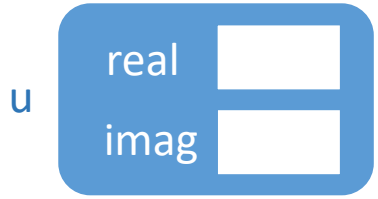
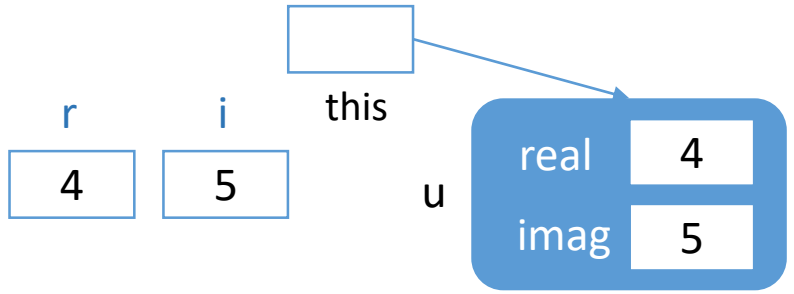
Step-by-step walkthrough for example on page 15 – 16 of the lecture notes:
Revision Example, Pointer, Reference & const-ness

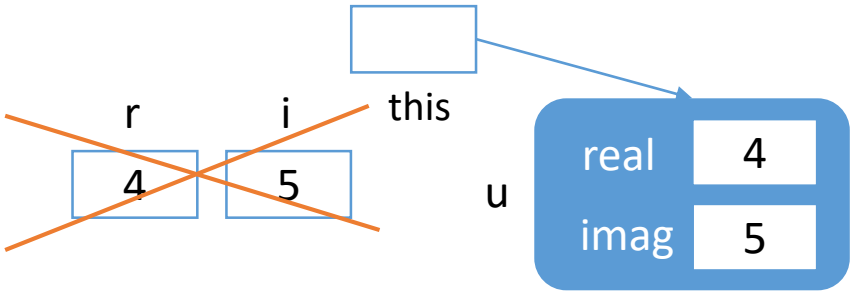
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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"

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;
}
```



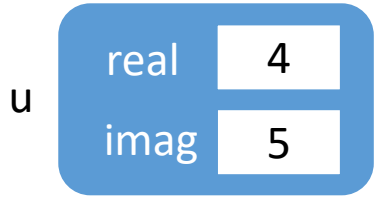



```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```

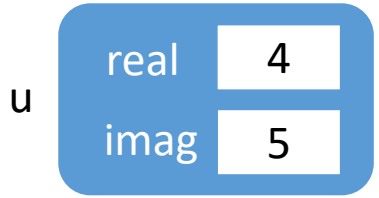


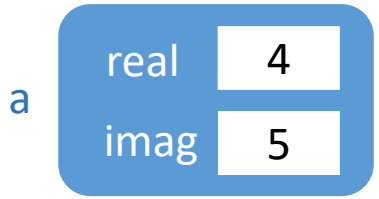

```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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"

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;
}
```

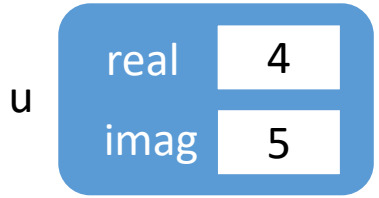


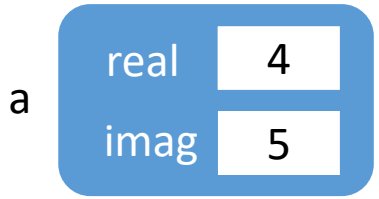
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```



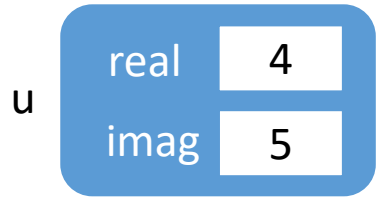


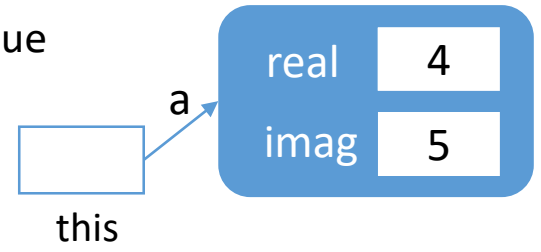
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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);
    }
};
```

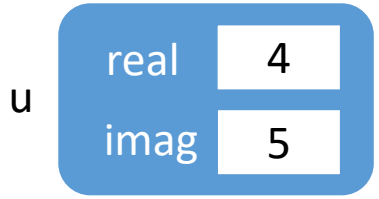


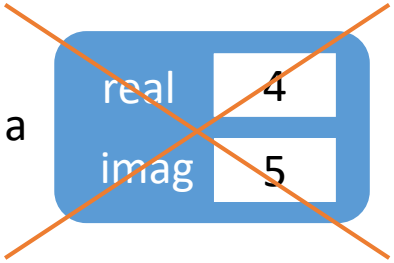


Output

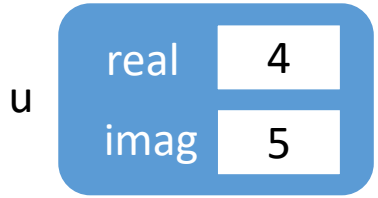
```
( 4 , 5 )
```

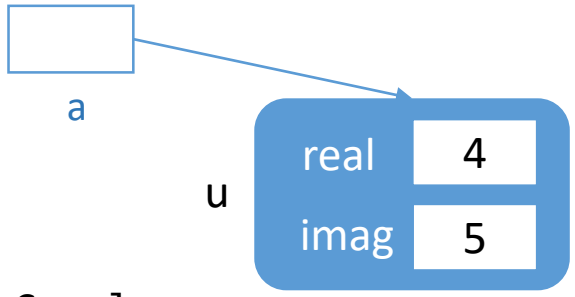
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```



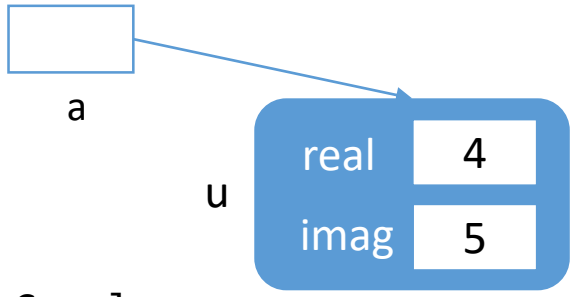


