

Step-by-step walkthrough for example on
page 15 – 16 of the lecture notes:
Revision Example, Pointer,
Reference & const-ness

```
class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x) // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x) // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	<input type="text"/>
imag	<input type="text"/>

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```
class Complex /* File: complex.h */  
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
{
```

```
    real += x.real; imag += x.imag;
```

```
    return (*this);
```

```
}
```

```
        Complex* add2(const Complex& x) // Return by value using pointer
```

```
{
```

```
    real += x.real; imag += x.imag;
```

```
    return this;
```

```
}
```

```
        Complex& add3(const Complex& x) // Return by reference
```

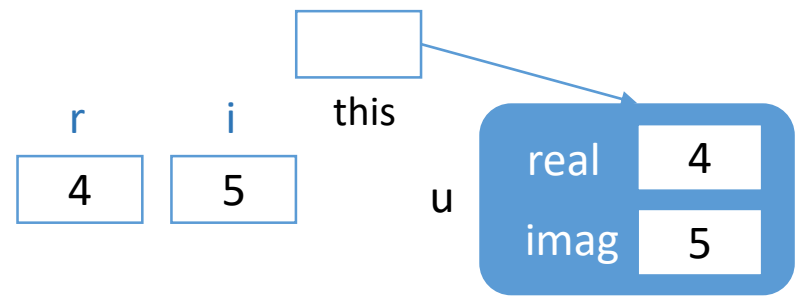
```
{
```

```
    real += x.real; imag += x.imag;
```

```
    return (*this);
```

```
}
```

```
};
```



```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x) // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x) // Return by reference
```

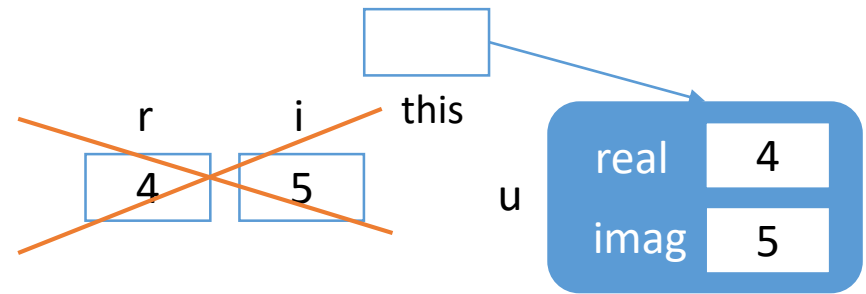
```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
};
```



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

a

real	4
imag	5

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
```

```
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

a

real	4
imag	5


```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x) // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x) // Return by reference
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
};
```

u

real	4
imag	5

a



this

real	4
imag	5

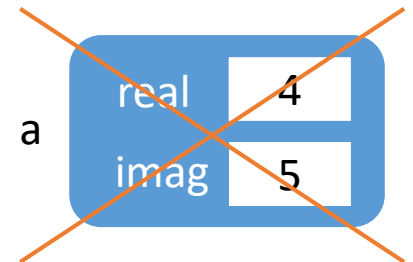
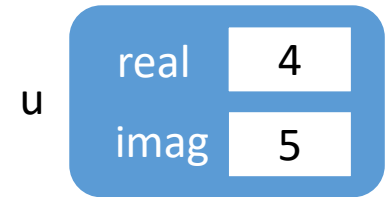
Output

(4 , 5)

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

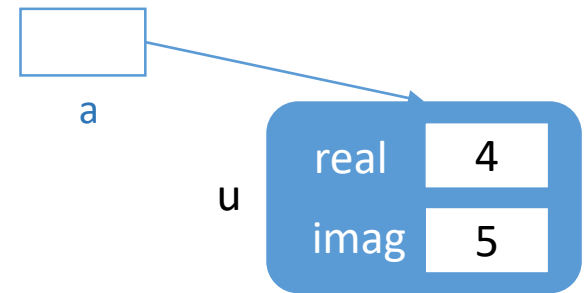
```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

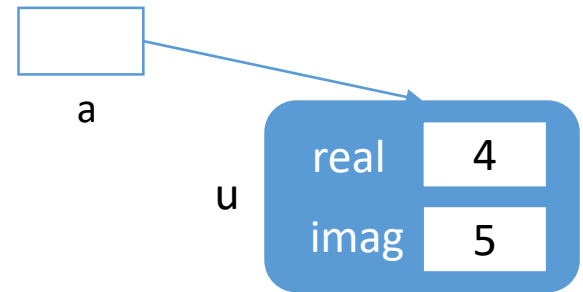
```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```



```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x) // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x) // Return by reference
```

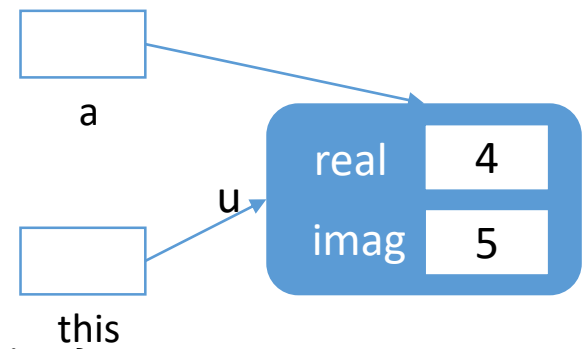
```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
};
```



Output

(4 , 5)

(4 , 5)

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

```



a

u



```

void f(const Complex a) { a.print(); }    // const Complex a = u
void g(const Complex* a) { a->print(); }  // const Complex* a = &u
void h(const Complex& a) { a.print(); }   // const Complex& a = u

