**西北大学信息学院数据结构实验课程实验报告**

**实验一 数据结构入门**

**班级：2020计科2 学号： 2020115025 姓名：薄劲阳**

**一 实验目的**

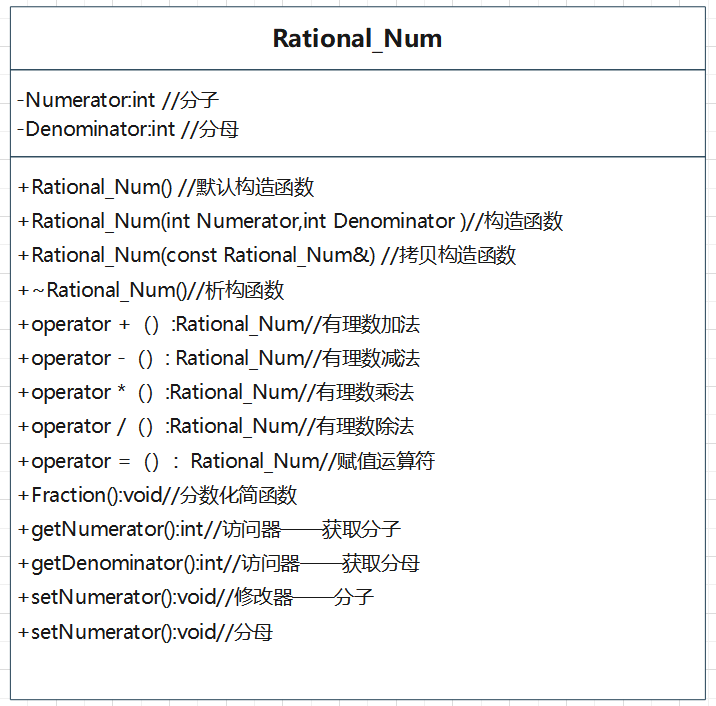
1. 熟悉结构体用法
2. 加深对抽象数据类型的理解

**二 实验内容**

一个有理数可看做由分子和分母构成。设计并实现一个有理数的抽象数据类型，其基本操作包括有理数的加法、减法、乘法和除法，以及求分子和分母，并对这些操作进行测试。

**三 实验设计**

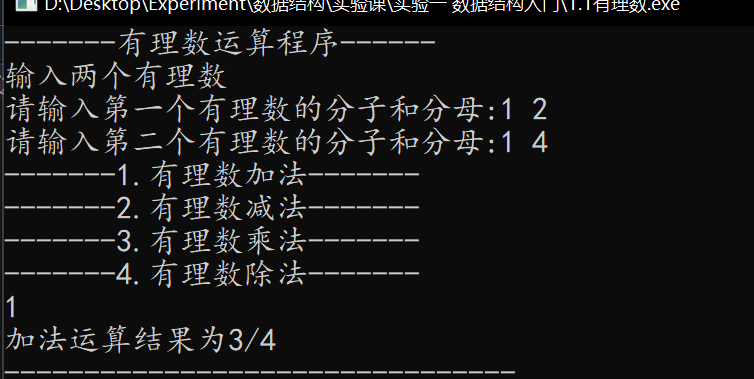
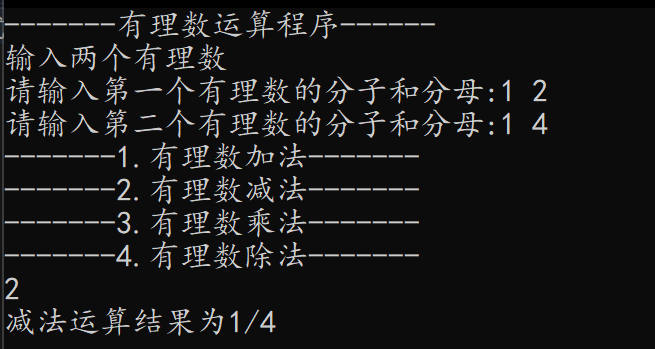
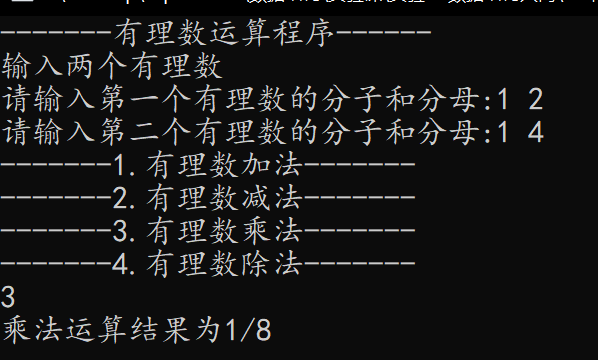
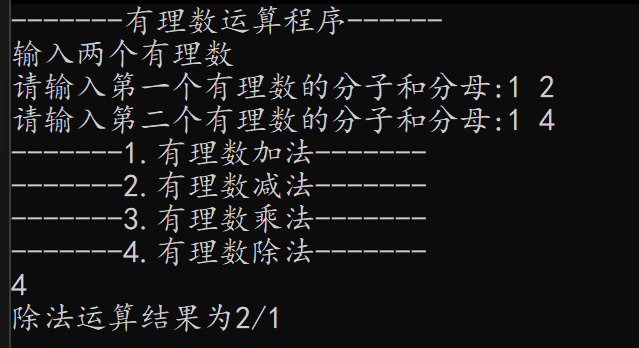
* 1. 数据结构——类
     1. 数据成员
        1. 分子——private
        2. 分母——private
     2. 成员函数
        1. 构造函数
           1. 默认构造函数
           2. 构造函数
           3. 拷贝构造函数
        2. 析构函数
        3. 有理数加法——public
        4. 有理数减法——public
        5. 有理数乘法——public
        6. 有理数除法——public
        7. 分数化简
        8. 求分子——访问器
        9. 求分母——访问器
        10. 修改分子——修改器
        11. 修改分母——修改器



