

# 面向对象程序设计 42042002

## 作业：Fraction

学号：1951510      姓名：姜文渊

Tongji University

更新：2022 年 7 月 13 日

测试截图放在 imgs 目录中。

## 1 功能概览

本次作业中用 C++ 语言完成了完成一个分数类（fraction）的构建。分数类支持如下功能：

1. 取负运算（例： $+2/3 \rightarrow -2/3$ ，或者  $-2/3 \rightarrow +2/3$ ）
2. 求倒数（例： $2/3 \rightarrow 3/2$ ）
3. 约分（例： $6/9 \rightarrow 2/3$ ）
4. 从 double 型构造分数（例： $0.25 \rightarrow 1/4$ ）
5. 从字符串构造分数（例“ $1/4$ ” $\rightarrow 1/4$ ）
6. 高精度算术运算（加、减、乘、除）
7. 关系运算（ $>$ 、 $<$ 、 $>=$ 、 $<=$ 、 $==$ 、 $!=$ ）
8. 分数转换为字符串，显示分数：当出现分母为 1 时，只输出分子；当分子分母相同时输出 1；当分母是 0 时报告异常

由于笔者实现的 fraction 类的底层是基于字符串的计算，故而可以支持高精度的计算（实际取决于机器的内存大小）。为了方便手动测试该 fraction 类，笔者同时实现了一个简单的 REPL（Read-Eval-Print Loop，“读取-求值-输出”循环），支持通过命令行的方式使用该类。该 REPL 支持如下指令：

1. help 显示帮助信息
2. init 清空所有变量，重置工作区
3. li [var\_name:str] [a/b] 分数变量赋值
4. lf [var\_name:str] [f: double] 将浮点数转化为靠近的分数

5. `show [var_name:str]` 显示变量的值
6. `list` 显示工作区所有变量的值
7. `reduce [var:str]` 将分数化为最简形式
8. `neg [var:str]` 取负数
9. `inv [var:str]` 取倒数
10. `add/sub/mul/div [t:str] [s1:str] [s2:str]` 四则运算,  $t = s1 \text{ op } s2$
11. `eq/gt/lt/geq/leq/neq [s1:str] [s2:str]` 比较运算,  $s1 \text{ op } s2$

若要在二次开发中使用笔者的分数类, 需要引入以下源文件:

1. `mynat.h`
2. `mynat.cpp`
3. `myint.h`
4. `myint.cpp`
5. `fraction.h`
6. `fraction.cpp`

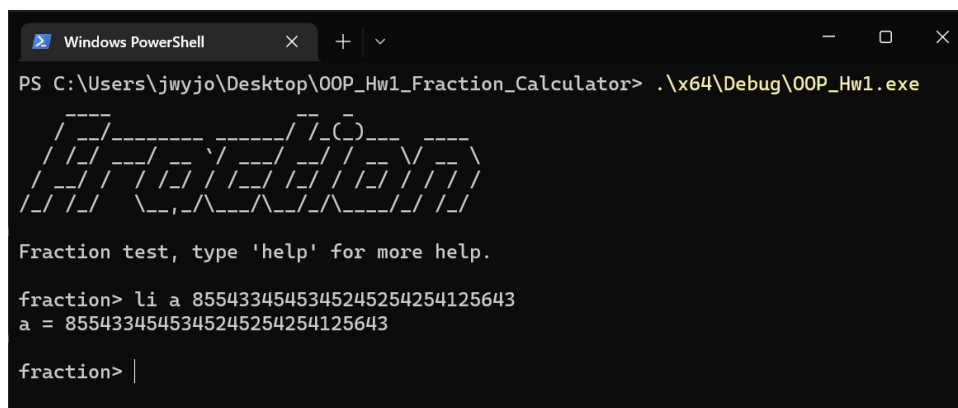
并在使用到分数类的源码文件中引入 `#include"fraction.h"`, 然后即可使用 `fraction` 类。具体接口见 `fraction.h`。