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print();    a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print();    c.print();
    return 0;
}

```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

a	real	4
u	imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

a
u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

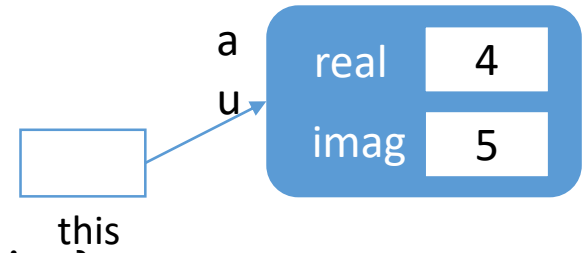
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x) // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x) // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```



A blue rounded rectangle represents a Complex object. Inside, there are two white boxes. The top box is labeled 'real' and contains the number '4'. The bottom box is labeled 'imag' and contains the number '5'. To the left of the rectangle, the letter 'u' is written, with a red 'X' over it, indicating that the variable 'u' is the object being passed.

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
```

```
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

w

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x) // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x) // Return by reference
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

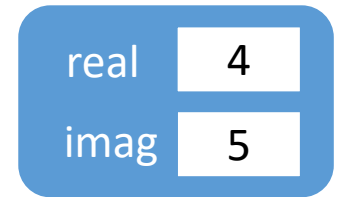
```
            return (*this);
```

```
        }
```

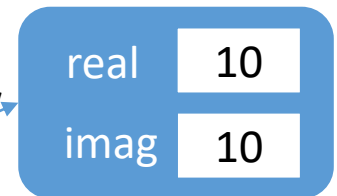
```
};
```



u



w



Output

(4 , 5)

(4 , 5)

(4 , 5)

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x) // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x) // Return by reference
```

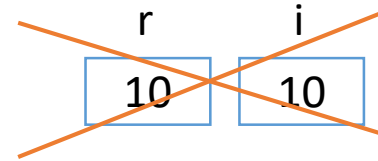
```
        {
```

```
            real += x.real; imag += x.imag;
```

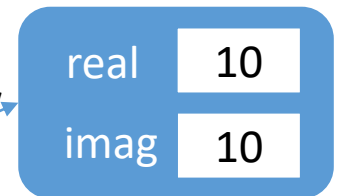
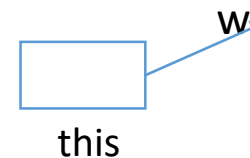
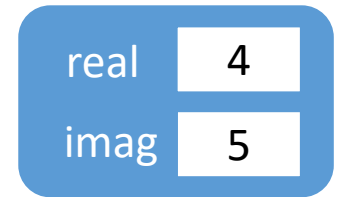
```
            return (*this);
```

```
        }
```

```
};
```



u



Output

(4 , 5)

(4 , 5)

(4 , 5)

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

Cursor here




```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
```

```
    return 0;
```

```
}
```

u

real	4
imag	5

w

real	10
imag	10

x

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return this;
```

```
    }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

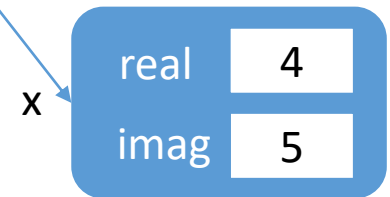
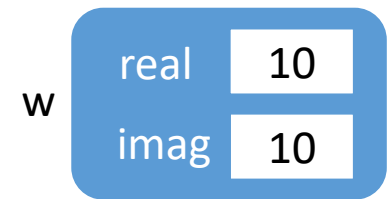
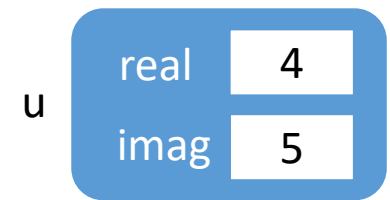
```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
};
```



Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return this;
```

```
    }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

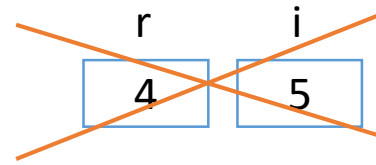
```
    {
```

```
        real += x.real; imag += x.imag;
```

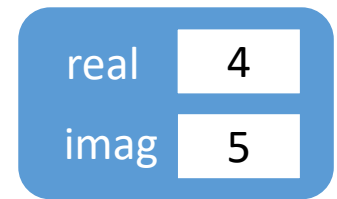
```
        return (*this);
```

```
    }
```

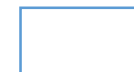
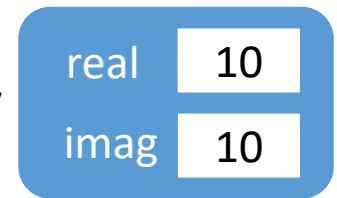
```
};
```



u

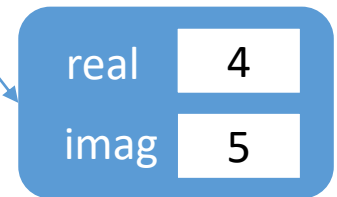


w



this

x



Output

(4 , 5)

(4 , 5)

(4 , 5)

