代码：

#include<iostream>

#include<cmath>

#include<cstdlib>

using namespace std;

void simplize(int& z, int& m);

class Fraction {

private:

int zi;

int mu;

public:

Fraction(int z = 0, int m = 1) {

zi = z; mu = m;

}

Fraction(double x) {

for (int i=1;;) {

if (x - (int)x == 0) {

zi = x;

mu = i;

break;

}

x \*= 10;

i \*= 10;

}

simplize(zi, mu);

}

bool operator == (Fraction x) {

return this->zi == x.zi&&

this->mu == x.mu;

}

bool operator != (Fraction x) {

return !(this->zi == x.zi &&

this->mu == x.mu);

}

int operator[](int i) const

{

if (i == 0) return zi ;

if (i == 1) return mu ;

return 0;

}

int& operator[](int i)

{

if (i == 0) return zi;

if (i == 1) return mu;

}

Fraction& operator = (Fraction x) {

this->zi = x.zi;

this->mu = x.mu;

return \*this;

}

Fraction& operator ++() {

zi +=mu;

return \*this ;

}

Fraction& operator ++(int) {

Fraction a = \*this;

zi += mu;

return a;

}

Fraction& operator --() {

zi -= mu;

return \*this;

}

Fraction& operator --(int) {

Fraction a = \*this;

zi -= mu;

return a;

}

friend ostream& operator <<(ostream& o, const Fraction& f);

friend Fraction operator +(const Fraction& a,const Fraction& b);

friend Fraction operator -(const Fraction& a,const Fraction& b);

friend Fraction operator \*(const Fraction& a,const Fraction& b);

friend Fraction operator /(const Fraction& a,const Fraction& b);

friend Fraction& operator +=(Fraction& a, const Fraction& b);

friend Fraction& operator -=(Fraction& a, const Fraction& b);

friend Fraction& operator \*=(Fraction& a, const Fraction& b);

friend Fraction& operator /=(Fraction& a, const Fraction& b);

};

void simplize(int& z, int& m) {

if (z % m == 0) {

z = z / m;

m = 1;

}

else {

int i = abs(z) / z, j = abs(m) / m;

z = abs(z); m = abs(m);

for (int i = 2; i <= z; ++i) {

if (z % i == 0 && m % i == 0) {

z /= i;

m /= i;

i = 2;

}

}

z \*= (j \* i);

}

}

Fraction operator +(const Fraction& a,const Fraction& b) {

int z, m;

z = a.zi \* b.mu + a.mu \* b.zi;

m = a.mu \* b.mu;

simplize(z, m);

return Fraction(z, m);

}

Fraction operator -(const Fraction& a,const Fraction& b) {

int z, m;

z = a.zi \* b.mu - a.mu \* b.zi;

m = a.mu \* b.mu;

simplize(z, m);

return Fraction(z, m);

}

Fraction operator \*(const Fraction& a,const Fraction& b) {

int z, m;

z = a.zi \* b.zi;

m = a.mu \* b.mu;

simplize(z, m);

return Fraction(z, m);

}

Fraction operator /(const Fraction& a,const Fraction& b) {

int z, m;

z = a.zi \* b.mu;

m = a.mu \* b.zi;

simplize(z, m);

return Fraction(z, m);

}

Fraction& operator +=(Fraction& a, const Fraction& b) {

int z, m;

z = a.zi \* b.mu + a.mu \* b.zi;

m = a.mu \* b.mu;

simplize(z, m);

a.zi = z;

a.mu = m;

return a;

}

Fraction& operator -=(Fraction& a, const Fraction& b) {

int z, m;

z = a.zi \* b.mu - a.mu \* b.zi;

m = a.mu \* b.mu;

simplize(z, m);

a.zi = z;

a.mu = m;

return a;

}

Fraction& operator \*=(Fraction& a, const Fraction& b) {

int z, m;

z = a.zi \* b.zi;

m = a.mu \* b.mu;

simplize(z, m);

a.zi = z;

a.mu = m;

return a;

}

Fraction& operator /=(Fraction& a, const Fraction& b) {

int z, m;

z = a.zi \* b.mu;

m = a.mu \* b.zi;

simplize(z, m);

a.zi = z;

a.mu = m;

return a;

}

ostream& operator <<(ostream& o, const Fraction& f) {

if (f.mu == 1) {

o << f.zi << endl;

}

else {

if (f.zi > 0 && f.mu < 0) {

int z=f.zi \*( -1);

int m=f.mu \*( -1);

o << z << "/" << m << endl;

}else{

if (f.zi < 0 && f.mu < 0) {

int z = f.zi \* (-1);

int m = f.mu \* (-1);

o << z << "/" << m<<endl ;

}

else {

o <<f. zi << "/" << f.mu<<endl ;

}

}

}

return o;

}

int main()

{

Fraction f1(2, 5);

Fraction f2 = -4;

Fraction f3 = 0.5;

cout << "STEP1" << endl;

cout << " f1 = " << f1 << endl;

cout << " f2 = " << f2 << endl;

cout << " f3 = " << f3 << endl;

Fraction f4 = f1 + f2 ;

Fraction f5 = f1 - f2 ;

Fraction f6 = f1 \* f2 ;

Fraction f7 = f1 / f2 ;

cout << "STEP2" << endl;

cout << " f4 = " << f4 << endl;

cout << " f5 = " << f5 << endl;

cout << " f6 = " << f6 << endl;

cout << " f7 = " << f7 << endl;

f4 += f1;

f5 -= f1;

f6 \*= f1;

f7 /= f1;

cout << "STEP3" << endl;

cout << " f4 = " << f4 << endl;

cout << " f5 = " << f5 << endl;

cout << " f6 = " << f6 << endl;

cout << " f7 = " << f7 << endl;

f2 = 0.5;

f4 = 0.4;

f1 = f2++;

f3 = ++f4;

cout << "STEP4" << endl;

cout << " f1 = " << f1 << endl;

cout << " f2 = " << f2 << endl;

cout << " f3 = " << f3 << endl;

cout << " f4 = " << f4 << endl;

cout << "STEP5" << endl;

cout << " f4[0] = " << f4[0] << endl;

cout << " f4[1] = " << f4[1] << endl;

}

结果

