_clr, COLORREF br_clr, int
    pen_thick, string name);
    //AngledArc(AngledArc &angarc);
    ~AngledArc();
    virtual void draw(ConsolePixelFrame cp_frame);
    virtual void draw(); // // used for testing of late binding
    void fprint(ostream&);
    int getRadius() const { return radius; }
    void setRadius(int r) { radius = r; }
    AngledArc& operator=(const AngledArc& cir);

protected:
    int radius;
    int start_angle;
```

```
    int sweep_angle;
};

#endif
```

(10) class **Cylinder**

```
/* Cylinder.h */
#ifndef CYLINDER_H
#define CYLINDER_H

#include <string>
#include "Shape.h"
using namespace std;

class Cylinder : public Shape
{
    friend ostream& operator<<(ostream&, const Cylinder&);
public:
    Cylinder();
    Cylinder(string name);
    Cylinder(int px, int py, int r, int ang, int height, COLORREF ln_clr, COLORREF br_clr, int pen_thick, string name);
    //Cylinder(Cylinder &cyl);
    ~Cylinder();
    double getArea();
    virtual void draw(ConsolePixelFrame cp_frame);
    virtual void draw(); // // used for testing of late binding
    void fprint(ostream&);
    int getRadius() const { return radius; }
    void setRadius(int r) { radius = r; }
    Cylinder& operator=(const Cylinder& cir);

protected:
    int radius;
    int height;
};

#endif
```

6.2 main() function

(1) main function

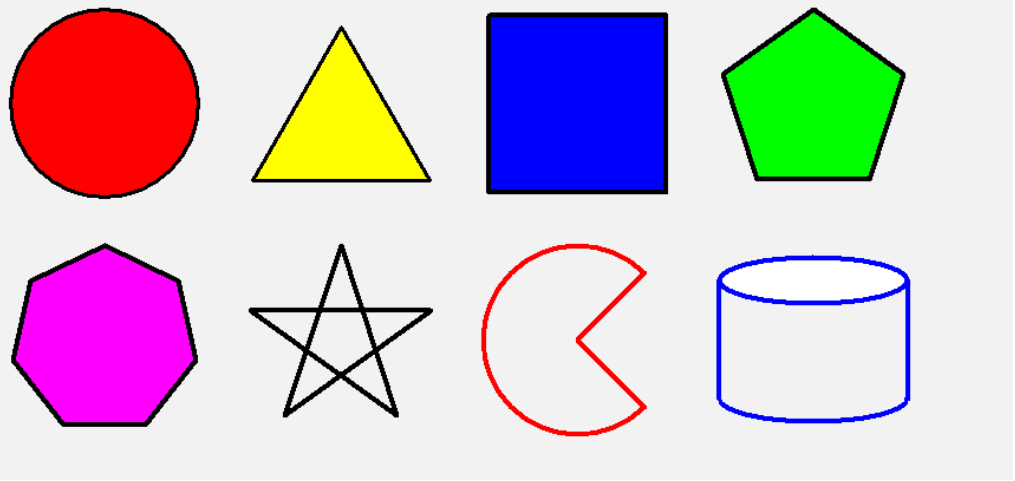
The main() function should contain following procedure to use class Shape, Circle, Cylinder, Rectangle, Pillar, Triangle, and Prism.

```
/** main.cpp    */
#include <iostream>
#include <string>
#include <fstream>
#include "ConsolePixelDrawing.h"
#include "Shape.h"
#include "Triangle.h"
#include "Circle.h"
#include "Rectang.h"
#include "Polygon.h"
#include "Star.h"
#include "AngledArc.h"
#include "Cylinder.h"

using namespace std;

int main()
{
        Circle    cir(100, 200, 80, 0, RGB_BLACK, RGB_RED, 3, "Circle");
        Triangle tri(300, 200, 150, 130, 0, RGB_BLACK, RGB_YELLOW, 3, "Triangle");
        Rectang rec(500, 200, 150, 150, 0, RGB_BLACK, RGB_BLUE, 4, "Rectangle");
        PolyGon poly_5(700, 200, 80, 5, 0, RGB_BLACK, RGB_GREEN, 4, "Polygon_5");
        PolyGon poly_7(100, 400, 80, 7, 0, RGB_BLACK, RGB_MAGENTA, 4, "Polygon_7");
```

```
        Star star_5(300, 400, 80, 5, 0, RGB_BLACK, RGB_GREEN, 4, "Star_5");
        AngledArc angle_arc(500, 400, 80, 0, 45, 270, RGB_RED, RGB_BLUE, 4, "Angle_Arc");
        Cylinder cyl(700, 400, 80, 0, 100, RGB_BLUE, RGB_WHITE, 4, "Cylinder");
        ConsolePixelFrame frame(50, 50); // fr_x, fr_y

        frame.addShape(&cir);
        frame.addShape(&tri);
        frame.addShape(&rec);
        frame.addShape(&poly_5);
        frame.addShape(&poly_7);
       frame.addShape(&star_5);
       frame.addShape(&angle_arc);
        frame.addShape(&cyl);

        frame.drawShapes();
         printf("hit any key to continue ....");
        _getch();
        return 0;

}   // end of main()
```

(2) Example of Output

```
Drawing 8 shapes :
hit any key to continue ....
```



<Oral Test>

6.1 다형성 (polymorphism)이 무엇이며, 왜 필요한가에 대하여 예를 들어 설명하라.

6.2 다형성 (polymorphism)을 구현하기 위하여 사용되는 가상함수 (vritual function)과 지연 바인딩 (late binding)이 무엇이며, 어떻게 동작하는지에 대하여 예를 들어 설명하라.

6.3 가상함수와 late binding 기능을 사용하여 화면에 class Shape으로부터 상속받은 다수의 도형들을 class Shape의 포인터로 drawing하는 방법에 대하여 상세하게 설명하라.

6.4 Upcasting slicing이 어떤 문제이며, 왜 발생하는가에 대하여 예를 들어 설명하라.