## 2 功能测试

这些测试里没有对输出进行单独的测试, 因为每种测试中都涉及到输出。

### 2.1 分数的构造

正整数 测试用例 `li a 8554334545345245254254125643`



```
Windows PowerShell
PS C:\Users\jwyjo\Desktop\OOP_Hw1_Fraction_Calculator> .\x64\Debug\OOP_Hw1.exe

      /---\
     / /   \ \
    / /     \ \
   / /       \ \
  / /         \ \
 / /           \ \
/ /             \ \
\ \             / /
 \ \           / /
  \ \         / /
   \ \       / /
    \ \     / /
     \ \   / /
      \---/

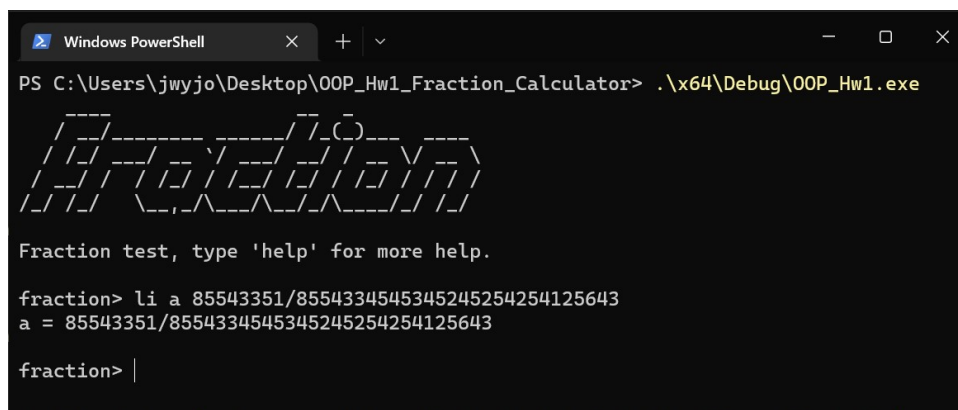
Fraction test, type 'help' for more help.

fraction> li a 8554334545345245254254125643
a = 8554334545345245254254125643

fraction> |
```

图 1: 分数的构造-正整数测试结果 (符合预期)

正分数 测试用例 li a 85543351/8554334545345245254254125643



```
Windows PowerShell
PS C:\Users\jwyjo\Desktop\OOP_Hw1_Fraction_Calculator> .\x64\Debug\OOP_Hw1.exe

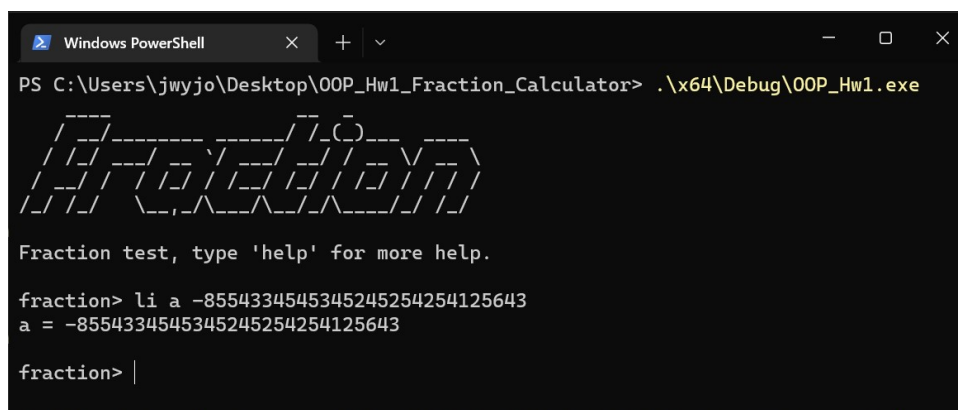
Fraction test, type 'help' for more help.

fraction> li a 85543351/8554334545345245254254125643
a = 85543351/8554334545345245254254125643

fraction> |
```

图 2: 分数的构造-正分数测试结果（符合预期）

负整数 测试用例 li a -8554334545345245254254125643



```
Windows PowerShell
PS C:\Users\jwyjo\Desktop\OOP_Hw1_Fraction_Calculator> .\x64\Debug\OOP_Hw1.exe

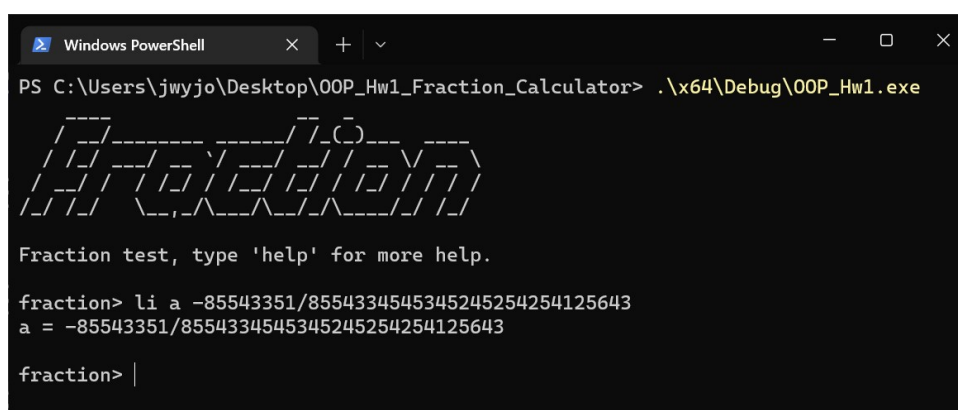
Fraction test, type 'help' for more help.

fraction> li a -8554334545345245254254125643
a = -8554334545345245254254125643

fraction> |
```