```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```

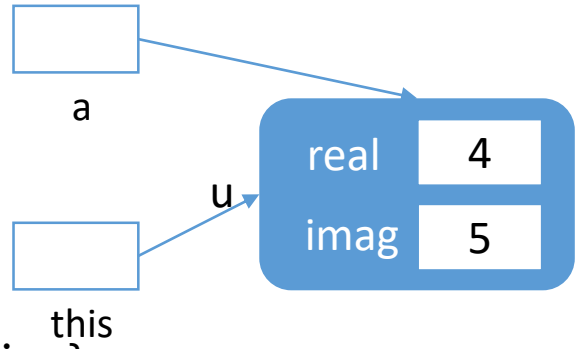





```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```



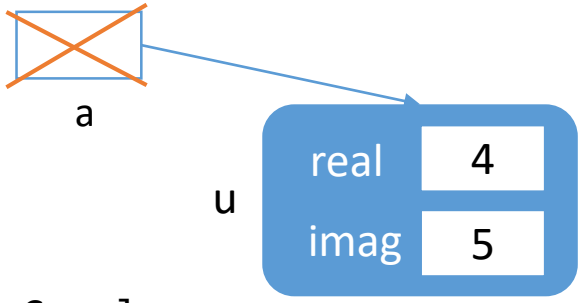

```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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);
    }
};
```

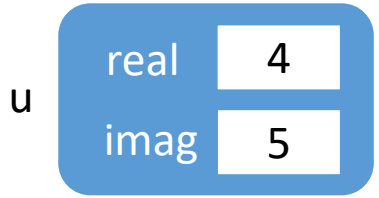
Output

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

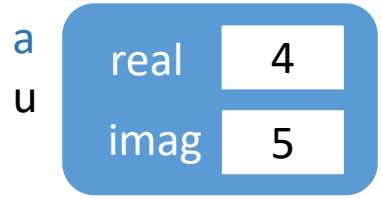



```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```

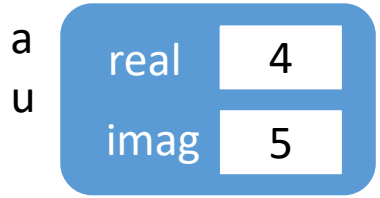
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```



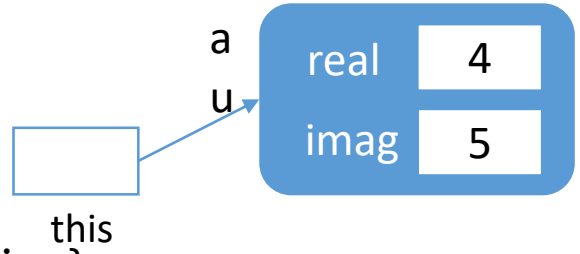
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```



```
#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;
}
```



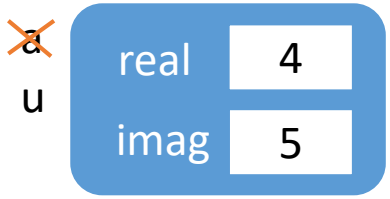
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



