实 验 报 告

实验人：梁允楷　　学号：18342055　　日期：2019、6、15
　　李赞辉　　18342053
院（系）：数据科学与计算机学院
专业（班级）：软件工程教务一班
实验题目：　字母画板

一. 实验目的

通过 c++类和对象完成继承和多态的操作，实现在命令行窗口中打印字母及其它图形的功能。

使用面向对象的编程方法，进而继续熟悉继承和多态。

二. 实验环境

本实验基于 Atom 和 Code block 开发，利用 GitHub 进行合作，参考主流的编码规范，如 Google C++Style Guide（中文版）

2.1 编程语言和开发工具

编程语言：　C++

开发工具：　Atom 和 Code block

2.2 编码规范

要求遵循良好的程序设计风格来设计和编写程序。基本编码规范：

1. 标识符的命名要到达顾名思义的程度；
2. 关键代码提供清晰、准确的注释；
3. 程序版面要求：
    a) 不同功能块用空行分隔；
    b) 一般一个语句一行；
    c) 语句缩进整齐、层次分明。

三．题目分析

项目要求在命令行窗口中绘制线条，从而打印我们需要显示的图案。

最基础的线条莫过于横线、竖线、对角线、反对角线，这些也是项目要求中所提及的线条。通过这些基础的线条，我们就可以完成项目的一项简单的要求——打印字母，
为了增加程序的灵活性，我们还可以加入斜线（可以指向任意角度）、圆弧（任意半径，任意弧度）、圆。

加入以上的线条，我们设计的图形可以更复杂，也可以美观。可以说，加入了上述的线条，我们可以很好地打印出任意图形，或者说，打印出其近似图形。

四.分析与设计

4.1、需求分析：

从用户需求的角度考虑，设计要达到的目的是根据用户输入，输出对应的字母。

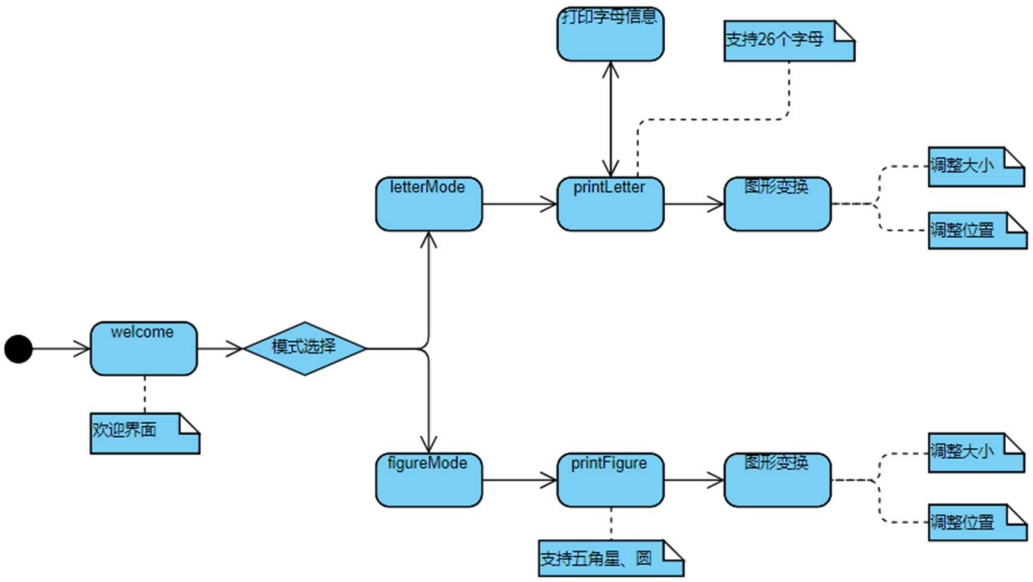
打印出字母后，用户还可以操作按键，打印出的字母进行调整，功能包括位置的调整（上下左右）、字母大小的调整。