图 3: 分数的构造-负整数测试结果（符合预期）

负分数 测试用例 li a -85543351/8554334545345245254254125643



```
Windows PowerShell
PS C:\Users\jwyjo\Desktop\OOP_Hw1_Fraction_Calculator> .\x64\Debug\OOP_Hw1.exe

Fraction test, type 'help' for more help.

fraction> li a -85543351/8554334545345245254254125643
a = -85543351/8554334545345245254254125643

fraction> |
```

图 4: 分数的构造-负分数测试结果（符合预期）

零 测试用例 li a 0

```

PS C:\Users\jwyjo\Desktop\00P_Hw1_Fraction_Calculator> .\x64\Debug\00P_Hw1.exe

      /---/-----/---/---( )---  ---\
     / /  /---/  /---/  /---/  /---/  \
    / /  /---/  /---/  /---/  /---/  \
   / /  /---/  /---/  /---/  /---/  \
  / /  /---/  /---/  /---/  /---/  \
 / /  /---/  /---/  /---/  /---/  \
/ /  /---/  /---/  /---/  /---/  \
 \---/  \---/  \---/  \---/  \---/  \
  \---/  \---/  \---/  \---/  \---/  \
   \---/  \---/  \---/  \---/  \---/  \
    \---/  \---/  \---/  \---/  \---/  \
     \---/  \---/  \---/  \---/  \---/  \
      \---/  \---/  \---/  \---/  \---/  \

Fraction test, type 'help' for more help.

fraction> li a 0
a = 0

fraction> |

```

**图 5: 分数的构造-零的测试结果 (符合预期)**

正浮点数 测试用例 lf a 0.85543345

```
Windows PowerShell X + v
```

```
PS C:\Users\jwyjo\Desktop\OOP_Hw1_Fraction_Calculator> .\x64\Debug\OOP_Hw1.exe
```

```
      _--_/_-----/_--(_)--_--_
    /_/ /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
   /_/ /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
  /_/ /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
 /_/ /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_

Fraction test, type 'help' for more help.
```

```
fraction> lf a 0.85543345
a = 3503/4095
```

```
fraction> |
```

**图 6: 分数的构造-正浮点数测试结果 (符合预期)**

负浮点数 测试用例 lf a -0.85543345

```
PS C:\Users\jwyjo\Desktop\OOP_Hw1_Fraction_Calculator> .\x64\Debug\OOP_Hw1.exe
```

A fractal-like ASCII art logo consisting of multiple rows of backslashes (\) and forward slashes (/). The pattern forms a central vertical column flanked by diagonal lines, creating a complex geometric shape.

Fraction test, type 'help' for more help.

```
fraction> lf a -0.85543345  
a = -3503/4095  
  
fraction> |
```

**图 7: 分数的构造-负浮点数测试结果 (符合预期)**

## 2.2 分数的化简

正分数 测试用例 549755813888/1099511627776

[illegible]

**图 8:** 分数的化简-正分数测试结果 (符合预期)

负分数 测试用例 -549755813888/1099511627776

[illegible]

**图 9: 分数的化简-负分数测试结果 (符合预期)**

正整数 测试用例 1099511627776/549755813888

[illegible]

**图 10: 分数的化简-正整数测试结果 (符合预期)**

负整数 测试用例 -1099511627776/549755813888

```
Windows PowerShell
```

```
/ _/_/----- / _(_)_-- _--  
// // _--/ _- \_ --/ _// _\ V _-- \  
/_ _// // // // // // // // // //  
/_// _\_ ,\_/_--\_/_/_--/_/_// //
```

Fraction test, type 'help' for more help.

```
fraction> li a -1099511627776/549755813888  
a = -1099511627776/549755813888
```

```
fraction> reduce a  
a = -1099511627776/549755813888 = -2
```

```
fraction>
```

**图 11: 分数的化简-负整数测试结果 (符合预期)**

### 零分母 测试用例 1/0

```
Windows PowerShell
```

```
[Warning] creating a fraction with denominator = 0
a = 1/0

fraction> reduce a
a = 1/0
[Warning] creating a fraction with denominator = 0

[Divide by zero] 1/0 分母不能为零

[Warning] creating a fraction with denominator = 0
= 1/0

fraction> |
```

**图 12:** 分数的化简-零分母测试结果 (符合预期)

零分子 测试用例 0/1099511627776

[illegible]

**图 13:** 分数的化简-零分母测试结果 (符合预期)

## 2.3 取负数

正数 测试用例 549755813888/1099511627776

```
Windows PowerShell
/ _/----- / _(_)--- _--
/ / / _/ _/ \ _/ _/ \ _/ \
/ _/ / / / / / / / / / / /
/ / / \ _/ \ _/ \ _/ \ _/ /
Fraction test, type 'help' for more help.

fraction> li a 549755813888/1099511627776
a = 549755813888/1099511627776

fraction> neg a
a = -549755813888/1099511627776

fraction> |
```