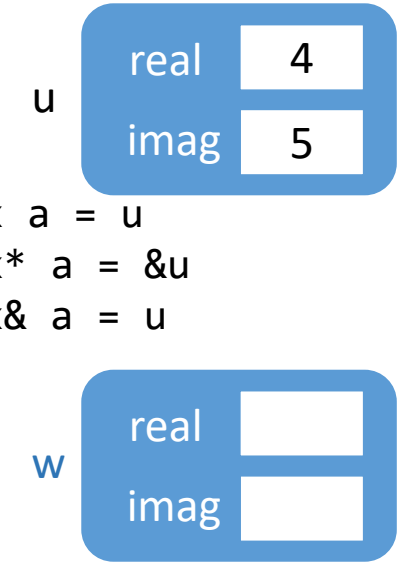
Output

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

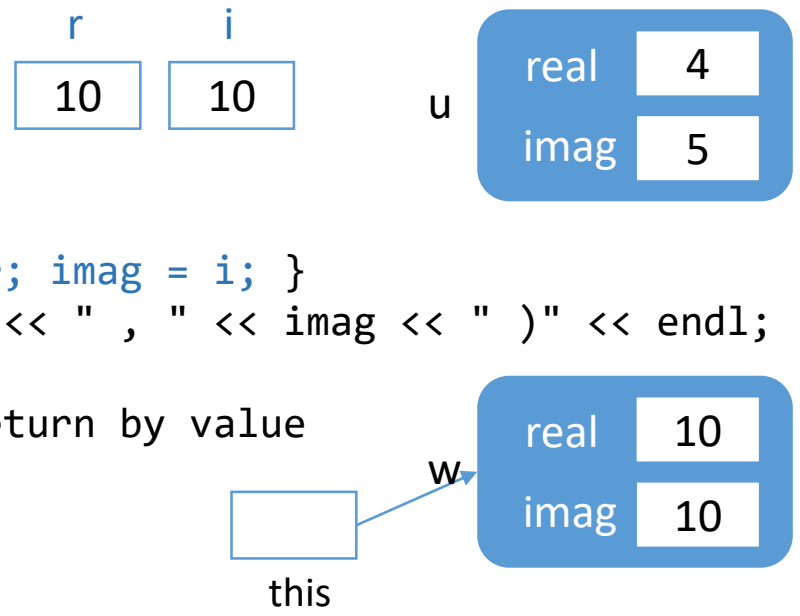
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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"

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;
}
```



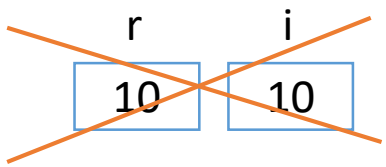
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```

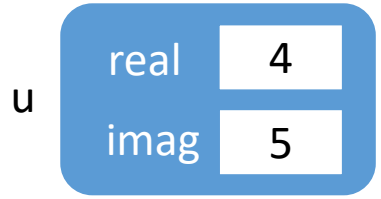


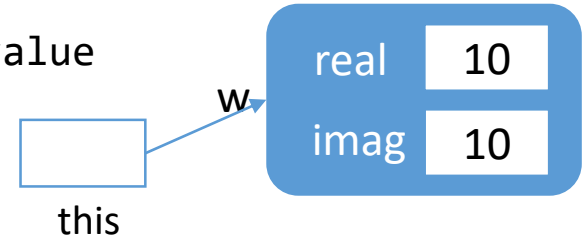
Output

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

```
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"

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

Output

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

Cursor here

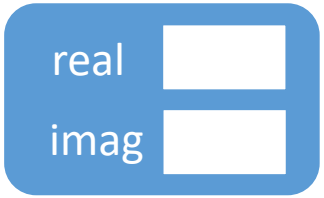
_

```
#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;
}
```

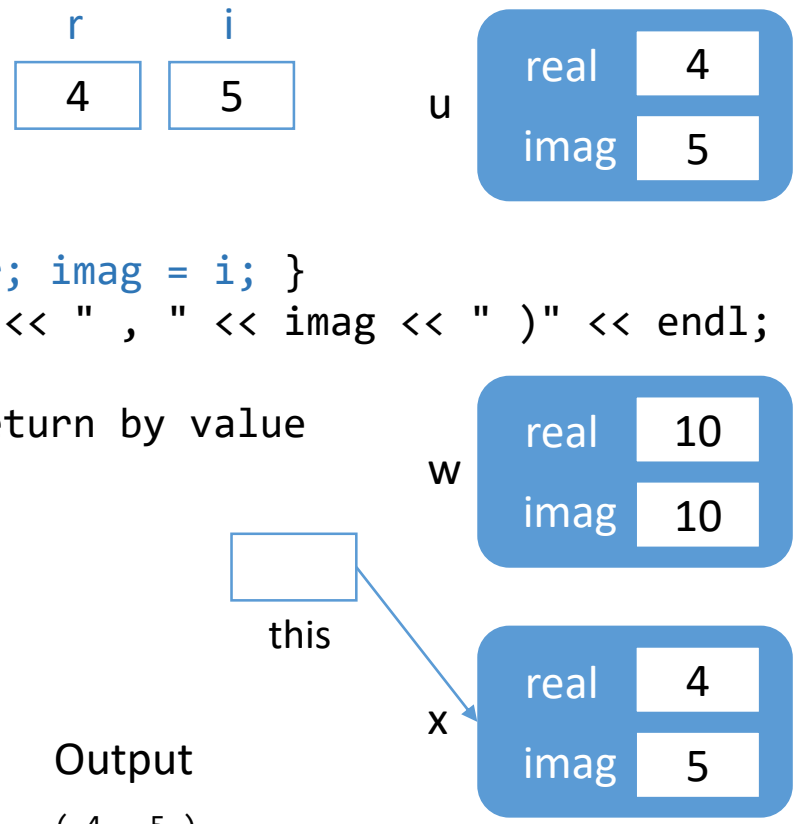
u

w

x



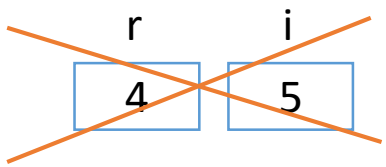
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```

