实 验 报 告

实验人：颜彬 学号：16337269 日期：2017年6月1日

院（系）：数据科学与计算机学院 专业（班级）： 计算机类（四班）

实验题目： **多项式计算器**

1. **实验目的**

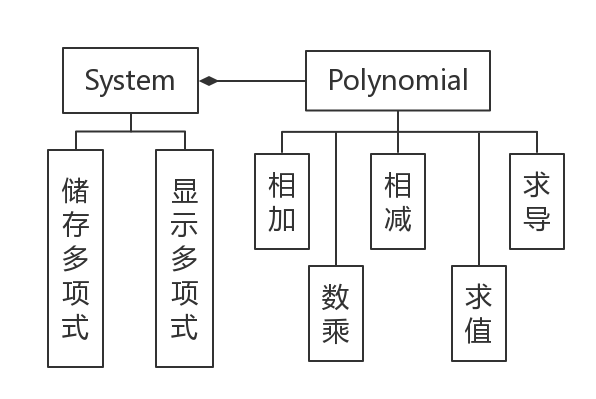
熟悉面向对象的编程思想以及类的使用。

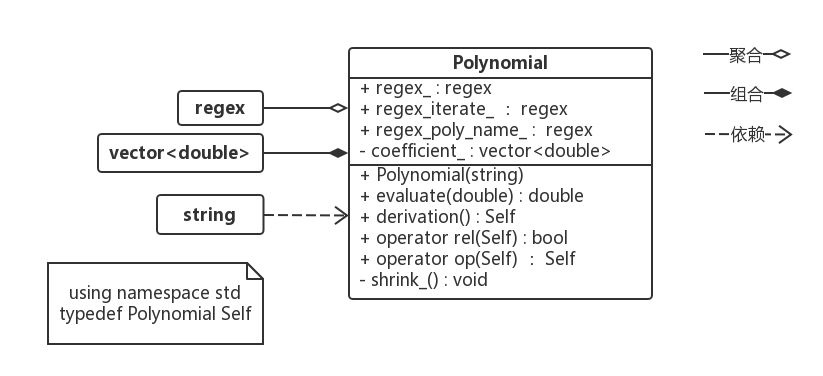
1. **实验环境**
   1. 编程语言和开发工具
      1. 语言采用ANSI C++（C++11）
      2. 开发工具：vim，g++，git，Visual Stdio（用于生成exe文件）
   2. **编码规范**

**编码规范中所有没有涉及到的内容，参见google style C++**

* + 1. **类名**
       1. 类名采用开头大写的方式命名
    2. **变量**
       1. 局部变量采用小驼峰法命名
       2. 类私有和保护的成员变量末尾加下划线
       3. 类公用变量命名同局部变量
    3. **函数**
       1. 函数采用大驼峰命名法命名
       2. 类的私有成员函数末尾加下划线，如 void shrink\_();
       3. get/set/isXXX等函数采用开头小写get/set/is，其后接开头大写单词的方式命名.如isValid, setName.
    4. **其他**
       1. 常量采用字母k + 大驼峰命名法命名。但类中的const变量依旧采用类变量的方式命名
       2. 类型名称（typedef）采用下划线命名法命名。单词最后加\_t。如name\_t.
       3. 左大括号“{”与前一语句在同一行.

1. **实验内容**
2. **分析与设计**
   1. 需求分析：
      1. 系统储存多项式
      2. 多项式相加
      3. 多项式相减
      4. 多项式与常数 数乘运算
      5. 多项式代入某点求值
      6. 系统显示储存的多项式
      7. 显示帮助
   2. 系统功能图



* 1. **类结构设计（Polynomial）**
  2. **细节设计**

**以下+表示public，-表示private，#表示protected**

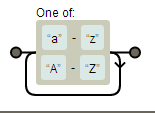
**以 [+-#] operation(type param) : return-type**

