```cpp
//Sources:complex.cpp
#include "complex.h"
#include <iostream>
#include <cmath>

bool complex::operator==(const complex & other)
{
/*
    一般我们比较两个对象是否相等
    会比较对象里的所有变量是否相等
    若相等则认为两个对象相等
    布尔值是c++关键字
*/
    return ( (real==other.real) && (imag==other.imag) );
}

complex operator+(float a, complex c)
{
/*
    这是非类里的函数，不需要类解析符
    但是访问类里私有变量因此要在类里加友元并声明
    重载了加号+运算符
    complex a=(float)c+(complex)b
    相当于 a = c.operator+(b);
    对一个复数实部进行加运算，虚部不变
*/
    complex temp;
    temp.real = a + c.real;
    temp.imag = c.imag;
    return temp;
}

complex operator-(float a, complex c)
{
/*
    这是非类里的函数，不需要类解析符
    但是访问类里私有变量因此要在类里加友元并声明
    重载了减号-运算符
    complex a=(float)c-(complex)b
    相当于 a = c.operator-(b);
    对一个复数实部进行减运算，虚部不变
*/
    complex temp;
    temp.real = a - c.real;
    temp.imag = c.imag;
    return temp;
}

complex operator*(float a, complex c)
{
/*
    这是非类里的函数，不需要类解析符
    但是访问类里私有变量因此要在类里加友元并声明
    重载了乘号*运算符
    complex a=(float)c*(complex)b
    相当于 a = c.operator*(b);
    对一个复数实部进行乘运算，虚部和实部分别相乘
*/
    complex temp;
    temp.real = a * c.real;
    temp.imag = a * c.imag;
    return temp;
}

complex operator/(float a, complex c)
{
/*
    这是非类里的函数，不需要类解析符
    但是访问类里私有变量因此要在类里加友元并声明
    重载了除号/运算符
    complex a=(float)c/(complex)b
    相当于 a = c.operator/(b);
    对一个复数实部进行除运算，虚部和实部分别相除
*/
    if(c.real==0&&c.imag==0)
    {
        std::cout<<"Input Error!"<<std::endl;
    }
    else
    {
        complex temp;
        temp.real = a / c.real;
        temp.imag = a / c.imag;
        return temp;
```

```
85          }
86      }
87
88
89      std::ostream &operator<<(std::ostream &output, complex c)
90      {
91      /*
92          重载了重定向<<运算符
93          重定向到标准输出流，由于要改变值，所以做引用传递
94      */
95          output<<"The complex is "<<c.real<<" + "<<c.imag<<"i"<<std::endl;
96          output<<"The complex 's real part is "<<c.real<<" ,its imag part is
    "<<c.imag<<std::endl;
97          return output;
98      }
99      std::istream &operator>>(std::istream &input, complex &c)
100     {
101     /*
102         重载了重定向>>运算符
103         重定向到标准输入流，由于要改变值，所以做引用传递
104     */
105         std::cout << "Please enter the real part:";
106         input >> c.real;
107         std::cout << "please enter the imaginery part:";
108         input >> c.imag;
109         return input;
110     }
111
112     complex complex::operator+(complex c)
113     {
114     /*
115         重载了加号+运算符
116         complex a=(complex)b+(complex)c
117         相当于 a = b.operator+(c);
118         对两个复数进行加运算
119     */
120         complex temp;
121         temp.real = real + c.real;
122         temp.imag = imag + c.imag;
123         return temp;
124     }
125     complex complex::operator+(float c)
126     {
127     /*
128         重载了加号+运算符
129         complex a=(complex)b+(float)c
130         相当于 a = b.operator+(c);
131         对一个复数实部进行加运算，虚部不变
132     */
133         complex temp;
134         temp.real = real + c;
135         temp.imag = imag;
136         return temp;
137     }
138
139
140     complex complex::operator-(complex c)
141     {
142     /*
143         重载了减号-运算符
144         complex a=(complex)b-(complex)c
145         相当于 a = b.operator-(c);
146         对两个复数进行减运算