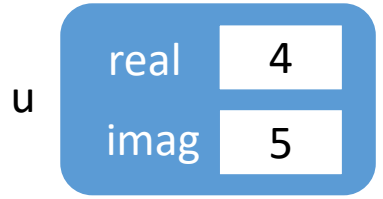


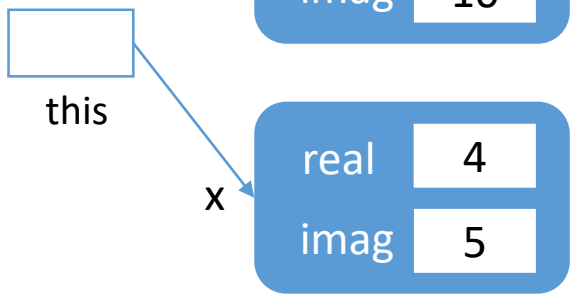
Output

```
( 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 )
```

```
#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

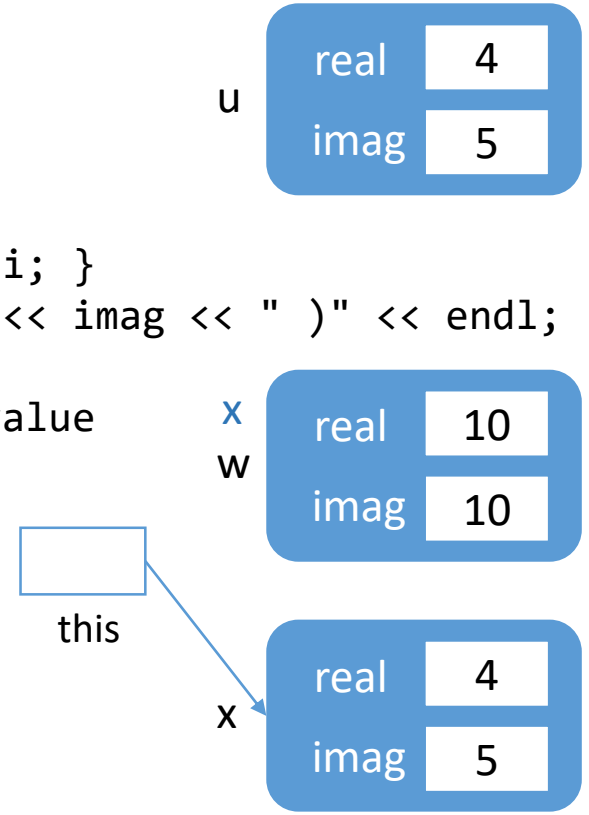
w

real	10
imag	10

x

real	4
imag	5

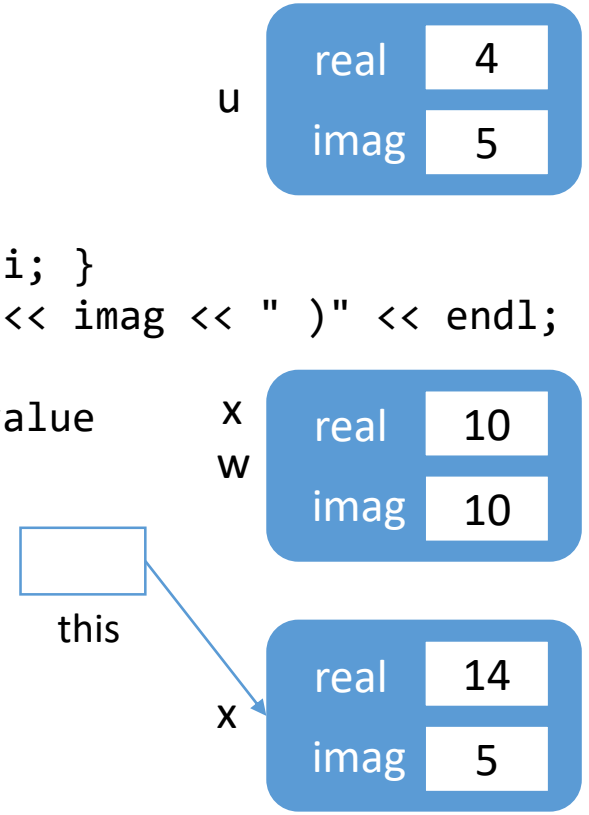
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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 )
```

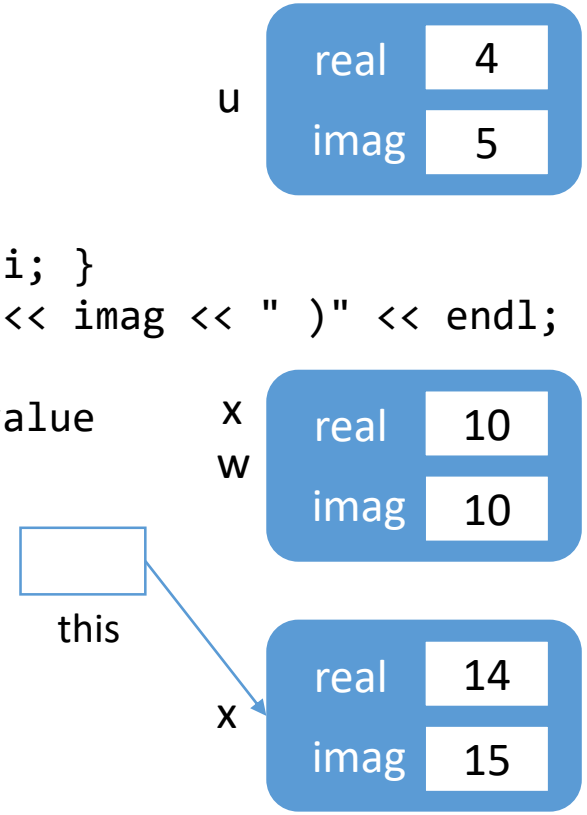
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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

