Text

Description automatically generated

Main.cpp

// Author: Diego Garcia, Brianna Sorianno, Kenry Yu

// Demo Time: 5:45 PM

#include "Fraction.h"

#include <iostream>

using namespace std;

int main() {

Fraction fract1;

Fraction fract2(14, 21);

Fraction fract3(11, -8);

Fraction fract4(fract3);

Fraction fract5(2, 0);

// Printing the object

cout << "Printing four fractions after constructed: " << endl;

cout << "fract1: ";

fract1.print();

cout << "fract2: ";

fract2.print();

cout << "fract3: ";

fract3.print();

cout << "fract4: ";

fract4.print();

cout << "fract5: ";

fract5.print();

// Using mutators

cout << "Changing the first two fractions and printing them:";

cout << endl;

fract1.setNumer(4);

cout << "fract1: ";

fract1.print();

fract2.setDenom(-5);

cout << "fract2: ";

fract2.print();

// Using accessors

cout << "Testing the changes in two fractions:" << endl;

cout << "fract1 numerator: " << fract1.getNumer() << endl;

cout << "fract2 denomerator: " << fract2.getDenom() << endl;

Fraction(1,3) + Fraction(1,6);

Fraction(1,2) \* Fraction(2,3);

return 0;

}

Fraction.h

class Fraction {

private:

int numer;

int denom;

int gcd(int, int);

public:

Fraction(int, int);

Fraction();

int getNumer();

int getDenom();

void print();

void setNumer(int);

void setDenom(int);

void operator+(Fraction);

void operator\*(Fraction);

};

Fraction.cpp

#include "Fraction.h"

#include <iostream>

int Fraction::gcd(int num, int den) {

if (num == 0 || den == 0)

return 1;

int small = num, large = den, gcde = 1;

if (large < small) {

small = den;

large = num;

}

for (int i = 1; i <= small; i++)

if (small % i == 0 && large % i == 0)

gcde = i;

return gcde;

}

Fraction::Fraction(int num, int den) {

if (den == 0) {

std::cout << "Denominator error\n";

this->numer = 0;

this->denom = 1;

} else {

// gcd of (num & den) divided by num

this->numer = num / gcd(num, den);

this->denom = den / gcd(num, den);

}

// if denon is less than 0:

if (this->denom < 0) {

this->numer \*= -1;

this->denom \*= -1;

}

}

Fraction::Fraction() {

this->numer = 0;

this->denom = 1;

}

int Fraction::getNumer() { return this->numer; }

int Fraction::getDenom() { return this->denom; }

void Fraction::setNumer(int num) { this->numer = num; }

void Fraction::setDenom(int den) {

this->denom = den;

if (this->denom < 0) {

this->numer \*= -1;

this->denom \*= -1;

}

}

void Fraction::print() {

std::cout << this->numer << "/" << this->denom << std::endl;

}

void Fraction::operator+(Fraction right) {

int num = (this->numer \* right.getDenom()) + (right.getNumer() \* this->denom);

int den = this->denom \* right.getDenom();

Fraction(num, den).print();

}

void Fraction::operator\*(Fraction right) {

int num = this->numer \* right.getNumer();

int den = this->denom \* right.getDenom();

Fraction(num, den).print();

}