**图 14:** 取负数-正分数测试结果 (符合预期)

负数 测试用例 -549755813888/1099511627776

[illegible]

**图 15:** 取负数-负分数测试结果（符合预期）

## 零分母 测试用例 1/0

```
Windows PowerShell
Fraction test, type 'help' for more help.

fraction> li a 1/0

[Warning] creating a fraction with denominator = 0
a = 1/0

fraction> neg a

[Warning] creating a fraction with denominator = 0
a = -1/0

fraction> |
```

**图 16: 取负数-零分母测试结果 (符合预期)**

零分子 测试用例 0/1099511627776

[illegible]

**图 17: 取负数-零分母测试结果 (符合预期)**

## 2.4 取倒数

正数 测试用例 549755813888/1099511627776

[illegible]

**图 18:** 取倒数-正分数测试结果 (符合预期)

负数 测试用例 -549755813888/1099511627776

```
Windows PowerShell
```

```
/ _/_/----- /_(_)_-- _--  
// // _--/ _-- \ _--/ _-- \ V _-- \  
/_ _--/_ // // // // // // // // //  
/_/_/_ \ _--,\ _--\ _--\ _--\ _--/_/_/  
  
Fraction test, type 'help' for more help.  
  
fraction> li a -549755813888/1099511627776  
a = -549755813888/1099511627776  
  
fraction> inv a  
a = 1099511627776/-549755813888  
  
fraction>
```

**图 19: 取倒数-负分数测试结果 (符合预期)**



## 零分母 测试用例 1/0

```
Windows PowerShell
/ _/ / / / _/ / _/ / _/ / _/ /
/_/ / _/ \ _/ _/ \ _/ _/ \ _/ /
Fraction test, type 'help' for more help.

fraction> li a 1/0

[Warning] creating a fraction with denominator = 0
a = 1/0

fraction> inv a
a = 0

fraction> |
```

**图 20:** 取倒数-零分母测试结果 (符合预期)

零分子 测试用例 0/1099511627776

[illegible]

**图 21:** 取倒数-零分母测试结果 (符合预期)

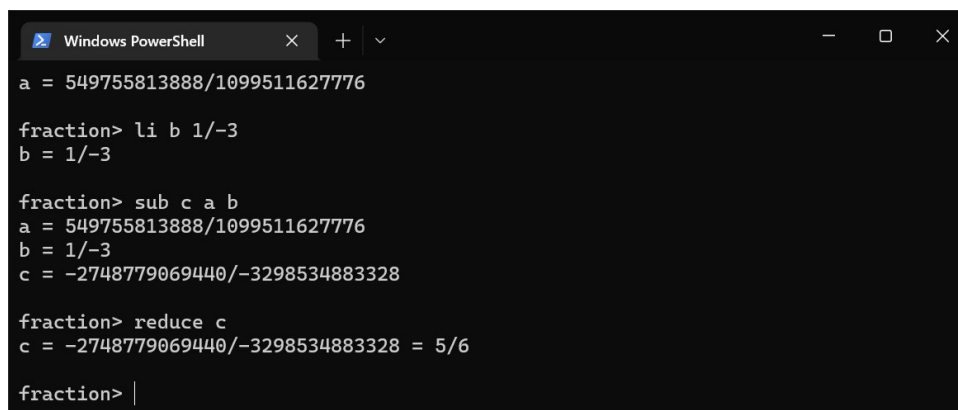
## 2.5 四则运算

加法 测试用例 549755813888/1099511627776 + 1/-3

```
Windows PowerShell
a = 549755813888/1099511627776
fraction> li b 1/-3
b = 1/-3
fraction> add c a b
a = 549755813888/1099511627776
b = 1/-3
c = -549755813888/-3298534883328
fraction> reduce c
c = -549755813888/-3298534883328 = 1/6
fraction> |
```