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
```

```
}
```

x

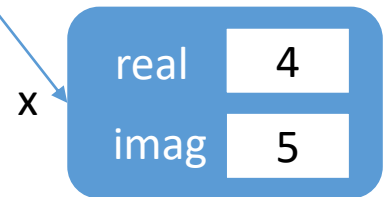
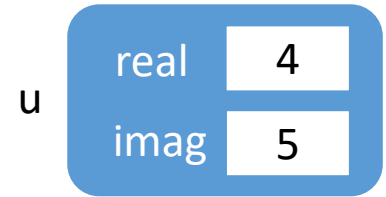
real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

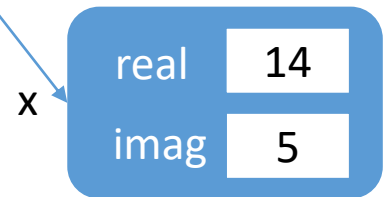
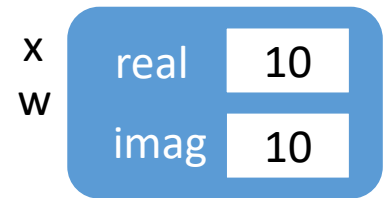
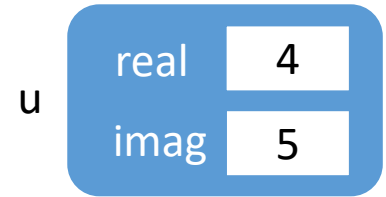
(4 , 5)
(4 , 5)
(4 , 5)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

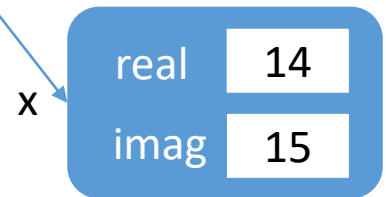
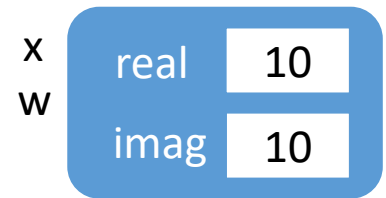
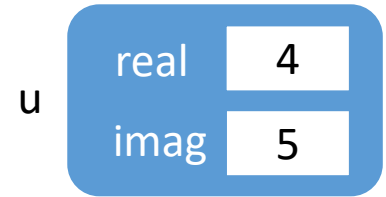
(4 , 5)
(4 , 5)
(4 , 5)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

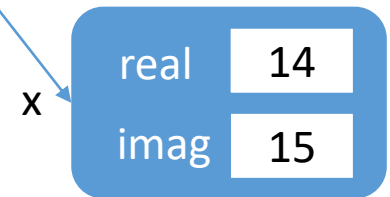
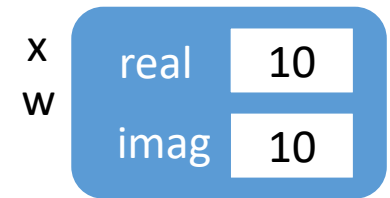
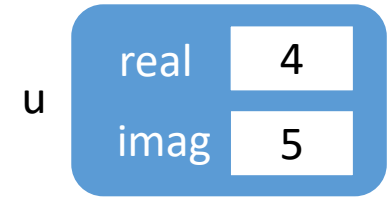
(4 , 5)
(4 , 5)
(4 , 5)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



*this is x,
Returning it by value, we need to construct a
temporary object "temp" which is a copy of x

Output

(4 , 5)
(4 , 5)
(4 , 5)


```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

This is temp

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	4
imag	5

temp

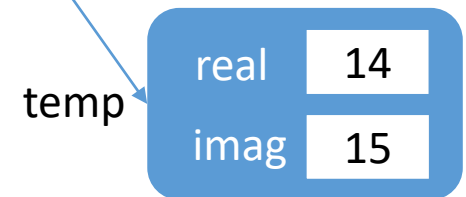
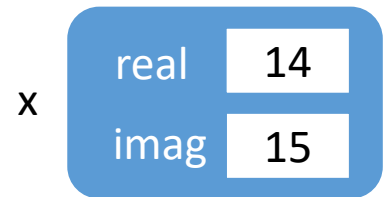
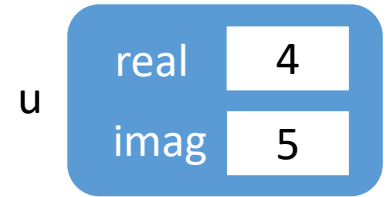
real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)



this

temp

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

This is temp

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	14
imag	15

temp

real	14
imag	15

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	14
imag	15

y

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return this;
```

```
    }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

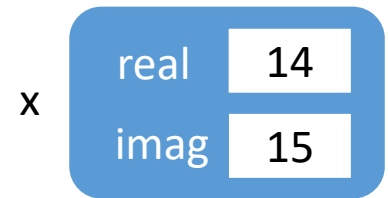
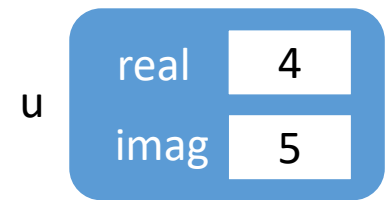
```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

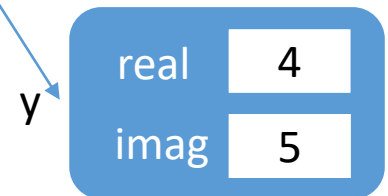
```
    }
```

```
};
```



Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```



```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return this;
```

```
    }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

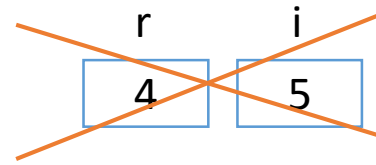
```
    {
```

```
        real += x.real; imag += x.imag;
```

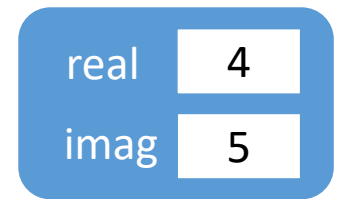
```
        return (*this);
```

```
    }
```