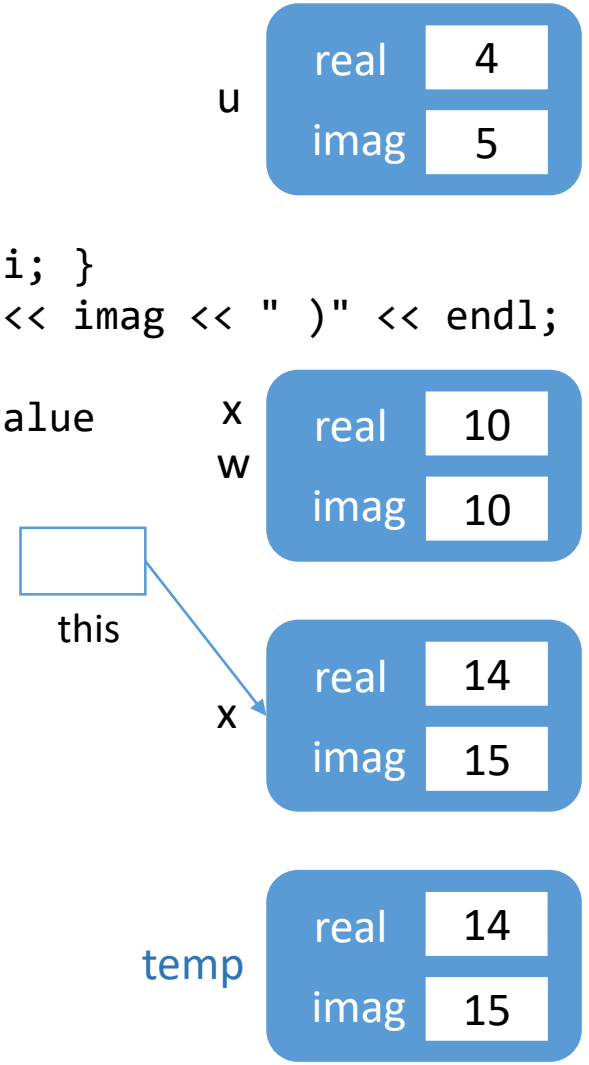
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

```
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"

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;
}
```