**图 22: 四则运算-加法测试结果 (符合预期)**

减法 测试用例  $549755813888/1099511627776 - 1/-3$



```
Windows PowerShell
a = 549755813888/1099511627776

fraction> li b 1/-3
b = 1/-3

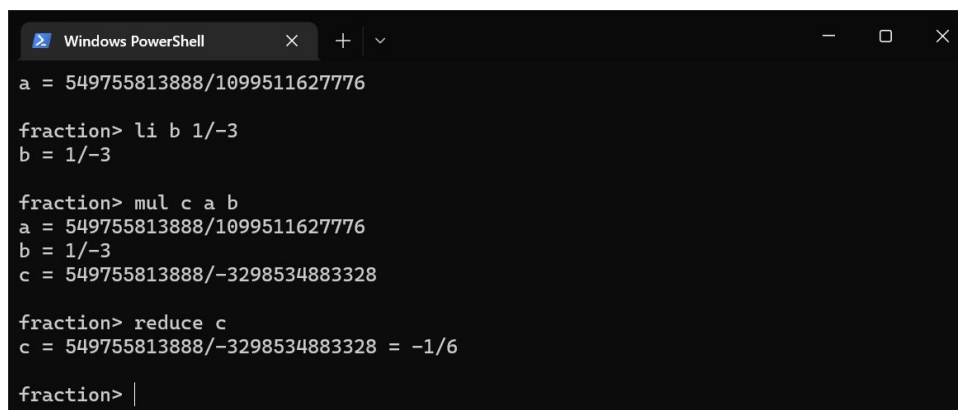
fraction> sub c a b
a = 549755813888/1099511627776
b = 1/-3
c = -2748779069440/-3298534883328

fraction> reduce c
c = -2748779069440/-3298534883328 = 5/6

fraction> |
```

图 23: 四则运算-减法测试结果（符合预期）

乘法 测试用例  $549755813888/1099511627776 * 1/-3$



```
Windows PowerShell
a = 549755813888/1099511627776

fraction> li b 1/-3
b = 1/-3

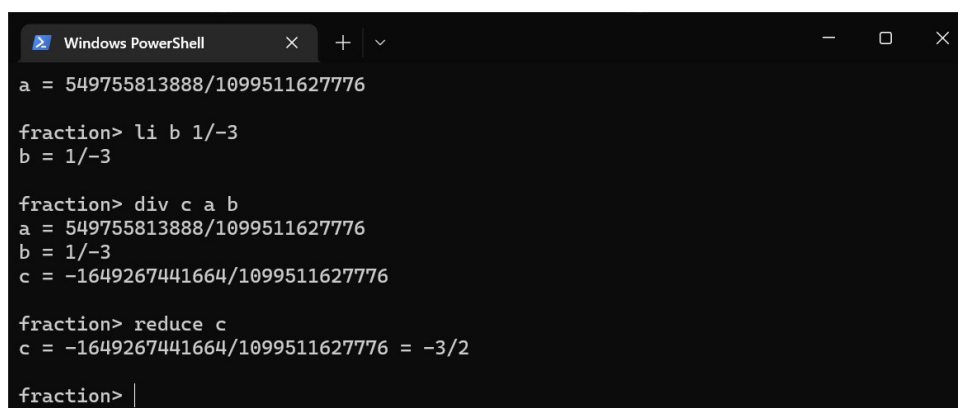
fraction> mul c a b
a = 549755813888/1099511627776
b = 1/-3
c = 549755813888/-3298534883328

fraction> reduce c
c = 549755813888/-3298534883328 = -1/6

fraction> |
```

图 24: 四则运算-乘法测试结果（符合预期）

除法 测试用例  $549755813888/1099511627776 / 1/-3$



```
Windows PowerShell
a = 549755813888/1099511627776

fraction> li b 1/-3
b = 1/-3

fraction> div c a b
a = 549755813888/1099511627776
b = 1/-3
c = -1649267441664/1099511627776

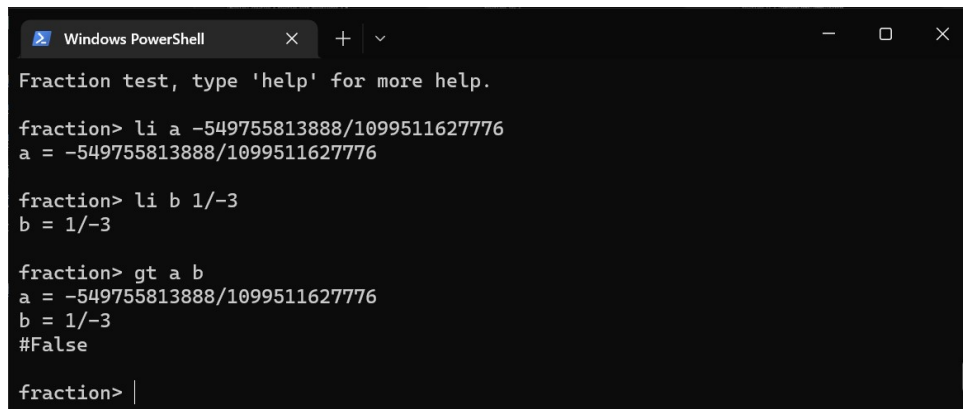
fraction> reduce c
c = -1649267441664/1099511627776 = -3/2

fraction> |
```