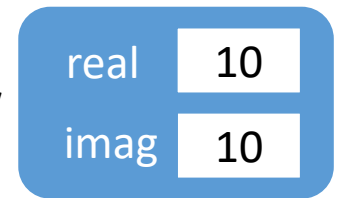
```
};
```



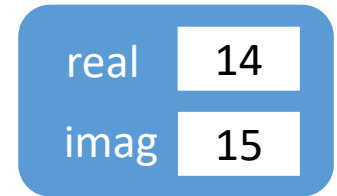
u



w



x

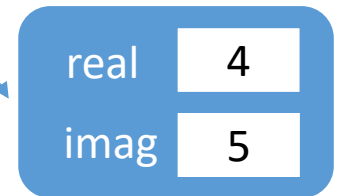


Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

this

y



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
```

```
}
```

x

real	14
imag	15

y

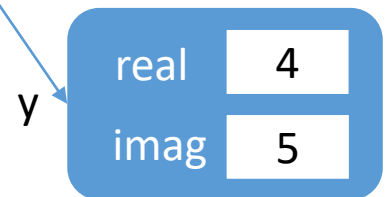
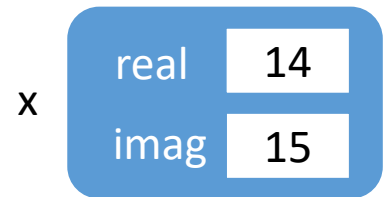
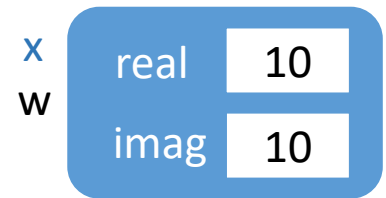
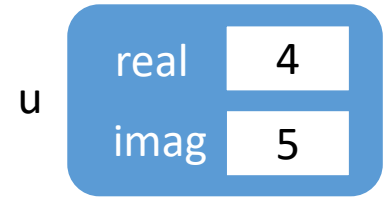
real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)

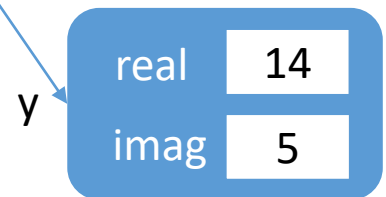
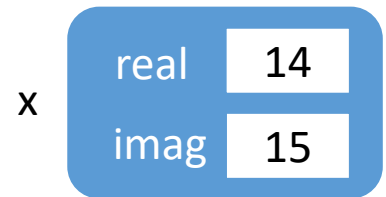
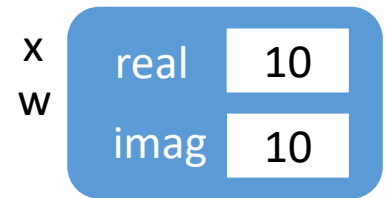
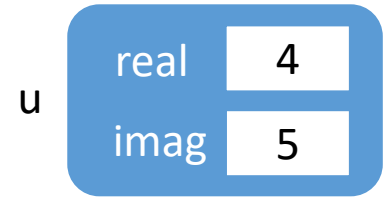



```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)

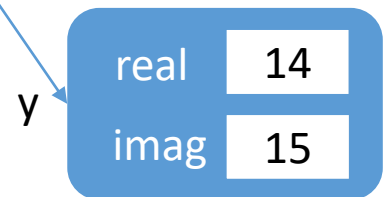
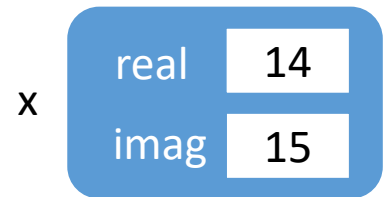
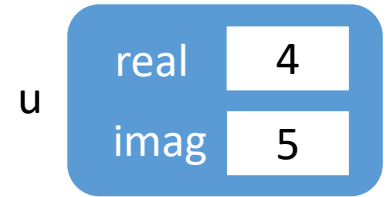


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)



this

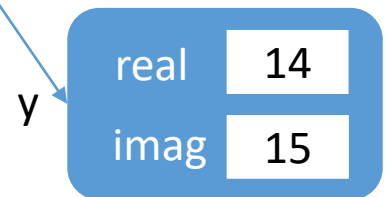
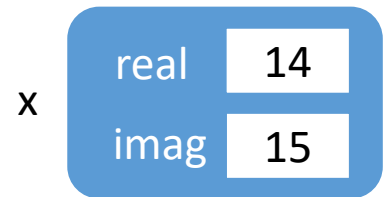
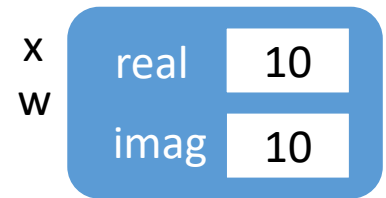
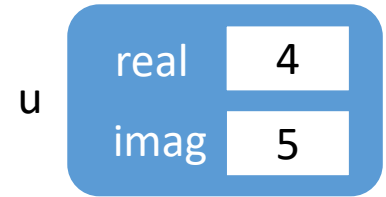


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)



this



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
```

```
{
```

This is &y, i.e. address of y

w

real	10
imag	10

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
```

```
    return 0;
```

```
}
```

x

real	14
imag	15

y

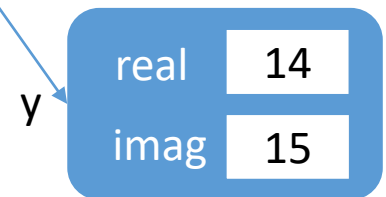
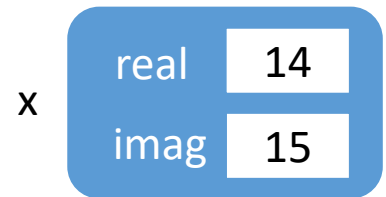
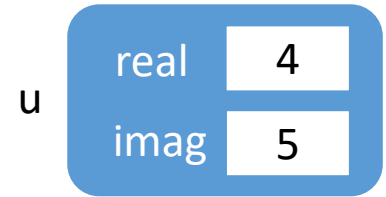
real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)