另外，我们还可以设计输出当前字母各种信息的功能。这样的信息可以帮助用户以更具

体精确的方式，确认自己得到的字母的信息，从而更好地进行修改调整，

除此以外，我们还可以设计其它图形，满足用户个性化的需求。

系统功能图



4.2、类结构设计

(1) 关于类的设计，我们首先设计了一个图案类 Pattern。这个类是一个抽象类，是所有类的祖先类。它包含了填充信息，同时，有打印图案和打印信息两个纯虚函数。提供了一个模板供后面的图形类还有构成图形的一些基础类使用。

(2) 接下来，就是设计构成图形的一些基础类。

a. 同样，我们先新建一个直线类 Line。

它也是一个抽象类，继承了 Pattern 类，声明了所有具体线条类所共有的变量。

继承 Line 类的有横线 HorizonLine、竖线 verticalLine、对角线 DiagonalLine、反对角线 AntiDiagonalLine、斜线 ObliqueLine、反斜线 AntiObliqueLine，共六个子类。

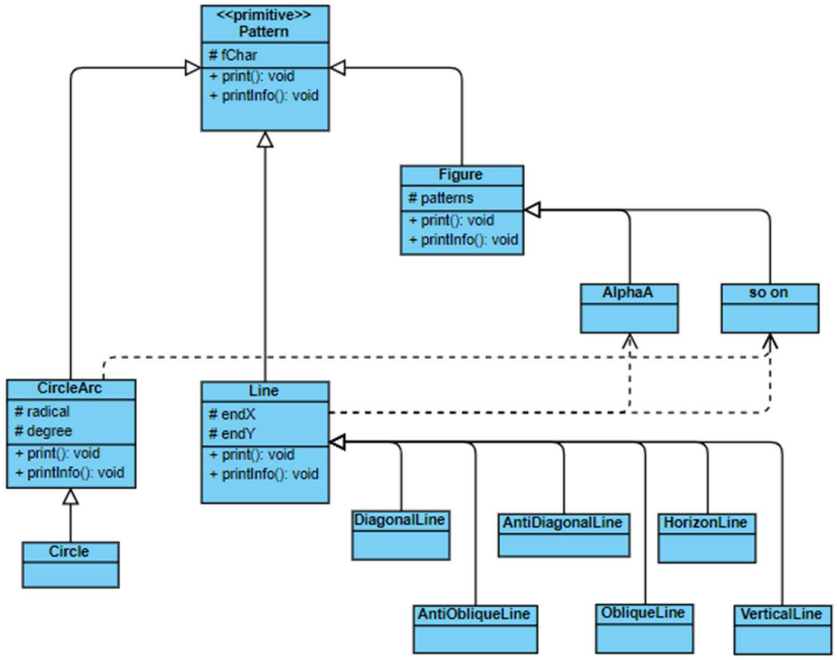
b. 我们还设计了另外一个基础类——圆弧类 CircleArc，继承 Pattern 类。

它是一个具体类，可以实现任意角度开始任意角度结束的圆弧，其中半径的大小也是可以设定。

我们还特例化了一个子类——圆类 Circle，方便用户使用，当然这个类也能调节半径大小。

(3) 然后，就是设计成品图形类 Figure，同样继承了 Pattern 类，包含了一个用于存储图案集合的一个数组。Figure 类之下，我们设计了二十六个大写字母的类，还有较为复杂的五角星类，作为 Figure 的子类。

类关系图：



4.3 细节设计

(1) 接口设计：

Pattern 类设计了相应的两个接口用于打印图案和打印信息。

```
// num1, num2 are used to decide the location of a pattern,
// k is used to describe the size
// In the line class, they means the coordinate of bottom right hand cor
// In the CircleArc class, they means the coordinate of center
class Pattern {
public:
    virtual void print(int num1, int num2, double k) const =0;
    virtual void printInfo(int initX, int initY, double k) const =0;
protected:
    static char fChar;
};
```