This is temp

u

real	4
imag	5

w

real	10
imag	10

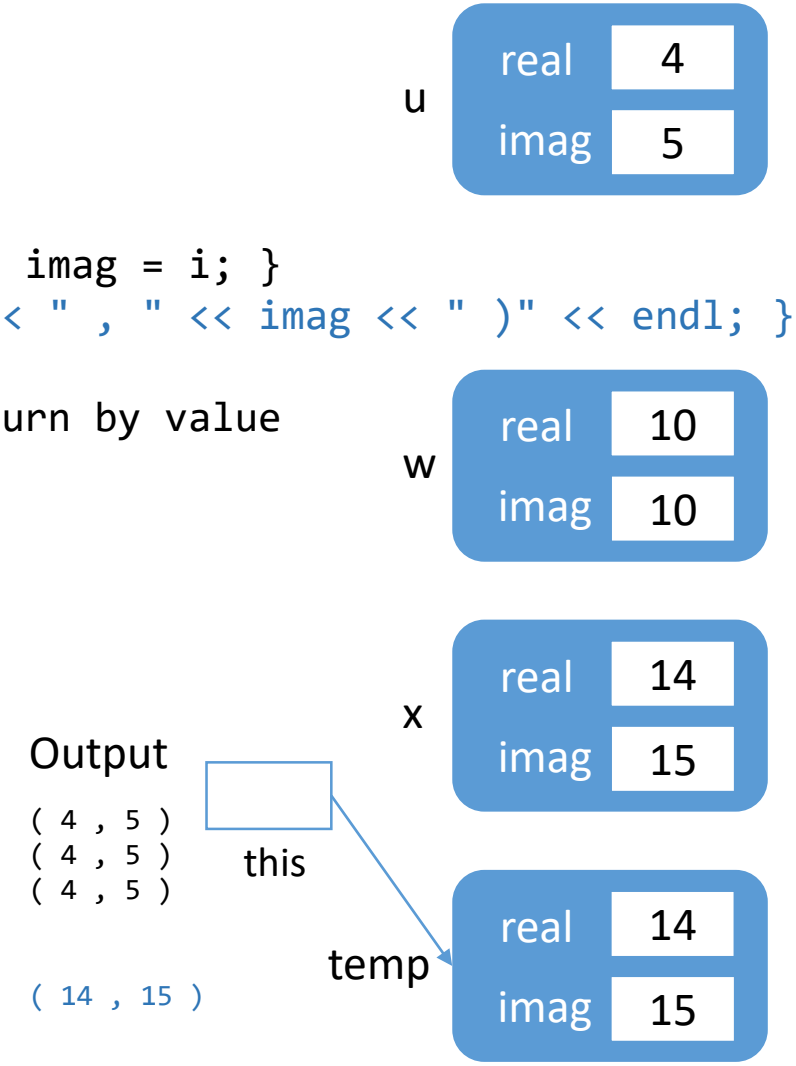
x

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);
    }
};
```



```
#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;
}
```

This is temp

u

real	4
imag	5

w

real	10
imag	10

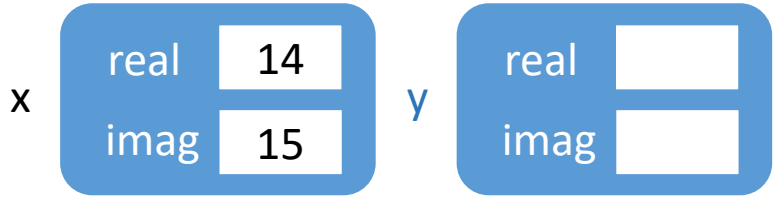
x

real	14
imag	15

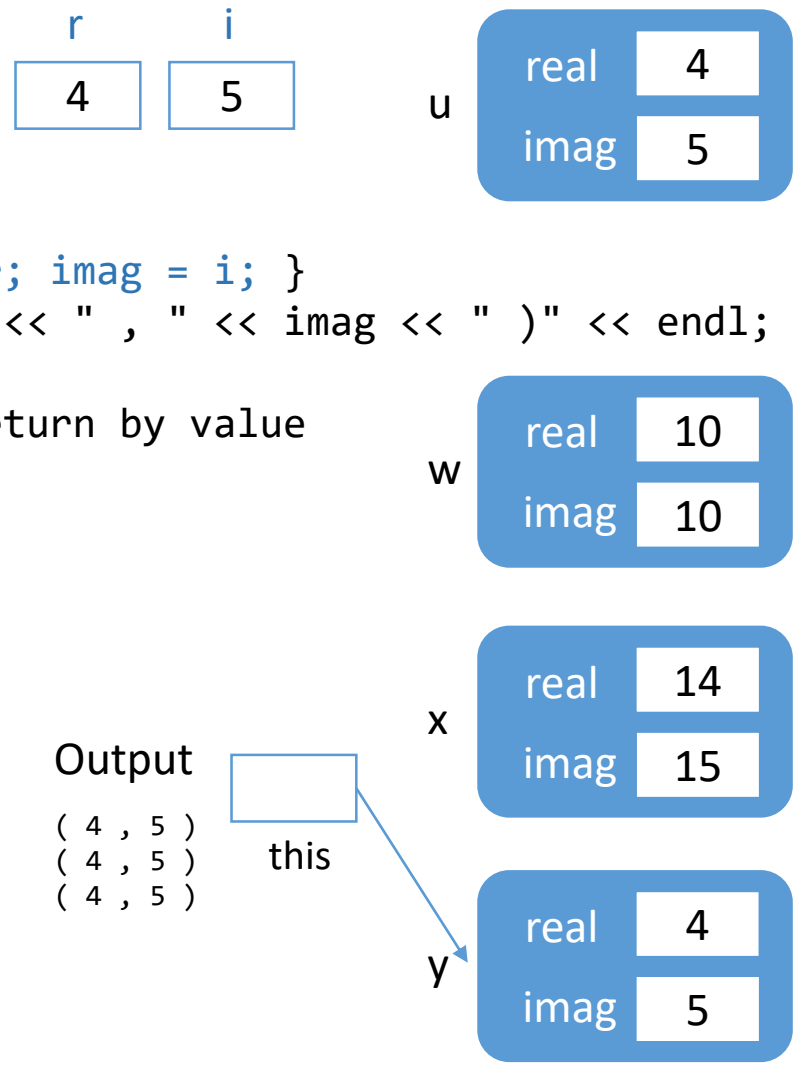
temp

~~real~~	~~14~~
~~imag~~	~~15~~

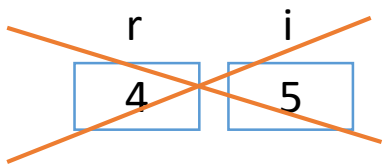
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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;
}
```

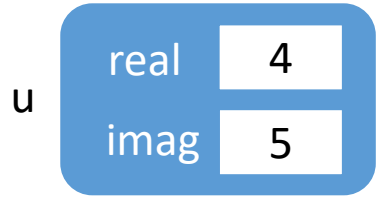


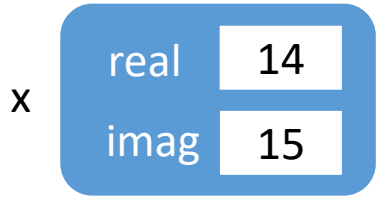
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



```
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);
    }
};
```