this

y

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	14
imag	15

y

real	14
imag	15

z

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

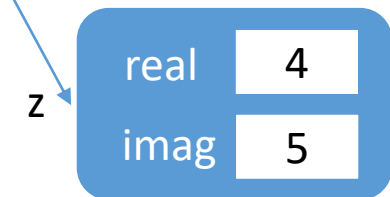
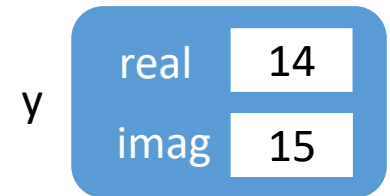
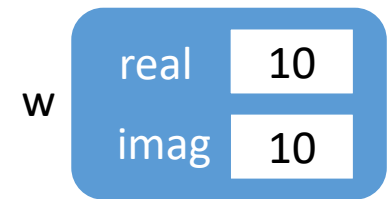
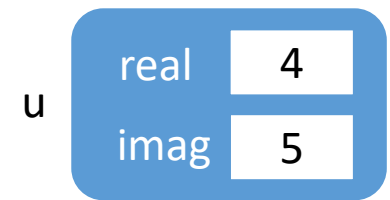
```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
};
```



Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
```



```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

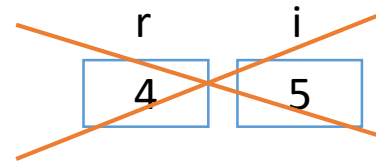
```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
};
```



u

real	4
imag	5

w

real	10
imag	10

x

real	14
imag	15

y

real	14
imag	15

z

real	4
imag	5

Output

(4 , 5)

(4 , 5)

(4 , 5)

(14 , 15)

(14 , 15)




```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	14
imag	15

y

real	14
imag	15

z

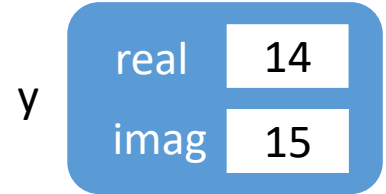
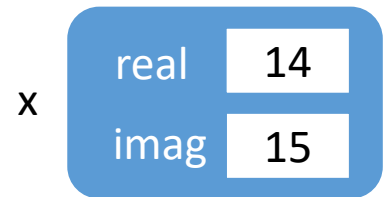
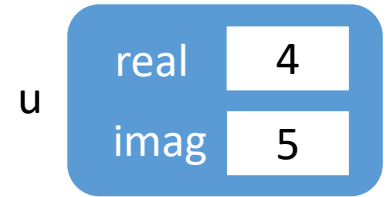
real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)

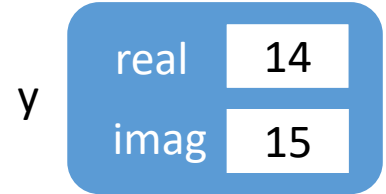
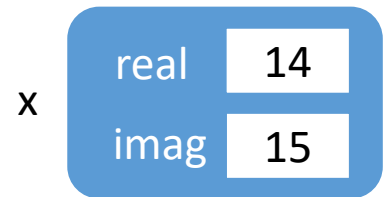
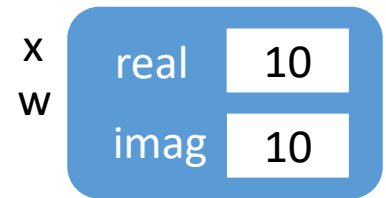
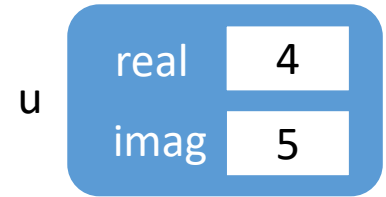


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)

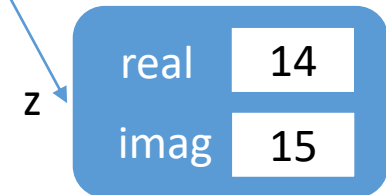
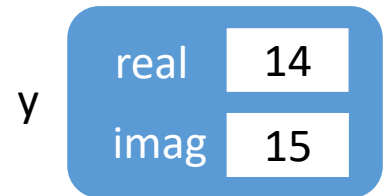
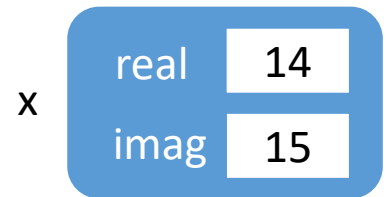
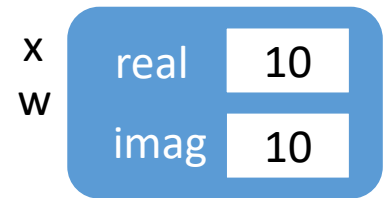
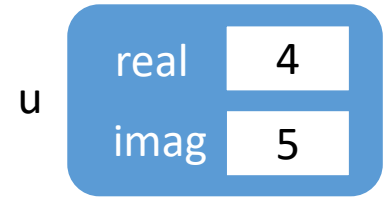


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)

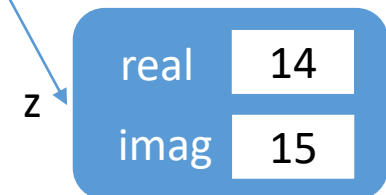
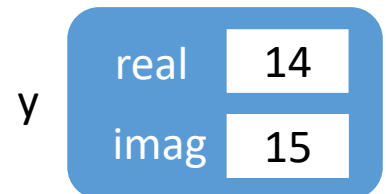
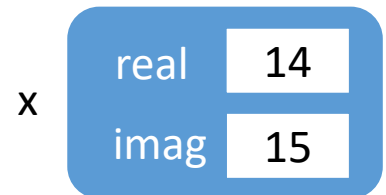
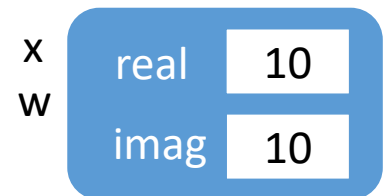
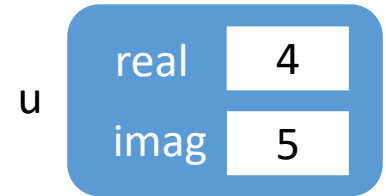


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)



*this is z,
Returning it by reference, i.e. returning z

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

u

real	4
imag	5