147     */
148         complex temp;
149         temp.real = real - c.real;
150         temp.imag = imag - c.imag;
151         return temp;
152     }
153     complex complex::operator-(float c)
154     {
155     /*
156         重载了减号-运算符
157         complex a=(complex)b-(float)c
158         相当于 a = b.operator-(c);
159         对一个复数实部进行减运算，虚部不变
160     */
161         complex temp;
162         temp.real = real - c;
163         temp.imag = imag;
164         return temp;
165     }
166
167     complex complex::operator*(complex c)
```

```
{
/*
    重载了乘号*运算符
    complex a=(complex)b*(complex)c
    相当于 a = b.operator*(c);
    对两个复数进行乘运算
    （a＋bi）*（c＋di）＝（ac—bd）＋（bc＋ad）i,
    a=real;
    b=imag;
    c=c.real;
    d=c.imag;
    temp.real=(real*c.real-imag*c.imag)
    temp.imag=(imag*c.real-real*c.imag)
*/
    if(c.real==0&&c.imag==0)
    {
        std::cout<<"Input Error!"<<std::endl;
    }
    else
    {
        complex temp;
        temp.real = (real*c.real-imag*c.imag);
        temp.imag = (imag*c.real-real*c.imag);
        return temp;
    }
}
complex complex::operator*(float c)
{
/*
    重载了乘号*运算符
    complex a=(complex)b*(float)c
    相当于 a = b.operator*(c);
    对一个复数实部进行乘运算，虚部和实部分别相乘
*/
    complex temp;
    temp.real = real * c;
    temp.imag = imag * c;
    return temp;
}

complex complex::operator/(complex c)
{
/*
    重载了除号/运算符
    complex a=(complex)b/(complex)c
    相当于 a = b.operator/(c);
    对两个复数进行/运算
    （a＋bi）/（c＋di）＝（（ac+bd）/（c^2+d^2））＋（（bc—ad）/（c^2+d^2））i,
    a=real;
    b=imag;
    c=c.real;
    d=c.imag;
    temp.real=(real*c.real-imag*c.imag)/(c.real*c.real+c.imag*c.imag);
    temp.imag=(imag*c.real-real*c.imag)/(c.real*c.real+c.imag*c.imag);
*/
    if(c.real==0&&c.imag==0)
    {
        std::cout<<"Input Error!"<<std::endl;
    }
    else
    {
        complex temp;
        temp.real = (real*c.real-imag*c.imag)/(c.real*c.real+c.imag*c.imag);
        temp.imag = (imag*c.real-real*c.imag)/(c.real*c.real+c.imag*c.imag);
        return temp;
    }
}
complex complex::operator/(float c)
{
/*
    重载了减号/运算符
    complex a=(complex)b/(float)c
    相当于 a = b.operator/(c);
    对一个复数实部进行除运算，虚部和实部分别相除
*/
    if(c==0)
    {
        std::cout<<"Input Error!"<<std::endl;
    }
    else
    {
        complex temp;
        temp.real = real / c;
        temp.imag = imag / c;
```

```
252             return temp;
253         }
254     }
255
256     float complex::abs()
257     {
258         return sqrt(real*real+imag*imag);
259     }
260
261     complex::complex()
262     {
263         real=0.0;
264         imag=0.0;
265     }
266
267     complex::complex(float r,float i)
268     {
269         real=r;
270         imag=i;
271     }
272
273     void complex::PrintComplex()
274     {
275         std::cout<<"The complex is ("<<real<<") + ("<<imag<<") i"<<std::endl;
276         std::cout<<"The complex 's real part is "<<real<<" ,its imag part is "<<imag<<std::endl;
277     }
278
279     void complex::setComplex(float r,float i)
280     {
281         real=r;
282         imag=i;
283     }
284
285     float complex::getImag()
286     {
287         return imag;
288     }
289
290     float complex::getReal()
291     {
292         return real;
293     }
294
295
```