**或 [+-#] attribute : type**

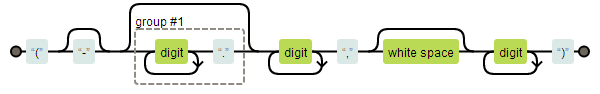
**等形式给出。**

**self代表类本身（Polynomial）**

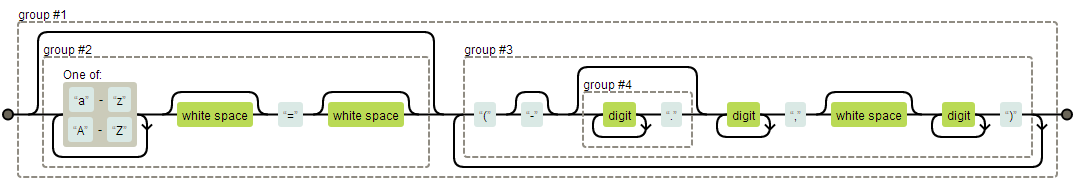
* + 1. **接口设计：**
       1. + typedef vector<double> coefficient\_t;
       2. + typedef Polynomial self;
       3. + Polynomial(string); // 构造函数，对输入作parsing。输入应形为”(1, 3)(2, 4)(4, 6)”
       4. + operator rel(const self&) : bool //关系运算符。rel 为 == 与 !=，判断两个多项式是否相等。
       5. + operator op(const self&) : self //运算符。op为+，-，\*中的一个
       6. + operator op(const self&) : self& //运算符。op为+=，-=，\*=中的一个
       7. + operator op(double) : self //数乘运算符。op为\*, \*=中的一个
       8. + operator<<(ostream&, self) : ostream& // 作cout用
       9. + getDegree() : size\_t // 返回该多项式的次数
       10. + setDegree(int) : void // 设置多项式的次数。若设置的次数低于多项式原本的次数，设置被拒绝。
       11. + getCoefficient() : vector<double> // 返回一个数组，储存多项式系数
       12. + derivation() :　self // 返回该多项式求导后得到的多项式
       13. + evaluate(double) : double // 返回多项式代入某点值后的值
    2. **成员函数：**
       1. – shrink\_() : void // 简化多项式。即将多项式系数为零且没有必要储存的项舍去。
    3. **数据成员：**
       1. – coefficient : vector<double> //储存多项式的系数
    4. **静态成员：**
       1. + isValid(string) : bool // 接受一个代表多项式的字符串，判断其是否合法
       2. + pattern\_ : const string // 用于正则表达式的字符串。判断多项式输入是否合法。用于isValid
       3. + regex\_ : regex //判断输入是否合法的正则。使用了pattern\_
       4. + pattern\_iterate\_ : string //用于正则表达式的字符串。用于提取括号。
       5. + regex\_iterate\_ : const regex // 提取括号。使用了pattern\_iterate\_
       6. + poly\_name\_pattern\_ : const string //用于正则表达式。检测用户提供的多项式的名字是否合法
       7. + regex\_poly\_name\_ : regex // 正则表达式。判断用户提供的多项式的名字是否合法。
  1. 正则匹配



匹配一个单词



提取一个括号



判断输入是否合法

1. **测试心得**

本次项目相比于前几次项目，难度较低。唯一的难点是parsing，即如何将用户的输入正确地转换为多项式。parsing可以用灵活的正则表达式完成。其次的难点是用户交互界面，即如何用尽量简短的代码完成交互界面的需求，写出DRY（don’t repeat yourself）的代码。

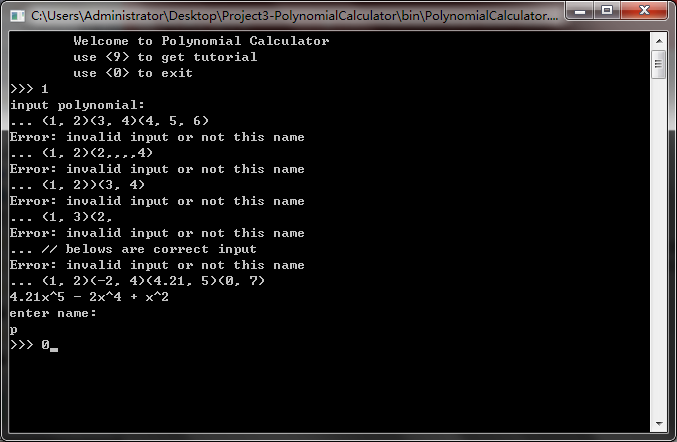
相比之下，多项式类的实现反而显得难度较低。可以采用vector<double>储存按索引储存系数，或采用足够大的double[]储存系数。（本项目不考虑指数为负的情况）。

本次项目有些地方可以更加细致。例如根据“面向接口编程”思想，“交互”应该与“数据”和“接口”分离开。Polynomial类应该从一个Interface\_Polynomial类继承而来。由前者作为接口，后者作为实现。main函数中所有函数的参数都使用Interface\_Polynomial而非Polynomial。

本次项目还可以拓展成允许矩阵运算的计算器。将两者结合在一起。

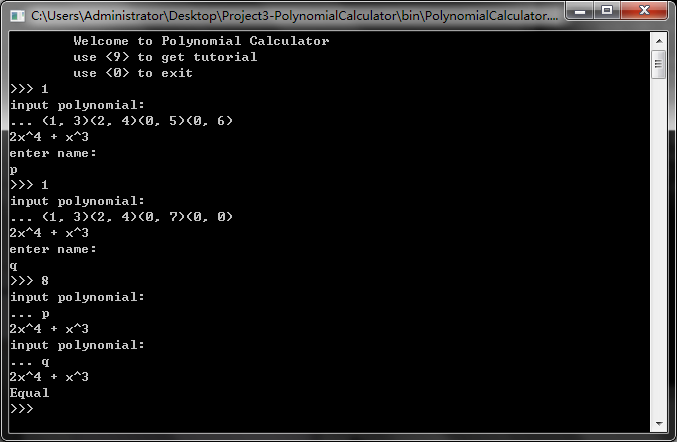
本次项目让我对C++的使用更加熟练。让我对正则的书写更为了解。加深了我的对面向对象的理解。

1. **实验结果**

进行总结，描述所获得的经验和心得体会等。

上述测试基本涵盖了所有可能出现的错误情况。由正则表达式匹配用户输入以保证程序的健壮性。

最后一个输入是正确的输入。可以看到程序正确地识别了系数为0的情况。选择省略系数为零的项的输出。

程序正确地实现了判等操作。程序正确地识别了系数为零的情况。