```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	14
imag	15

y

real	14
imag	15

z

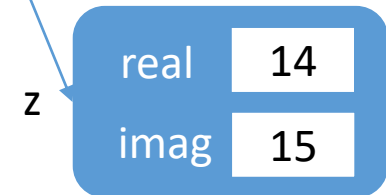
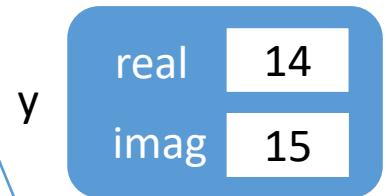
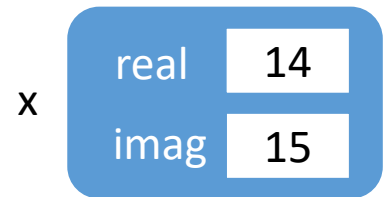
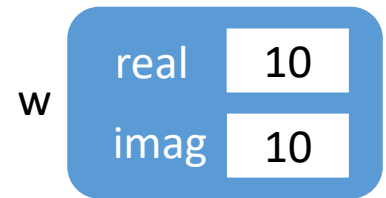
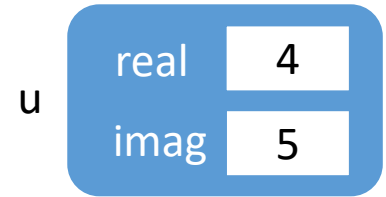
real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

Cursor here

u

real	4
imag	5

w

real	10
imag	10

x

real	14
imag	15

y

real	14
imag	15

z

real	14
imag	15


```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

As the objects u, x, y and z are not used in the remaining code, their corresponding boxes are not drawn from now.

w	real	10
	imag	10

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
int main()
{
```

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
```

```
    return 0;
```

```
}
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

Cursor here

w

real	10
imag	10

a

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

```
        }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return this;
```

```
        }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

```
        {
```

```
            real += x.real; imag += x.imag;
```

```
            return (*this);
```

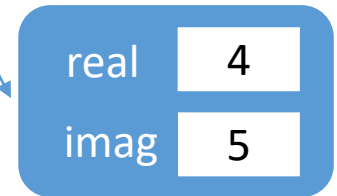
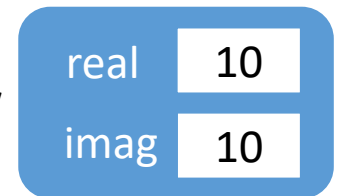
```
        }
```

```
};
```



w

a



Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

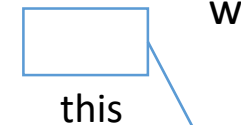
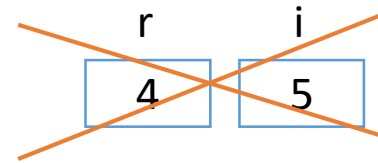
```
        {
            real += x.real; imag += x.imag;
            return this;
        }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

```
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
```

```
};
```



w

a

real	10
imag	10

real	4
imag	5

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

(4 , 5)
(4 , 5)
(4 , 5)

```
int main()
{
```

(14 , 15)
(14 , 15)
(14 , 15)

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

a

real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



this

x
w

a

real	10
imag	10

real	4
imag	5

Output

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

this

x
w

a

real	10
imag	10

real	14
imag	5

Output

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```



this

x
w

a

real	10
imag	10

real	14
imag	15

Output

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

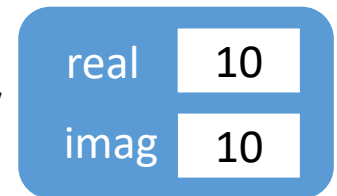
```

*this is a,
Returning it by value, we need to construct a
temporary object "temp1" which is a copy of a

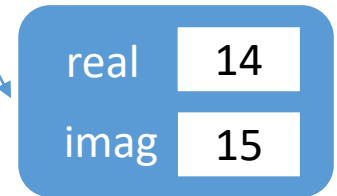


this

x
w



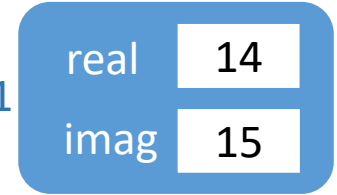
a



Output

(4 , 5)
(4 , 5)
(4 , 5)

temp1



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
int main()
{
```

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
```

```
    return 0;
```

```
}
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

w

real	10
imag	10

This is temp1

a

real	14
imag	15

temp1

real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

x
w

real	10
imag	10

a

real	14
imag	15

Output

(4 , 5)
(4 , 5)
(4 , 5)

this

temp1

real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

x
w

real	10
imag	10

a

real	14
imag	15

Output

(4 , 5)
(4 , 5)
(4 , 5)

this

temp1

real	24
imag	15

(14 , 15)
(14 , 15)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

x
w

real	10
imag	10

a

real	14
imag	15

Output

(4 , 5)
(4 , 5)
(4 , 5)

this

temp1

real	24
imag	25

(14 , 15)
(14 , 15)