图 25: 四则运算-除法测试结果（符合预期）

## 2.6 比较运算

大于 测试用例  $-549755813888/1099511627776 > 1/-3$



```
Windows PowerShell
Fraction test, type 'help' for more help.

fraction> li a -549755813888/1099511627776
a = -549755813888/1099511627776

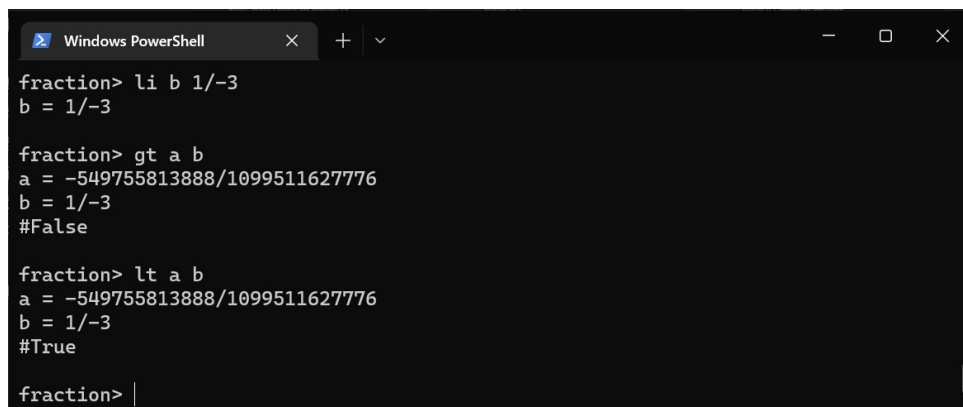
fraction> li b 1/-3
b = 1/-3

fraction> gt a b
a = -549755813888/1099511627776
b = 1/-3
#False

fraction> |
```

图 26: 比较运算-除法测试结果 (符合预期)

小于 测试用例  $-549755813888/1099511627776 < 1/-3$



```
Windows PowerShell
fraction> li b 1/-3
b = 1/-3

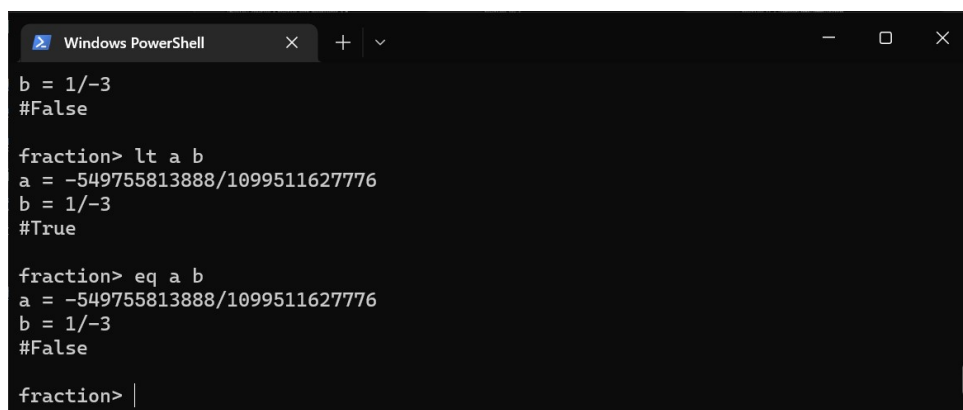
fraction> gt a b
a = -549755813888/1099511627776
b = 1/-3
#False

fraction> lt a b
a = -549755813888/1099511627776
b = 1/-3
#True

fraction> |
```

图 27: 比较运算-除法测试结果 (符合预期)

等于 测试用例  $-549755813888/1099511627776 == 1/-3$



```
Windows PowerShell
b = 1/-3
#False

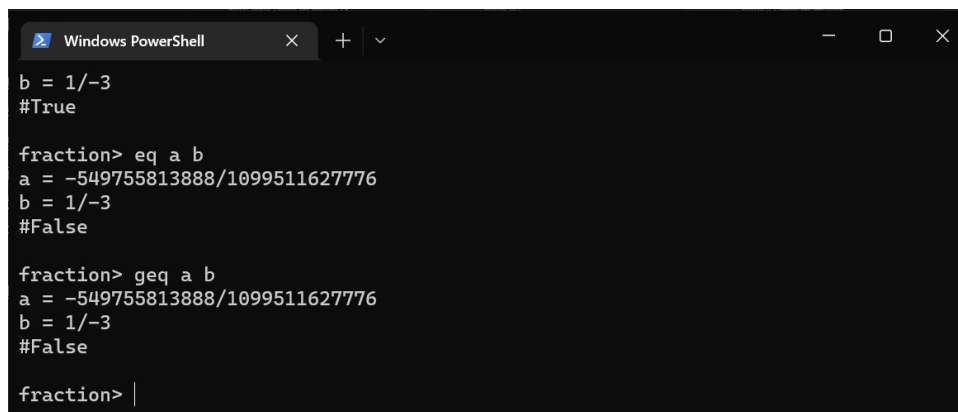
fraction> lt a b
a = -549755813888/1099511627776
b = 1/-3
#True

fraction> eq a b
a = -549755813888/1099511627776
b = 1/-3
#False

fraction> |
```