Line 类和 Figure 类继承下来仍然是纯虚的接口，供子类作为模板具体实现。

```
class Line: public Pattern {
public:
    Line(int endX, int endY);
    void print(int initX, int initY, double k) const =0;
    void printInfo(int initX, int initY, double k) const =0;
protected:
    int endX, endY;
};
```

```
class Figure: public Pattern {
public:
    void print(int initX, int initY, double k) const =0;
    void printInfo(int initX, int initY, double k) const =0;
protected:
    std::vector<Pattern *> patterns;
};
//standard size of the framework is 11*11, with the Letter size is 9*9
```

CircleArc 类、Circle 类、Line 类和 Figure 类的子类，每个具体类都实现了两个纯虚函数，实现了接口多态的设计。

(2) 数据成员设计：

Pattern 类中，只有一个字符类型数据成员 fChar，用来存储填充图形所用的字符。

Figure 类中，包含一个保存 Pattern 指针的数组 patterns，这里特别设计不用实现图形的基础类作为它的指针类型。因为考虑到日后的拓展性，一个图形应是由图案组成。具体的图形类直接继承这个数组，没有其它数据成员。

在我们的设计中，一个具体的图形类只包含其图案组成的大小信息，它的位置信息和大小信息由 print 函数确定。

在 Line 类中，我们所存储的信息是这根线条构成的矩形的右下角坐标，由两个 int 类型的变量 endX, endY 存储。继承 Line 类的 HorizonLine 类、VerticalLine 类、DiagonalLine 类和 AntiDiagonalLine 类直接继承 Line 的数据成员。特别地，在 ObliqueLine 类和 AntiObliqueLine 类中，加入了 theta 变量，用来存储斜线的角度。

在 CircleArc 类中，含有三个整型类型的数据 begDeg、endDeg、rad 记录起始角度、结束角度和半径长度。

(3) 成员函数：

类中的成员函数主要就是上述接口设计中所提及的 print 和 printInfo 两个函数。
在具体图形类的 print 函数中，其调用相应组成图案的 print 函数，传递进来的信息有打印的位置和大小系数。

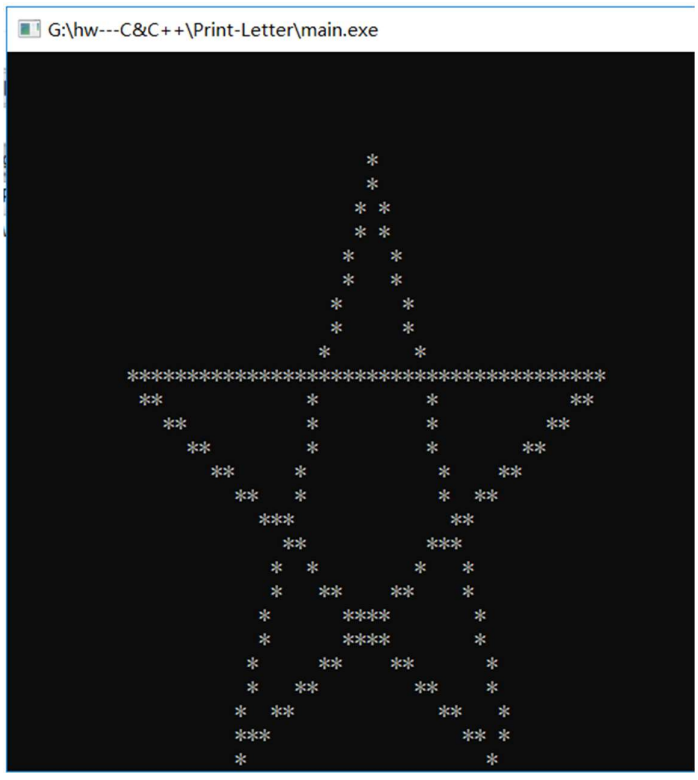
在具体图形类的 printInfo 函数中，传递进来的参数和 print 函数相同。首先是输出这个图形的位置信息，圆（弧）输出的是圆心坐标，其它的图形是输出其左上角的坐标。接下来，输出其组成图案的信息，亦包括对应组成图案的位置信息。