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

*this is temp1,
Returning it by value, we need to construct a
temporary object "temp2" which is a copy of temp1

x
w

real	10
imag	10

a

real	14
imag	15

Output

(4 , 5)
(4 , 5)
(4 , 5)

this

temp1

real	24
imag	25

temp2

real	24
imag	25

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
int main()
{
```

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
```

```
    return 0;
```

```
}
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

w

real	10
imag	10

This is temp2

a

real	14
imag	15

temp1

real	24
imag	25

temp2

real	24
imag	25

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )

```

x
w

real	10
imag	10

a

real	14
imag	15

temp1

real	24
imag	25



this

temp2

real	24
imag	25


```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
int main()
{
```

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); t << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))>print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
```

```
    return 0;
```

```
}
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
( 24 , 25 )
```

w

real	10
imag	10

a

real	14
imag	15

temp1

real	24
imag	25

temp2

real	24
imag	25

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
int main()
{
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

w

real	10
imag	10

```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

a

real	14
imag	15

```
( 24 , 25 )
```

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

this

w

a

real	10
imag	10

real	14
imag	15

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

Cursor here

w

real	10
imag	10

a

real	14
imag	15

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
int main()
```

```
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
```

w

real	10
imag	10

```
Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
cout << endl << endl; // What is the output now?
Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
return 0;
```

}

a

real	14
imag	15

b

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return this;
```

```
    }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
};
```



Output

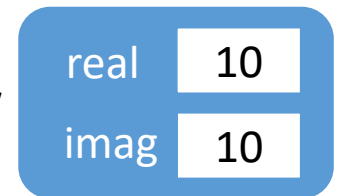
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

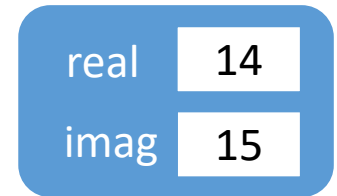
```
( 24 , 25 )
( 14 , 15 )
```

this

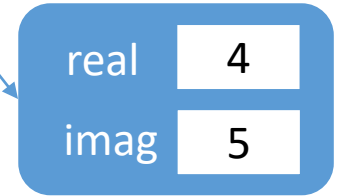
w



a



b



```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

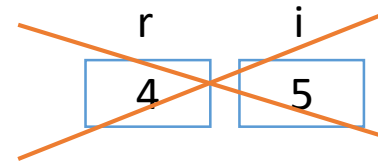
```
        {
            real += x.real; imag += x.imag;
            return this;
        }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

```
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
```

```
};
```



Output

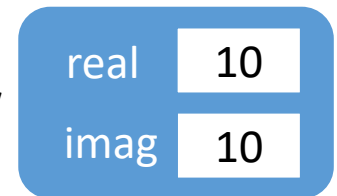
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

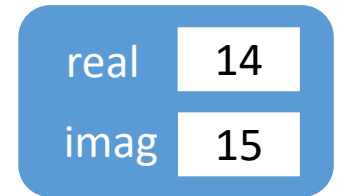
```
( 24 , 25 )
( 14 , 15 )
```

this

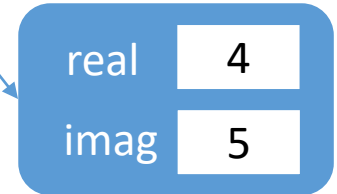
w



a



b



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
int main()
{
```

```
( 24 , 25 )
( 14 , 15 )
```

w

real	10
imag	10

```
    // Check the parameter passing methods
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print();    a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print();    c.print();
```

```
    return 0;
```

```
}
```

a

real	14
imag	15

b

real	4
imag	5


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	14
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	14
imag	15

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
int main()
```

```
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); c
```

w

real	10
imag	10

```

    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

This is &b, i.e. address of b

a

real	14
imag	15

b

real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

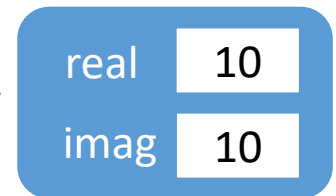
(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

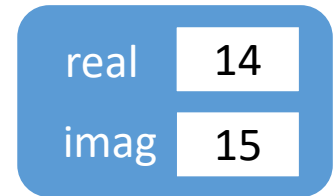
(24 , 25)
(14 , 15)

this

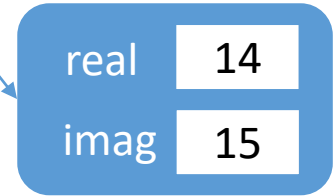
x
w



a



b



```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

this


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
int main()
```

```
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); c
```

w

real	10
imag	10

```
endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

This is &b, i.e. address of b

a

real	14
imag	15

b

real	24
imag	25

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

(24 , 25)

this

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"
```

Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
int main()
{
```

w

real	10
imag	10

```
    // Check the parameter passing methods
```

```
( 24 , 25 )
( 14 , 15 )
```

```
    Complex u(4, 5); f(u); g(&u); h(u);
```

```
    // Check the parameter returning methods
```

```
( 24 , 25 )
```

```
    Complex w(10, 10); cout << endl << endl;
```

```
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
```

```
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
```

```
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
```

```
    cout << endl << endl; // What is the output now?
```

```
    Complex a(4, 5); a.add1(w).add1(w).print();    a.print(); cout << endl;
```

```
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
```

```
    Complex c(4, 5); c.add3(w).add3(w).print();    c.print();
```

```
    return 0;
```

```
}
```

a

real	14
imag	15

b

real	24
imag	25

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

(24 , 25)
(24 , 25)

this

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)
 (24 , 25)
 (24 , 25)

