

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 class Rational
5 {
6 private:
7     int numer;
8     int denom;
9 public:
10    int getNumer() const;
11    int getDenom() const;
12    void setNumer(int);
13    void setDenom(int);
14    void input();
15    void output() const;
16    Rational();
17    Rational(int, int = 1);
18    void reduce();
19    Rational friend operator+(const Rational& a, const Rational& b);
20 };
21 void Rational::reduce()
22 {
23     int x = abs(numer);
24     int y = abs(denom);
25     // find minimum of x and y
26     int min = x;
27     if (y < x)
28         min = y;
29
30     // finding a common factor greater than 1
31     int gcf = 1;
32     for (int i = 2; i <= min; i++) {
33         if (x % i == 0 && y % i == 0) {
34             gcf = i;
35         }
36     }
37     numer = numer / gcf;
38     denom = denom / gcf;
39     if (denom < 0)
40     {
41         numer = -numer;
42         denom = -denom;
43     }
44 }
45 Rational::Rational()
46 {
47     numer = 0;
48     denom = 1;
49 }
```

```
50 Rational::Rational(int x, int y)
51 {
52     numer = x;
53     if (y != 0)
54         denom = y;
55     else
56         denom = 1;
57     reduce();
58 }
59 int Rational::getNum() const
60 {
61     return numer;
62 }
63 int Rational::getDenom() const
64 {
65     return denom;
66 }
67 void Rational::setNumer(int x)
68 {
69     numer = x;
70     reduce();
71 }
72 void Rational::setDenom(int x)
73 {
74     denom = x;
75     if (denom == 0)
76         denom = 1;
77     reduce();
78 }
79 void Rational::input()
80 {
81     cout << "Numerator? ";
82     cin >> numer;
83     cout << "Denominator? ";
84     cin >> denom;
85     while (denom == 0)
86     {
87         cout << "Denominator can't be zero!\n";
88         cout << "Denominator? ";
89         cin >> denom;
90     }
91     reduce();
92 }
93 void Rational::output() const
94 {
95     if (denom != 1)
96         cout << numer << "/" << denom << endl;
97     else
98         cout << numer << endl;
```

```
99 }
100 Rational operator+(const Rational &a, const Rational &b)
101 {
102     Rational c;
103     c.setNumerator(a.getNumerator() * b.getDenominator() + a.getDenominator() * b.getNumerator());
104     c.setDenominator(a.getDenominator() * b.getDenominator());
105     c.reduce();
106     return c;
107 }
108 Rational operator-(const Rational& a, const Rational& b)
109 {
110     int x = a.getNumerator() * b.getDenominator() - a.getDenominator() * b.getNumerator();
111     int y = a.getDenominator() * b.getDenominator();
112     return Rational(x,y);
113 }
114 Rational operator*(const Rational& a, const Rational& b)
115 {
116     Rational c;
117     c.setNumerator(a.getNumerator() * b.getNumerator());
118     c.setDenominator(a.getDenominator() * b.getDenominator());
119     c.reduce();
120     return c;
121 }
122 Rational operator/(const Rational& a, const Rational& b)
123 {
124     Rational c;
125     c.setNumerator(a.getNumerator() * b.getDenominator());
126     c.setDenominator(a.getDenominator() * b.getNumerator());
127     c.reduce();
128     return c;
129 }
130 void operator+=(Rational& a, const Rational& b)
131 {
132     a = a + b;
133 }
134 void operator-=(Rational& a, const Rational& b)
135 {
136     Rational c;
137     c.setNumerator(a.getNumerator() * b.getDenominator() - a.getDenominator() * b.getNumerator());
138     c.setDenominator(a.getDenominator() * b.getDenominator());
139     c.reduce();
140     a = c;
141 }
142 void operator*=(Rational& a, const Rational& b)
143 {
144     Rational c;
145     c.setNumerator(a.getNumerator() * b.getNumerator());
146     c.setDenominator(a.getDenominator() * b.getDenominator());
147     c.reduce();
```

```
148     a = c;
149 }
150 void operator/=(Rational& a, const Rational& b)
151 {
152     Rational c;
153     c.setNumer(a.getNumer() * b.getDenom());
154     c.setDenom(a.getDenom() * b.getNumer());
155     c.reduce();
156     a = c;
157 }
158 bool operator<(const Rational& a, const Rational& b)
159 {
160     return (a.getNumer() * b.getDenom()) < (a.getDenom() * b.getNumer());
161 }
162 bool operator<=(const Rational& a, const Rational& b)
163 {
164     return (a.getNumer() * b.getDenom()) <= (a.getDenom() * b.getNumer());
165 }
166 bool operator>(const Rational& a, const Rational& b)
167 {
168     return (a.getNumer() * b.getDenom()) > (a.getDenom() * b.getNumer());
169 }
170 bool operator>=(const Rational& a, const Rational& b)
171 {
172     return (a.getNumer() * b.getDenom()) >= (a.getDenom() * b.getNumer());
173 }
174 bool operator==(const Rational& a, const Rational& b)
175 {
176     return (a.getNumer() * b.getDenom()) == (a.getDenom() * b.getNumer());
177 }
178 bool operator!=(const Rational& a, const Rational& b)
179 {
180     return (a.getNumer() * b.getDenom()) != (a.getDenom() * b.getNumer());
181 }
182 Rational operator++(Rational& a) // prefix ++x
183 {
184     a.setNumer(a.getNumer() + a.getDenom());
185     return a;
186 }
187 Rational operator++(Rational& a, int n) // postfix x++
188 {
189     Rational b = a;
190     a.setNumer(a.getNumer() + a.getDenom());
191     return b;
192 }
193
194 int main()
195 {
196     Rational a(1, 7), b(1);
```

---

```
197     a += b;
198     a.output();
199     b.output();
200
201
202     return 0;
203 }
```