

Solución de Clase Fracciones -

Ma. Guadalupe Roque Díaz de León
TC1030 - **SobreCarga de Operadores**



Fracciones.hpp

```

6 // Created by Ma. Guadalupe Roque Díaz de León on
7 // 5/27/20.
8 // Copyright © 2020 Invitado. All rights reserved.
9 //
10 #include "Fracciones.hpp"
11 #include <iostream>
12 #include <string>
13 using namespace std;
14
15 // Constructor default : 0/1 2/4 debe ser 1/2
16 Fracciones::Fracciones(){
17     iNum = 0;
18     iDen = 1;
19 }
20
21 // Constructor con parámetros : todos positivos
22 Fracciones::Fracciones(int num, int den){
23     if (num >= 0 )
24         iNum = num;
25     else
26         iNum = - num;
27     if (den > 0)
28         iDen = den;
29     if (den < 0)
30         iDen = - den;
31 }
32
33 void Fracciones::setNum(int num){
34     iNum = num;
35 }
36
37 int Fracciones::getNum(){
38     return iNum;
39 }
40
41 void Fracciones::setDen(int den){
42     iDen = den;
43 }
44
45 int Fracciones::getDen(){
46     return iDen;
47 }
48
49 string Fracciones::show()
50 {
51     //std::cout << iNum << "/" << iDen << std::endl;
52     return (to_string(iNum) + "/" + to_string(iDen));
53 }
54
55 double Fracciones::strFlotante(){
56     return double(iNum) / iDen;
57 }
58
59 // Lee una fraccion
60 void Fracciones::input(){
61     cout << "Ingresa el numerador:";
62     cin >> iNum;
63     cout << "Ingresa el denominador:";
64     cin >> iDen;
65 }

```

```

66 // despliega la fraccion en la pantalla
67 void Fracciones::output(){
68     std::cout << iNum << " / " << iDen << std::endl;
69 }
70
71 Fracciones & Fracciones::operator ++()
72 {
73     cout << "++Prefix : " << endl;
74     iNum++;
75     iDen++;
76     return *this;
77 }
78
79 // Prefix and Postfix
80 Fracciones Fracciones::operator ++(int iF)
81 {
82     cout << "Postfix++ : " << endl;
83     Fracciones f1(iNum, iDen);
84     iNum++; iDen++;
85     return f1;
86 }
87
88 ostream & operator << (ostream &out, const Fracciones &f)
89 {
90     out << f.iNum << endl;
91     out << "___" << endl << f.iDen << endl;
92     return out;
93 }
94
95 // funcion amiga - suma(f1,f2);
96 Fracciones suma(Fracciones f1, Fracciones f2){
97     int iNum, iDen;
98
99     if (f1.iDen == f2.iDen)
100     {
101         iNum = f1.iNum + f2.iNum;
102         iDen = f1.iDen ;
103     }
104     else
105     {
106         iNum = (f1.iNum * f2.iDen + f1.iDen * f2.iNum);
107         iDen = (f1.iDen * f2.iDen);
108     }
109
110     Fracciones nuevo(iNum, iDen);
111     return nuevo;
112 }
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127

```



Fracciones.cpp

```

1 //
2 // Created by Ma. Guadalupe Roque on 5/27/20.
3 // Copyright © 2020 Invitado. All rights reserved.
4 //
5
6 #ifndef Fracciones_hpp
7 #define Fracciones_hpp
8
9 #include <string>
10 using namespace std;
11
12 #include <stdio.h>
13 class Fracciones{
14 private:
15     int iNum, iDen;
16
17 public:
18     // Constructor default : 0/1 2/4 debe ser 1/2
19     Fracciones();
20     // Constructor con parámetros : todos positivos
21     Fracciones(int,int);
22     // método set cambia el valor del numerador
23     void setNum(int);
24     // get retorna el valor del numerador
25     int getNum();
26     // set cambia el valor del denominador
27     void setDen(int);
28     // get retorna el valor del denominador
29     int getDen();
30     // str retorna iNum/iDen
31     string show();
32     // strPtoFlotante la forma en punto flotante
33     double strFlotante();
34     // Lee una fraccion
35     void input();
36     // despliega la fraccion
37     void output();
38
39     //void suma(Fracciones); // Funcion miembro f1.suma(f2) ;
40
41     // Funcion amiga // suma(f1,f2)
42     friend Fracciones suma(Fracciones, Fracciones);
43
44     // - Sobrecarga de operadores aritméticos -
45     // Sobrecargar el operador + : Suma de dos Fracciones
46     // f1 + f2
47     Fracciones operator + (Fracciones f2){
48         Fracciones resp;
49         // Si son iguales los denominadores -
50         if (iDen == f2.iDen)
51         {
52             resp.iNum = iNum + f2.iNum;
53             resp.iDen = iDen ;
54         }
55         else
56         {
57             resp.iNum = (iNum * f2.iDen + iDen * f2.iNum);
58             resp.iDen = (iDen * f2.iDen);
59         }
60         return resp;
61     }
62 }
63