Cursor here

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	
imag	

```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
        Complex* add2(const Complex& x)
```

```
        // Return by value using pointer
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return this;
```

```
    }
```

```
        Complex& add3(const Complex& x)
```

```
        // Return by reference
```

```
    {
```

```
        real += x.real; imag += x.imag;
```

```
        return (*this);
```

```
    }
```

```
};
```



Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

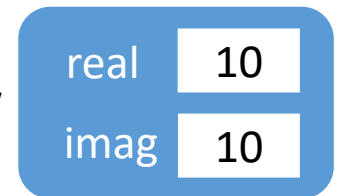
```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
( 24 , 25 )
( 14 , 15 )
```

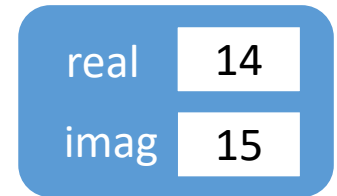
```
( 24 , 25 )
( 24 , 25 )
```

this

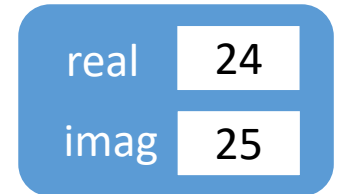
w



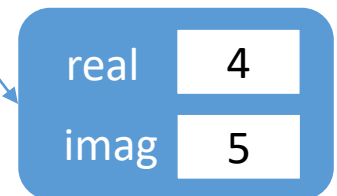
a



b



c




```
class Complex /* File: complex.h */
{
```

```
    private:
```

```
        float real; float imag;
```

```
    public:
```

```
        Complex(float r, float i) { real = r; imag = i; }
```

```
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
        Complex add1(const Complex& x) // Return by value
```

```
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
```

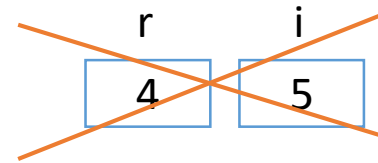
```
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
```

```
            real += x.real; imag += x.imag;
            return this;
        }
```

```
        Complex& add3(const Complex& x)
        // Return by reference
        {
```

```
            real += x.real; imag += x.imag;
            return (*this);
        }
```

```
};
```



Output

```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

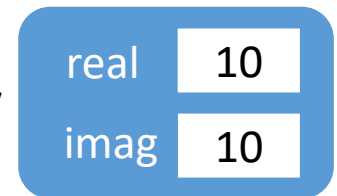
```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```

```
( 24 , 25 )
( 14 , 15 )
```

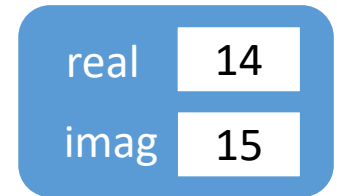
```
( 24 , 25 )
( 24 , 25 )
```

this

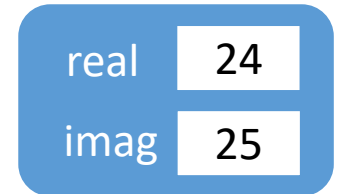
w



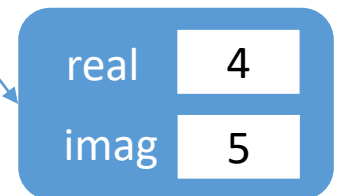
a



b



c



Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	4
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	5

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

*this is c,
Returning it by reference, i.e. returning c

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x. This is c ).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15


```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	15

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

x
w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

*this is c,
Returning it by reference, i.e. returning c

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x. This is c ).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

```

class Complex /* File: complex.h */
{
    private:
        float real; float imag;
    public:
        Complex(float r, float i) { real = r; imag = i; }
        void print() { cout << "( " << real << " , " << imag << " )" << endl; }

        Complex add1(const Complex& x) // Return by value
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
        Complex* add2(const Complex& x)
        // Return by value using pointer
        {
            real += x.real; imag += x.imag;
            return this;
        }
        Complex& add3(const Complex& x)
        // Return by reference
        {
            real += x.real; imag += x.imag;
            return (*this);
        }
};

```

Output

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

(24 , 25)
(24 , 25)

(24 , 25)

this

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

```

class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x)
    // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex& x)
    // Return by reference
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
};

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

```

```

( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

```

( 24 , 25 )
( 14 , 15 )

```

```

( 24 , 25 )
( 24 , 25 )

```

```

( 24 , 25 )
( 24 , 25 )

```

this

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

a

real	14
imag	15

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)

w

real	10
imag	10

Output

```

#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}

```

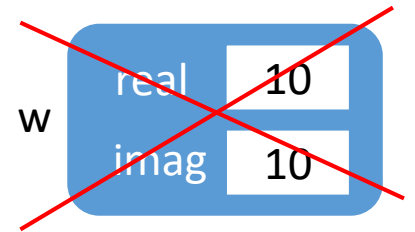
(4 , 5)
 (4 , 5)
 (4 , 5)

(14 , 15)
 (14 , 15)
 (14 , 15)

(24 , 25)
 (14 , 15)

(24 , 25)
 (24 , 25)

(24 , 25)
 (24 , 25)





Output

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

(24 , 25)
(24 , 25)

(24 , 25)
(24 , 25)

Output

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }

int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* temp = this = &y
    Complex z(4, 5); (z.add3(w)).print(); // Complex& temp = *this = z
    cout << endl << endl; // What is the output now?
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

(4 , 5)
(4 , 5)
(4 , 5)

(14 , 15)
(14 , 15)
(14 , 15)

(24 , 25)
(14 , 15)

(24 , 25)
(24 , 25)

(24 , 25)
(24 , 25)

Done!!! :)