Solución de Clase Fracciones -

Ma. Guadalupe Roque Díaz de León

TC1030 -SobreCarga de Operadores

Fracciones.hpp

```cpp
//  Created by Ma. Guadalupe Roque Díaz de León on 5/27/20.
//  Copyright © 2020 Invitado. All rights reserved.
//

#include "Fracciones.hpp"

#include <iostream>
#include <string>
using namespace std;

// Constructor default : 0/1  2/4 debe ser 1/2
Fracciones::Fracciones(){
    iNum = 0;
    iDen = 1;
}

// Constructor con parámetros :todos positivos
Fracciones::Fracciones(int num,int den){
    if (num >= 0 )
        iNum = num;
    else
        iNum = - num;
    if (den > 0)
        iDen = den;
    if (den < 0)
        iDen = - den;
}

void Fracciones::setNum(int num){
    iNum = num;
}

int Fracciones::getNum(){
    return iNum;
}

void Fracciones::setDen(int den){
    iDen = den;
}

int Fracciones::getDen(){
    return iDen;
}

string Fracciones::show()
{
    //std::cout << iNum << "/" << iDen << std::endl;
    return (to_string(iNum) + "/" + to_string(iDen));
}

double  Fracciones::strFlotante(){
    return  double(iNum) / iDen;
}

// Lee una fraccion
void Fracciones::input(){
    cout << "Ingresa el numerador:";
    cin >> iNum;
    cout << "Ingresa el denominador:";
    cin >> iDen;
}

// despliega la fraccion en la pantalla
void Fracciones::output(){
    std::cout << iNum << " / " << iDen << std::endl;
}

Fracciones & Fracciones::operator ++()
{
    cout << "++Prefix : " << endl;
    iNum++;
    iDen++;
    return  *this;
}

// Prefix and Postfix
Fracciones Fracciones::operator ++(int iF)
{
    cout << "Postfix++ :" << endl;
    Fracciones f1(iNum, iDen);
    iNum++; iDen++;
    return f1;
}

ostream & operator << (ostream &out, const Fracciones &f)
{
    out << f.iNum << endl;
    out << "__" << endl << f.iDen << endl;
    return out;
}

// funcion amiga - suma(f1,f2);
Fracciones suma(Fracciones f1, Fracciones f2){
    int iNum, iDen;

    if (f1.iDen == f2.iDen)
    {
        iNum = f1.iNum + f2.iNum;
        iDen  = f1.iDen ;
    }
    else
    {
        iNum = (f1.iNum * f2.iDen + f1.iDen * f2.iNum);
        iDen = (f1.iDen * f2.iDen);
    }

    Fracciones nuevo(iNum, iDen);
    return nuevo;
}
```

Fracciones.cpp

```
//
//  Created by Ma. Guadalupe Roque on 5/27/20.
//  Copyright © 2020 Invitado. All rights reserved.
//

#ifndef Fracciones_hpp
#define Fracciones_hpp

#include <string>
using namespace std;

#include <stdio.h>
class Fracciones{

private:
    int iNum, iDen;

public:
    // Constructor default : 0/1  2/4 debe ser 1/2
    Fracciones();
    // Constructor con parámetros :todos positivos
    Fracciones(int,int);
    // método set cambia el valor del numerador
    void setNum(int);
    // get retorna el valor del numerador
    int getNum();
    // set cambia el valor del denominador
    void setDen(int);
    // get retorna el valor del denominador
    int getDen();
    // str retorna iNum/iDen
    string show();
    // strPtoFlotante la forma en punto flotante
    double strFlotante();
    // Lee una fraccion
    void input();
    // despliega la fraccion
    void output();

    //void suma(Fracciones);  // Funcion miembro f1.suma(f2) ;

    // Funcion amiga  // suma(f1,f2)
    friend Fracciones suma(Fracciones, Fracciones);

    // - Sobrecarga de operadores aritméticos -
    // Sobrecargar el operador  + :Suma de dos Fracciones
    // f1 + f2
    Fracciones operator + (Fracciones f2){
        Fracciones resp;
        // Si son iguales los denominadores -
        if (iDen == f2.iDen)
        {
            resp.iNum = iNum + f2.iNum;
            resp.iDen = iDen ;
        }
        else
        {
            resp.iNum = (iNum * f2.iDen + iDen * f2.iNum);
            resp.iDen = (iDen * f2.iDen);
        }
        return resp;
    }
```

```
    // Sobrecargar el operador  - : Resta de dos Fracciones
    Fracciones operator-(Fracciones f2){
        Fracciones resp;
        if (iDen == f2.iDen)
        {
            resp.iNum = (iNum - f2.iNum);
            resp.iDen = iDen ;
        }
        else
        {
            resp.iNum = (iNum * f2.iDen - iDen * f2.iNum);
            resp.iDen = (iDen * f2.iDen);
        }
        return resp;
    }


    // Sobrecargar el operador  * : multiplica de dos Fracciones
    Fracciones operator *(Fracciones f2){
        Fracciones resp;
        // numerador X numerador y denominador X denominador
        resp.iNum = (iNum * f2.iNum);
        resp.iDen = iDen * f2.iDen;

        return resp;
    }

    // Sobrecargar el operador  / : multiplica de dos Fracciones
    Fracciones operator/(Fracciones f2){
        Fracciones resp;

        resp.setNum(iNum * f2.iDen);
        resp.setDen(iDen * f2.iNum);

        return resp;
    }

    // Sobrecargar el operador  < :overloaded < operator
    bool operator <(Fracciones f2) {

        if( ( double(iNum) / f2.iNum) < ( double(iDen) / f2.iDen) ) {
            // si hace este return termina la ejecucion de la funcion
            return true;
        }

        return false;
    }

    // Sobrecargar el operador  > :overloaded < operator
    bool operator > (Fracciones f2) {

        if(iNum > f2.iNum && iDen > f2.iDen) {
            // si hace este return termina la ejecucion de la funcion
            return true;
        }

        return false;
    }
```

```


     // Sobrecargar el operador  == :overloaded == operator
    bool operator == (Fracciones f2) {

        if(iNum == f2.iNum && iDen == f2.iDen) {
            // si hace este return termina la ejecucion de la funcion
            return true;
        }

        return false;
    }

    // Sobrecargar el operador  == :overloaded == operator
    Fracciones  & operator ++();

    // Sobrecargar el operador  == :overloaded == operator
    Fracciones operator ++(int);

    // friend   Sobrecargar el operador  == :overloaded == operator
    friend ostream & operator << (ostream &out, const Fracciones &f);
};
#endif /* Fracciones_hpp */
```