图 28: 比较运算-除法测试结果 (符合预期)

大于等于 测试用例  $-549755813888/1099511627776 \geq 1/-3$



```
Windows PowerShell
b = 1/-3
#True

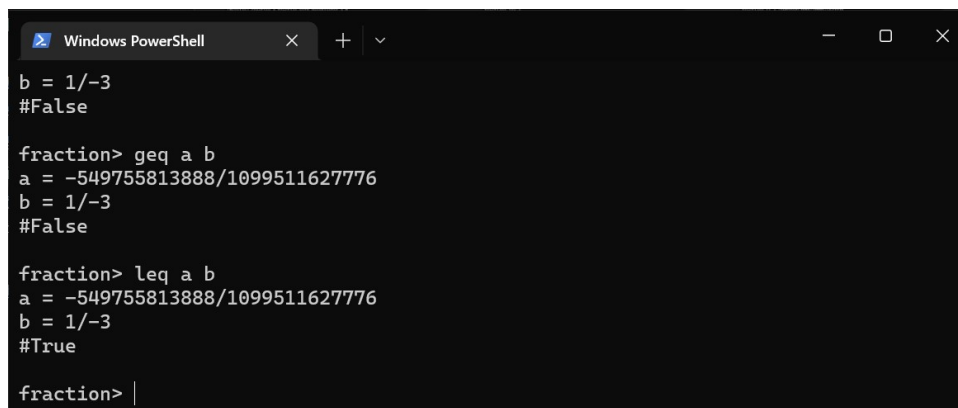
fraction> eq a b
a = -549755813888/1099511627776
b = 1/-3
#False

fraction> geq a b
a = -549755813888/1099511627776
b = 1/-3
#False

fraction> |
```

图 29: 比较运算-大于等于测试结果 (符合预期)

小于等于 测试用例  $-549755813888/1099511627776 \leq 1/-3$



```
Windows PowerShell
b = 1/-3
#False

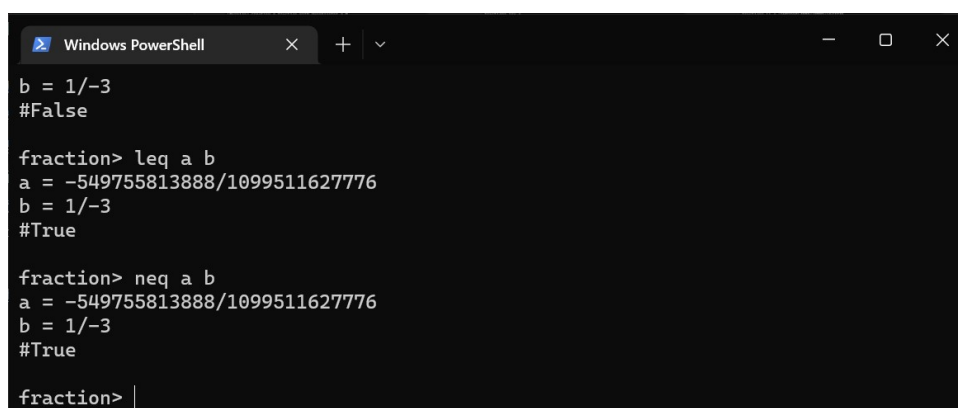
fraction> geq a b
a = -549755813888/1099511627776
b = 1/-3
#False

fraction> leq a b
a = -549755813888/1099511627776
b = 1/-3
#True

fraction> |
```

图 30: 比较运算-小于等于测试结果 (符合预期)

不等于 测试用例  $-549755813888/1099511627776 \neq 1/-3$



```
Windows PowerShell
b = 1/-3
#False

fraction> leq a b
a = -549755813888/1099511627776
b = 1/-3
#True

fraction> neq a b
a = -549755813888/1099511627776
b = 1/-3
#True

fraction> |
```

图 31: 比较运算-不等于测试结果 (符合预期)

## 3 设计思路

### 3.1 分层开发

为了实现一个支持任意精度运算分数类，直接使用 `int64` 等常规的数据类型是不够的。同时为了方便开发与调试，应当优先完成分数类的核心功能，即完成分数的构造，运算和比较功能的实现，然后再对其底层依赖的整数进行高精度的改造。

于是笔者使用了如下的策略：首先创建一个 `myint` 类，该类最初是 `int` 的简单封装，提供了整数的构造，运算和比较功能。然后，在构建 `fraction` 类时，分子和分母使用 `myint` 类的对象。在基于 `myint` 类提供的接口实现并测试好 `fraction` 类后，便着手将 `myint` 类重构为支持高精度的版本。