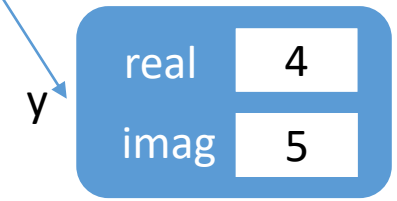






Output

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



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

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

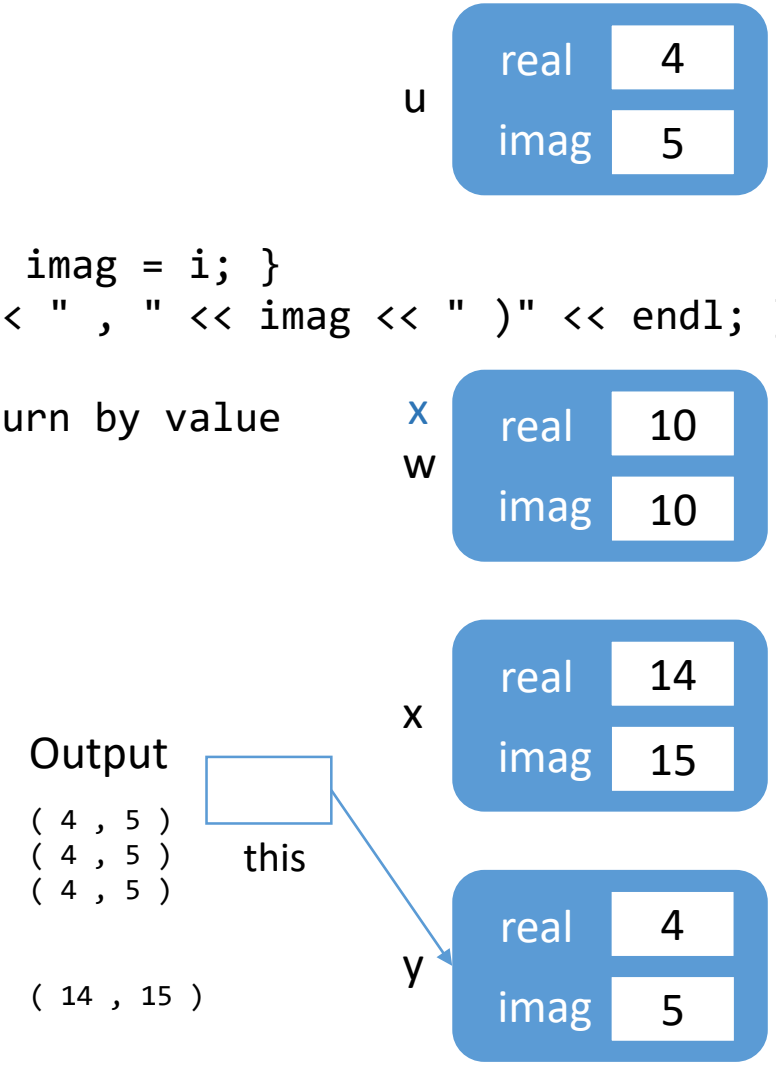
x

real	14
imag	15

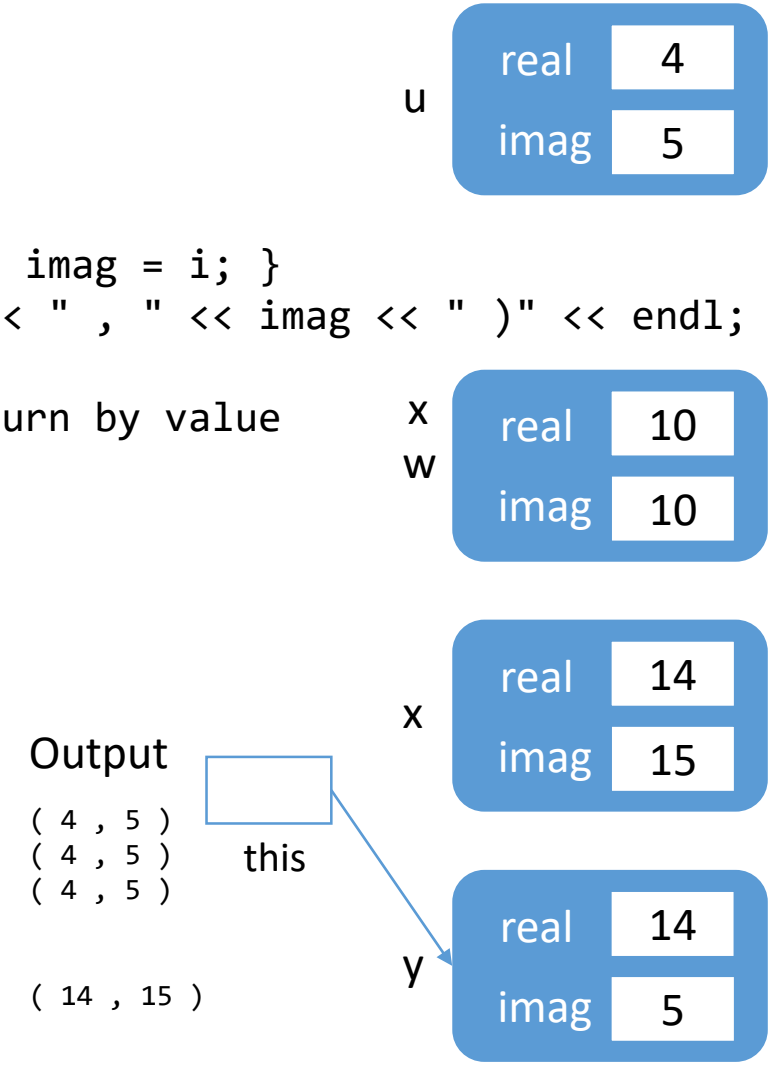
y

real	4
imag	5

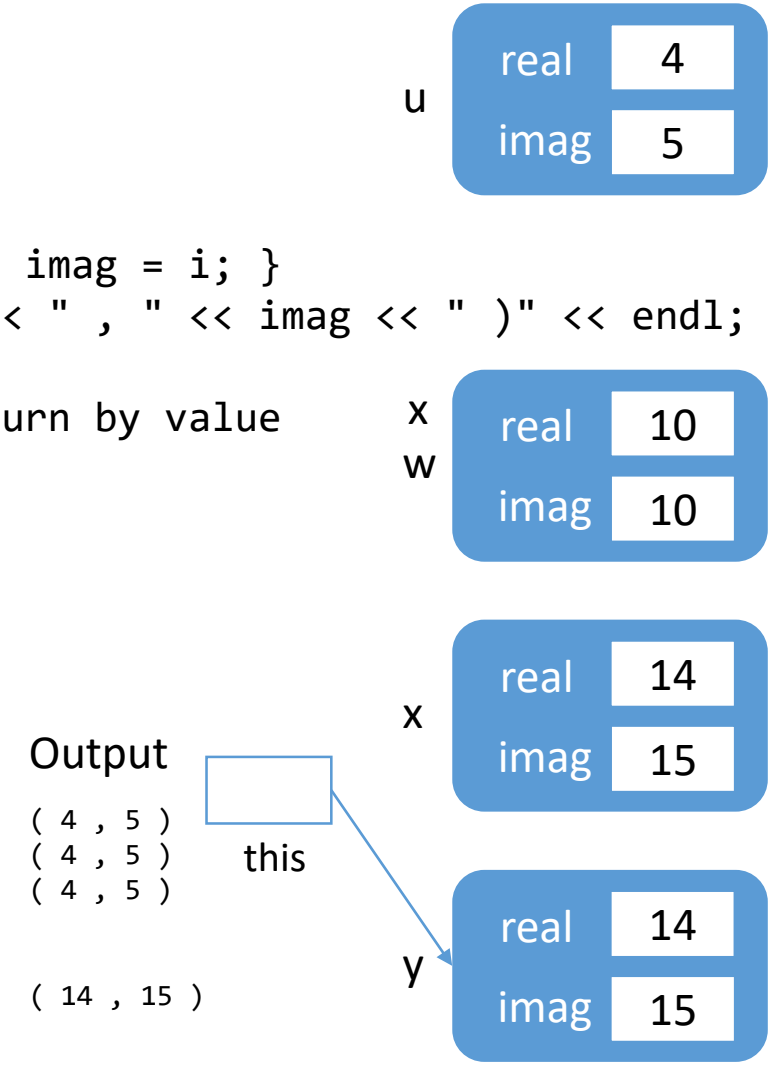