```

```

64
65 // Sobrecargar el operador - : Resta de dos Fracciones
66 Fracciones operator-(Fracciones f2){
67     Fracciones resp;
68     if (iDen == f2.iDen)
69     {
70         resp.iNum = (iNum - f2.iNum);
71         resp.iDen = iDen ;
72     }
73     else
74     {
75         resp.iNum = (iNum * f2.iDen - iDen * f2.iNum);
76         resp.iDen = (iDen * f2.iDen);
77     }
78     return resp;
79 }
80
81 // Sobrecargar el operador * : multiplica de dos Fracciones
82 Fracciones operator *(Fracciones f2){
83     Fracciones resp;
84     // numerador X numerador y denominador X denominador
85     resp.iNum = (iNum * f2.iNum);
86     resp.iDen = iDen * f2.iDen;
87
88     return resp;
89 }
90
91 // Sobrecargar el operador / : multiplica de dos Fracciones
92 Fracciones operator/(Fracciones f2){
93     Fracciones resp;
94
95     resp.setNum(iNum * f2.iDen);
96     resp.setDen(iDen * f2.iNum);
97
98     return resp;
99 }
100
101 // Sobrecargar el operador < : overloaded < operator
102 bool operator <(Fracciones f2) {
103
104     if( ( double(iNum) / f2.iNum) < ( double(iDen) / f2.iDen) ) {
105         // si hace este return termina la ejecucion de la funcion
106         return true;
107     }
108
109     return false;
110 }
111
112 // Sobrecargar el operador > : overloaded < operator
113 bool operator > (Fracciones f2) {
114
115     if(iNum > f2.iNum && iDen > f2.iDen) {
116         // si hace este return termina la ejecucion de la funcion
117         return true;
118     }
119
120     return false;
121 }
122
123
124
125
126
127
128

```

```

129
130 // Sobrecargar el operador == : overloaded == operator
131 bool operator == (Fracciones f2) {
132
133     if(iNum == f2.iNum && iDen == f2.iDen) {
134         // si hace este return termina la ejecucion de la funcion
135         return true;
136     }
137
138     return false;
139 }
140
141 // Sobrecargar el operador == : overloaded == operator
142 Fracciones & operator ++();
143
144 // Sobrecargar el operador == : overloaded == operator
145 Fracciones operator ++(int);
146
147 // friend Sobrecargar el operador == : overloaded == operator
148 friend ostream & operator << (ostream &out, const Fracciones &f);
149 };
150 #endif /* Fracciones_hpp */
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188

```



main.cpp

```

1 // main.cpp
2 // Revision
3 //
4 //
5 // Created by Ma. Guadalupe Roque Díaz de León on 5/27/20.
6 // Copyright © 2020 Invitado. All rights reserved.
7 //
8
9 #include <stdio.h>
10 // Example program
11 #include <iostream>
12 #include "Fracciones.hpp"
13
14 using namespace std;
15
16 int menu(){
17     int iOpcion;
18     cout << "\n1. Ingrese la Fraccion1:";
19     cout << "\n2. Ingrese la fraccion2:";
20     cout << "\n3. f1 + f2 ";
21     cout << "\n4. f1 - f2 ";
22     cout << "\n5. f1 * f2 ";
23     cout << "\n6. f1 / f2 ";
24     cout << "\n7. f1 < f2 ";
25     cout << "\n8. f1++ ";
26     cout << "\n9. ++f1 ";
27     cout << "\n10. cout << fraccion f1";
28     cout << "\n11. cout << fraccion f2";
29     cout << "\n12. funcion amiga suma(f1,f2)";
30     cout << "\n0. Salir";
31     cout << "\nTeclea la opcion:";
32     cin >> iOpcion;
33     return iOpcion;
34 }
35
36 int main() {
37     Fracciones f1(3,5), f2(1,4);
38     Fracciones f3(3,5), f4(2,6);
39     Fracciones resultado;
40     int iOpcion;
41
42     // usando la sobrecarga del operador +
43     resultado = f1 + f2;
44
45     cout << f1.show() << " + " << f2.show() << " = " << resultado.show() << endl;
46
47     resultado = f3 + f4;
48     cout << f3.show() << " + " << f4.show() << " = " << resultado.show() << endl;
49
50     // Inicializar la vcc antes del ciclo
51     iOpcion = menu();
52     while (iOpcion != 0) {
53         switch (iOpcion) {
54             case 1:
55                 f1.input();
56                 cout << f1 << endl;
57                 break;
58
59             case 2:
60                 f2.input();
61                 f2.output();
62                 break;

```

```

65         case 3: // +
66             resultado = f1 + f2;
67             cout << f1.show() << " + " << f2.show() << " = " << resultado.show() << " = " << resultado.strFlotante() << endl;
68             break;
69         case 4: // -
70             resultado = f1 - f2;
71             cout << f1.show() << " - " << f2.show() << " = " << resultado.show() << " = " << resultado.strFlotante() << endl;
72             break;
73         case 5: // *
74             resultado = f1 * f2;
75             cout << f1.show() << " * " << f2.show() << " = " << resultado.show() << " = " << resultado.strFlotante() << endl;
76             break;
77         case 6: // /
78             resultado = f1 / f2;
79             cout << f1.show() << " / " << f2.show() << " = " << resultado.show() << " = " << resultado.strFlotante() << endl;
80             break;
81         case 7: // <
82             // Probando la sobrecarga del operador relacional <
83             if (f1 == f2){
84                 cout << f2.show() << " < " << f1.show() << endl;
85             }
86             else
87                 cout << f1 << " > " << f2 << endl;
88             break;
89         case 8: // sobrecarga ++ posfijo
90             cout << f1++ ;
91             // ver el valor actualizado de f1
92             cout << f1 ;
93             break;
94         case 9:// ++ prefijo
95             cout << ++f1;
96             break;
97         case 10: // sobrecarga del operador << sobre f1
98             cout << f1;
99             break;
100         case 11:// sobrecarga del operador << sobre f2
101             cout << f2;
102             break;
103         case 12: // llamada a la función amiga con sobrecarga
104             cout << f1 + f2 << endl; // f1.suma(f2)
105             cout << suma(f1,f2) << endl;
106             break;
107         default:
108             cout << "Opcion Incorrecta!! Adios!!\n";
109             break;
110     }
111     // Actualizar la vcc dentro del ciclo
112     iOpcion = menu();
113 } // fin while
114
115 return 0;
116 }
117
118
119

```