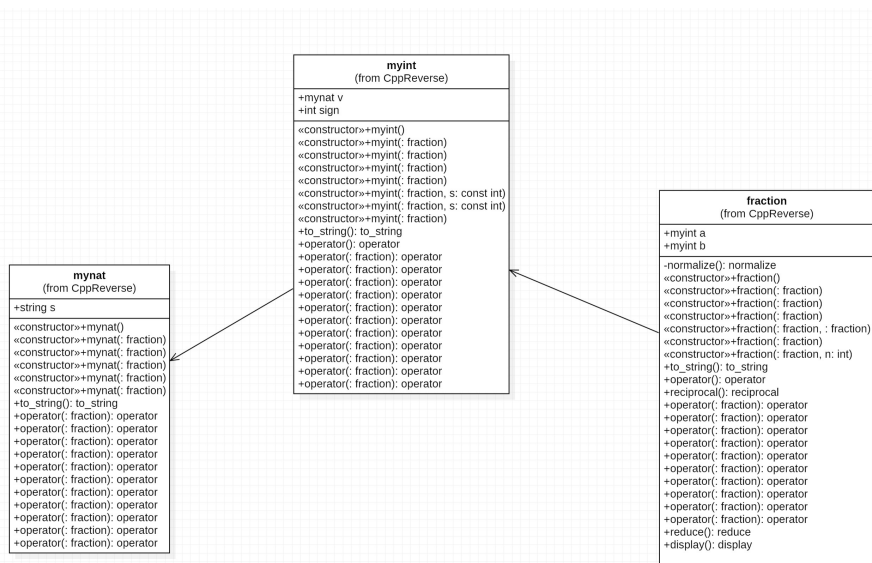
重构 `myint` 类时，为了方便调试，笔者创建了一个 `mynat` 类用于进行高精度的自然数操作，然后在 `myint` 中引入 `mynat` 对象和正负号，就可以方便地实现 `fraction` 类所需要的方法。

### 3.2 组合优于继承

面向对象编程中，有一条非常经典的设计原则，那就是：组合优于继承，多用组合少用继承。这个表述最早出现在 GoF 的《Design Patterns: Elements of Reusable Object-Oriented Software》中。在本次作业中，笔者并没有让整数类继承自然数类，也没有让分数类继承整数类，而是使用组合的方式，在整数类中使用自然数类的对象并在分数类中使用整数类的对象。

这样使用组合的方式，相比使用继承的机制，有诸多优势。除了可以避免继承带来的代码重复率高的问题外，组合对象的方式对于小型项目可以方便重构的进行。一般而言，继承应当在接口和实例之间使用，而非用于实例之间。

### 3.3 类图



**图 32: 本次作业的类图**

## 4 实现细节

## 4.1 高精度正整数类

高精度正整数类的实践主要依托 `std::string` 类，其各个运算的实现均和正常人使用竖式进行计算的过程一致。在二次开发中值得注意的是，由于正整数不包含负数，故而重载的运算符减号实现的是两数字的差的绝对值。

在实现的细节上，`std::string` 类用于倒序存储一个十进制整数的各位数字，即最低位存储在 `s[0]` 中，十位存储在 `s[1]` 中，且笔者的实现保证除了 0 以外，其余正整数的字符串中没有前导的 0。

## 4.2 高精度整数类

高精度整数类使用一个高精度正整数类对象表示其绝对值，使用一个整数表示其符号，该整数的取值为 1 或 -1。对于整数 0，我们规定其符号为正。

其余函数的实现几乎都是按操作数的符号分情况讨论,并直接调用正整数类的成员函数。二次开发时,注意取操作结果的正负号只与被取模的数一致。

### 4.3 高精度分数类

高精度分数类使用两个正整数类对象分别表示分子和分母，一个高精度分数类对象的标准形式为：分子符号表示分数的符号，分母的符号为正，且分子分母的最大公因数为 1。

高精度分数类对象的四则运算主要基于正常的通分进行，且结果不会进行标准化，因为按照作业的要求，对于一个分数的约分和标准化是需要手动进行的。在出现分母为 0 的情况时，分数类会报异常，并且此时的计算结果是不可信的。

## 5 作业小结

在本次作业中，笔者完成了 `fraction` 类的构建与测试，从而实践了面向对象程序设计的一些基本概念，例如封装和多态等。作业中通过构造自然数类和整数类，从而完成了分数类的构建，从而实践了面向对象程序设计的一些思路。

笔者构建的分数类的计算效率主要取决于自然数类的实现，如果将自然数类中的乘法使用 FFT（快速傅里叶变换）或者 NTT（快速数论变换）进行改进，则可以较大地提升性能，而无需修改整数类或者分数类的实现。