main.cpp

```cpp
//  main.cpp
//  Revision
//
//  Created by Ma. Guadalupe Roque Díaz de León on 5/27/20.
//  Copyright © 2020 Invitado. All rights reserved.
//

#include <stdio.h>
// Example program
#include <iostream>
#include "Fracciones.hpp"

using namespace std;

int menu(){
    int iOpcion;
    cout << "\n1. Ingresa la Fraccion1:";
    cout << "\n2. Ingresa la fraccion2:";
    cout << "\n3. f1 + f2 ";
    cout << "\n4. f1 - f2 ";
    cout << "\n5. f1 * f2 ";
    cout << "\n6. f1 / f2 ";
    cout << "\n7. f1 < f2 ";
     cout << "\n8. f1++ ";
     cout << "\n9. ++f1 ";
     cout << "\n10. cout << fraccion f1";
    cout << "\n11. cout << fraccion f2";
     cout << "\n12. funcion amiga suma(f1,f2)";
    cout << "\n0. Salir";
    cout << "\nTeclea la opcion:";
    cin >> iOpcion;
    return iOpcion;
}

int main() {
    Fracciones f1(3,5), f2(1,4);
    Fracciones f3(3,5), f4(2,6);
    Fracciones resultado;
    int iOpcion;

    // usando la sobrecarga del operador +
    resultado = f1 + f2;

    cout << f1.show() << " + " << f2.show() << " = " << resultado.show() << endl;

    resultado = f3 + f4;
    cout << f3.show() << " + " << f4.show() << " = " << resultado.show() << endl;

    // Inicializar la vcc antes del ciclo
    iOpcion = menu();
    while (iOpcion != 0) {
        switch (iOpcion) {
            case 1:
                f1.input();
                cout << f1 << endl;
                break;

            case 2:
                f2.input();
                f2.output();
                break;
            case 3: // +
                resultado = f1 + f2;
                cout << f1.show() << " + " << f2.show() << "= " << resultado.show() <<  "= " << resultado.strFlotante()<< endl;
                break;
            case 4: // -
                resultado = f1 - f2;
                cout << f1.show() << " - " << f2.show() << "= " << resultado.show() <<  "= " << resultado.strFlotante()<< endl;
                break;
            case 5: // *
                resultado = f1 * f2;
                cout << f1.show() << " * " << f2.show() << "= " << resultado.show() <<  "= " << resultado.strFlotante()<< endl;
                break;
            case 6: // /
                resultado = f1 / f2;
                cout << f1.show() << " / " << f2.show() << "= " << resultado.show() <<  "= " << resultado.strFlotante()<< endl;
                break;
             case 7: // <
                // Probando la sobrecarga del operador relacional <
                if (f1 == f2){
                    cout <<f2.show() <<  " <  " << f1.show() << endl;
                }
                else
                    cout << f1 <<  " > " << f2  << endl;
                break;
            case 8: // sobrecarga ++ posfijo
                cout << f1++ ;
                // ver el valor actualizado de f1
                cout << f1 ;
                break;
            case 9:// ++ prefijo
                 cout << ++f1;
                break;
            case 10: // sobrecarga del operador << sobre f1
                cout << f1;
                 break;
            case 11:// sobrecarga del operador << sobre f2
                cout << f2;
                 break;
            case 12: // llamada a la función amiga con sobrecarga
                cout << f1 + f2 << endl; // f1.suma(f2)
                cout << suma(f1,f2) << endl;
                break;
            default:
                cout << "Opcion Incorrecta!! Adios!!\n";
                break;
        }
        // Actualizar la vcc dentro del ciclo
        iOpcion = menu();
    }// fin while

    return 0;
}
```