**四 实验代码**

#include<iostream>  
  
using namespace std;  
  
class Rational\_Num {  
        int Numerator;//  
        int Denominator;  
    public:  
        //默认构造函数  
        Rational\_Num(): Numerator(0), Denominator(1) {}  
        //构造函数  
        Rational\_Num(int Numerator, int Denominator) {  
            this->Numerator = Numerator;  
            //分母不能为零  
            if (Denominator != 0) {  
                this->Denominator = Denominator;  
            } else {  
                cout << "Invalid Input.分母不能为0" << endl;  
                cout << "请重新输入分母:";  
                cin >> this->Denominator;  
            }  
        }  
        Rational\_Num(const Rational\_Num & Num): Numerator(Num.Numerator), Denominator(Num.Denominator) {}  
        ~Rational\_Num() {  
        }  
        void setNumerator(int Numerator) {  
            this->Numerator = Numerator;  
        }  
        void setDenominator(int Denominator) {  
            if (Denominator != 0) {  
                this->Denominator = Denominator;  
            } else {  
                cout << "Invalid Input.分母不能为0" << endl;  
            }  
        }  
        int getNumerator() {  
            return Numerator;  
        }  
        int getDenominator() {  
            return Denominator;  
        }  
        //运算符重载技术（非成员函数）——双目运算符要给出两个运算数  
        //有理数加法  
        friend Rational\_Num operator + (Rational\_Num left, Rational\_Num right) {  
            int a = left.Numerator;  
            int b = left.Denominator;  
            int c = right.Numerator;  
            int d = right.Denominator;  
            //分数统一  
            int num = a \* d + b \* c;//求和后的分子  
            int den = b \* d;//分母  
            //化简  
            Rational\_Num Res(num, den);  
            Res.Fraction();//化简结果  
            return Res;  
        }  
        friend Rational\_Num operator - (Rational\_Num left, Rational\_Num right) {  
            //减法转加法  
            //右操作数转相反数  
            right.Denominator = -right.Denominator;//分母取反  
            return Rational\_Num(left + right);  
        }  
        friend Rational\_Num operator \* (Rational\_Num left, Rational\_Num right) {  
            Rational\_Num Res(left.Numerator \* right.Numerator, left.Denominator \* right.Denominator);  
            Res.Fraction();//化简结果  
            return Res;  
        }  
        friend Rational\_Num operator / (Rational\_Num left, Rational\_Num right) {  
            //除法转乘法  
            //被除数颠倒  
            int temp = right.Numerator;  
            right.Numerator = right.Denominator;//分母翻到上面  
            right.Denominator = temp;  
            return Rational\_Num(left \* right);  
        }  
        Rational\_Num& operator = (Rational\_Num num) {  
            //自赋值检测  
            if (this == &num) {  
                return \*this;  
            }  
            Numerator = num.Numerator;  
            Denominator = num.Denominator;  
            return \*this;  
        }  
        //分数化简  
        void Fraction() {  
            int a = Numerator;  
            int b = Denominator;  
            int temp;  
            while (a % b != 0) {  
                temp = a % b;  
                a = b;  
                b = temp;  
            }  
            Numerator = Numerator / b;  
            Denominator = Denominator / b;  
        }  
};  
  
void Menu() {  
    cout << "-------1.有理数加法-------" << endl;  
    cout << "-------2.有理数减法-------" << endl;  
    cout << "-------3.有理数乘法-------" << endl;  
    cout << "-------4.有理数除法-------" << endl;  
}  
  
  
int main() {  
    Rational\_Num Res;  
    cout << "-------有理数运算程序------" << endl;  
    cout << "输入两个有理数" << endl;  
    cout << "请输入第一个有理数的分子和分母:";  
    int num1\_Num, num1\_Den;  
    cin >> num1\_Num >> num1\_Den;  
    Rational\_Num num1(num1\_Num, num1\_Den);  
    cout << "请输入第二个有理数的分子和分母:";  
    int num2\_Num, num2\_Den;  
    cin >> num2\_Num >> num2\_Den;  
    Rational\_Num num2(num2\_Num, num2\_Den);  
loop:  
    Menu();  
    int flag = -1;  
    cin >> flag;  
    switch (flag) {  
        case 1:  
            Res = num1 + num2;  
            cout << "加法运算结果为" << Res.getNumerator() << "/" << Res.getDenominator();  
            break;  
        case 2:  
            Res = num1 - num2;  
            cout << "减法运算结果为" << Res.getNumerator() << "/" << Res.getDenominator();  
            break;  
        case 3:  
            Res = num1 \* num2;  
            cout << "乘法运算结果为" << Res.getNumerator() << "/" << Res.getDenominator();  
            break;  
        case 4:  
            Res = num1 / num2;  
            cout << "除法运算结果为" << Res.getNumerator() << "/" << Res.getDenominator();  
            break;  
        default:  
            cout << "输入有误，请重新输入" << endl;  
            goto loop;  
            break;  
    }  
  
    return 0;  
}

**五 实验效果**

* 加法
  + 
* 减法
  + 
* 乘法
  + 
* 除法
  + 

**六 经验分析**

* Rational\_Num类的设计
* 运算符重载技术
  + 自赋值运算符（只有成员函数写法）
  + 加法、减法、除法、乘法
* 分数的化简——求分子分母的最大公约数（辗转相除法）