## Exercise 4: Polymorphism and Inheritance

**1. Purpose and requirements**

**Purpose:** Familiar with Polymorphism and Inheritance, including: (1) Overloading; (2) Inheritance; (3) Overriding; (4) Virtual

**Requirements:** simplicity, clarity, and user friendly

**2. Experiment contents：**

(1) Give a Rational class

class Rational

{

public:

Rational(int nn=1,int mm=1); //constructor

Rational R\_add(Rational & A); //addition

Rational R\_sub(Rational & A); //substraction

Rational R\_mul(Rational & A); //multiplication

Rational R\_div(Rational & A); //division

void print(); //output the result, which should be a reduced form

private:

void simple( ); // [reduction](http://www.nciku.cn/search/en/reduction) of a [fraction](http://www.nciku.cn/search/en/fraction)

int m; // denominator

int n; // numerator

};

Please overload the following operator (denominator cannot be 0)

1） “＋”, “-”, “\*”, “/”

2） “>”, “ <=” and“==”

3） “<<”, output the rational, such as 1/2, -1/3.

(2) Define a base class myArray,

Class myArray

{

int \*alist;

int length;

public:

myArray( int leng);

~myArray();

void input();

void display();

};

Write constructor, destructor, and other function members. Using new and delete to dynamically distribute the memory.

1) Define an averArray class inherits myArray to compute the average, and output the result.

2) Define a revArray class inherits myArray to reverse the elements, and output the result.

3) Try different ways of inheritance (public, protected and private ) to compare the differences.

(3) Implement the examples in 12.7.2 - 12.7.10 (refer to 13.2 - 13.9), write your own graph class, replace all the names of subclass using your own subclass, such as Zhangsan\_circle, and save them in myGraphic.h and myGraphic.cpp.

Learn the subclass design, and familiar the virtual functions (go the definition and declarations). Write a summary to show what have you learnt from this.