五、实验结果

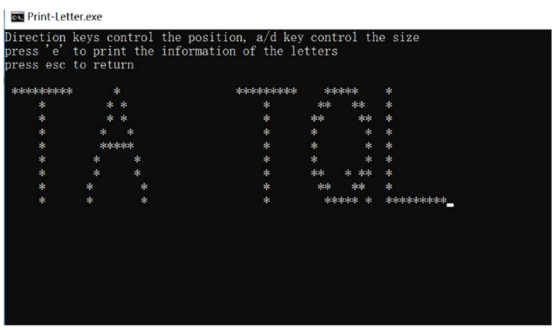
字母展示：

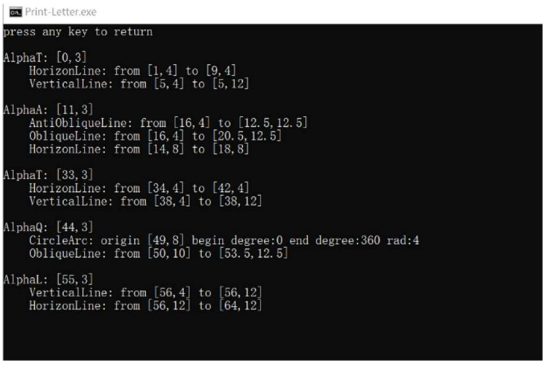
```
Direction keys control the position, a/d key control the size
press 'e' to print the information of the letters
press esc to return

  *******   *       *    *****    *********
     *      *       *   **   **       *
     *      *       *   *            *
     *      *       *   **           *
     *      *       *    *****       *
     *      *       *        **      *
     *      **     **   *     *      *
 ** **       **   **    **   **      *
  ***         *****      *****       *


           ******       *****                 *****    *********
           *    **     **   **                  *          *
           *     **   **     **                 *          *
           *      *   *       *                 *          *
           *      *   *       *                 *          *
           *      *   *       *                 *          *
           *     **   **     **                 *          *
           *    **     **   **                  *          *
           ******       *****                 *****       *
```

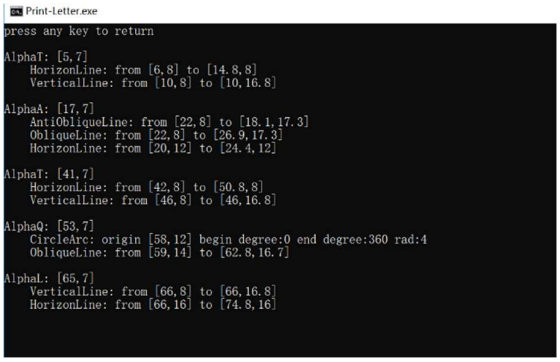
图形展示:



字母信息展示：







六、设计心得

本次实验中，我们实践了多态与继承，学习了在项目中构建复杂的继承关系、利用虚函数进行相应的应用，我们在原有项目基础上，也进行了很大程度的创新，特别是实现了任意角度的斜线和任意弧度的弧线，帮助我们画出更具辨识度的字母和更复杂的图形，在构建这些特殊的线中，我们进行了许多构想与设计。

在设计的过程中，我们经过了无数次的重构，思考图案的位置信息要不要存储在图案类中。思考的结果最终决定我们的设计理念：图案本身不考虑它自己的位置，而是由上一层（调用者）决定下一层的位置。如，字母类中存储的线条类也只有大小信息，在打印字母的时候由调用者传递参数确定字母打印在什么地方，然后在字母的 print 函数中，确定字母的线条打印的位置，传递到对应的 print 函数。

七、github 传送门

https://github.com/LEEzanhui/Print-Letter