上級プログラミング 2019 年前期

出題日：2019 年 5 月 17 日（金）14 時 55 分
提出期限：2019 年 5 月 28 日（火）23 時 59 分
提出方法：各グループで指定された形式・方法により提出すること

レポート課題 2

ソースコード 1に示す main 関数で分数についての各種演算ができるように，以下で述べるような分数のクラス Fraction を作成し，作成したクラスが正しく動作しているかを，図 1，図 2及び図 3の実行例を参考に確認せよ．ただし，実行例の図中の下線はキーボードからの入力を示す．

■Fraction クラスの要件　Fraction クラスはデータメンバとして 2 つの整数型変数を持ち，それぞれの変数は分子と分母を表す．コンストラクタでは，分数の初期化を行う．分数が負の場合は分子を負数にする．分母が 0 の場合はエラーを表示し，プログラムを終了する．コンストラクタに引数が与えられなかった場合，分母が 1，分子が 0 となるように実装する．Fraction クラスは自分自身を約分する reduce 関数をメンバに持つ．また，中置演算子（+,-,*,/,>,<）を用いての四則演算や大小比較も可能である．加えて，演算子 “<<” 及び “>>” をオーバロードしているため，分数の標準入出力も可能である．Fraction クラスを出力する際，一つの分数を括弧で括り，(1/2) や (-2/3) のように表示する．但し，分母が 1 の場合は分数の形式ではなく，通常の整数のように，2 や 3 のように表示する．

■レポート作成に関する注意点　四則演算と入出力演算子のオーバロードについては，メンバ関数かフレンド関数かをよく考え，なぜそのように実装したのかをレポートに明記すること．また，関数（メンバ関数も含む）の引数や，メンバ関数に const 修飾子を付与すべきかどうかについてもよく考え，その理由をレポートに明記すること．

ソースコード 1: Fraction クラスを用いる main 関数の例

```
#include <iostream>
#include <cstdlib> // 分母が0 -> exit(EXIT_FAILURE);
using namespace std;

// ここに要件を満たすFractionクラスを設計し、main関数
    が正常に動作するように実装する。

int main()
{
  const Fraction f1(1,2), f2(2,3), f3(2);
  Fraction f4(2,4), f5(-3,5), f6(0);
  Fraction g1,g2;
  cout << f1 << '+' << f2 << '=' << f1 + f2 << '\n';
  cout << f1 << '-' << f2 << '=' << f1 - f2 << '\n';
  cout << f1 << '*' << f3 << '=' << f1 * f3 << '\n';
  cout << f1 << '-' << f4 << '=' << f1 - f4 << '\n';
  cout << f2 << '/' << f4 << '=' << f2 / f4 << '\n';
  cout << f2 << '/' << f5 << '=' << f2 / f5 << '\n';
  cout << f6 << '*' << f2 << '=' << f6 * f2 << '\n';
  cout << 1  << '+' << f5 << '=' << 1  + f5 << '\n';
  cout << "分数を入力-->"; cin >> g1;
  cout << "分数を入力-->"; cin >> g2;
  if(g1 > g2) cout << g1 << '>' << g2 << '\n';
  if(g1 < g2) cout << g1 << '<' << g2 << '\n';
  if(g1 == g2) cout << g1 << "==" << g2 << '\n';
  cout << g1 << '*' << g2 << '=' << g1 * g2 << '\n';
  cout << g1 << '*' << 20 << '=' << g1 * 20<< '␣' ;
  cout << 20 << '*' << g1 << '=' << 20 * g1 << '\n';
  return 0;
}
```

```
comsv01% ./repo02
(1/2)+(2/3)=(7/6)
(1/2)-(2/3)=(-1/6)
(1/2)*2=1
(1/2)-(1/2)=0
(2/3)/(1/2)=(4/3)
(2/3)/(-3/5)=(-10/9)
0*(2/3)=0
1+(-3/5)=(2/5)
分数を入力-->12 16
分数を入力-->13 39
(3/4)>(1/3)
(3/4)*(1/3)=(1/4)
(3/4)*20=15 20*(3/4)=15
```

図 1: 実行例 1

```
comsv01% ./repo02
(1/2)+(2/3)=(7/6)
(1/2)-(2/3)=(-1/6)
(1/2)*2=1
(1/2)-(1/2)=0
(2/3)/(1/2)=(4/3)
(2/3)/(-3/5)=(-10/9)
0*(2/3)=0
1+(-3/5)=(2/5)
分数を入力-->2 1
分数を入力-->-4 1
2>-4
2*-4=-8
2*20=40 20*2=40
```

図 2: 実行例 2

```
comsv01% ./repo02
(1/2)+(2/3)=(7/6)
(1/2)-(2/3)=(-1/6)
(1/2)*2=1
(1/2)-(1/2)=0
(2/3)/(1/2)=(4/3)
(2/3)/(-3/5)=(-10/9)
0*(2/3)=0
1+(-3/5)=(2/5)
分数を入力-->1 0
分母に 0 が設定されています
```

図 3: 実行例 3