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



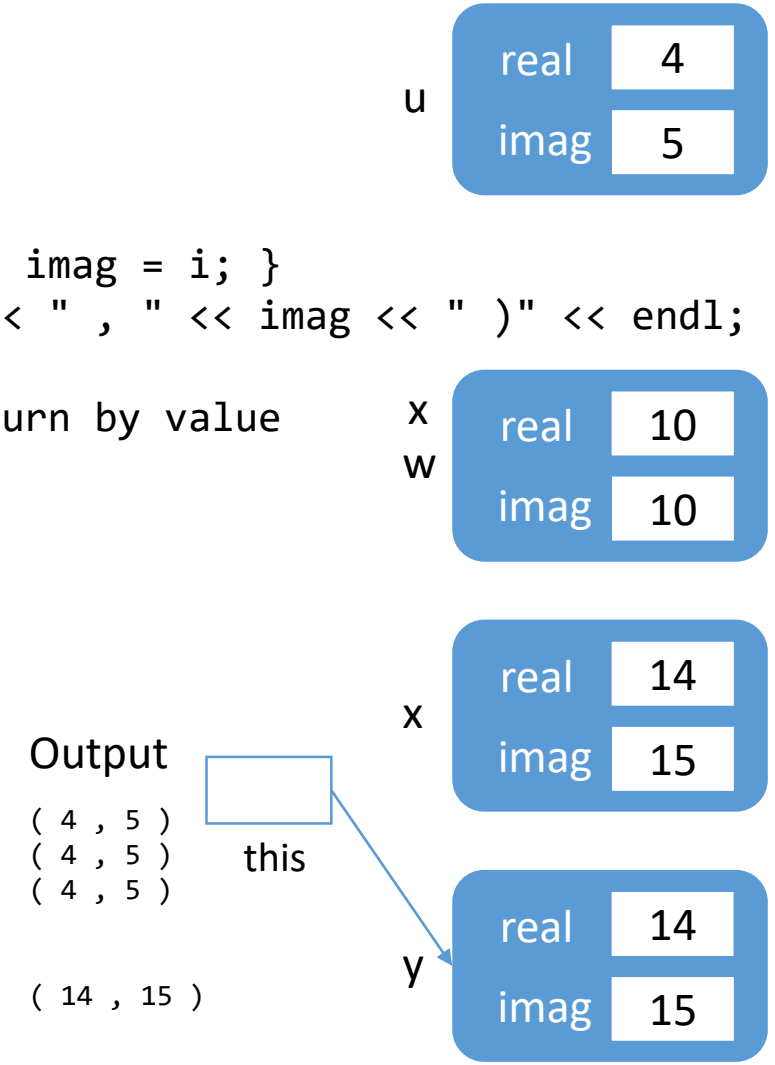
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



```
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"

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;
}
```

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

u

real	4
imag	5

w

real	10
imag	10

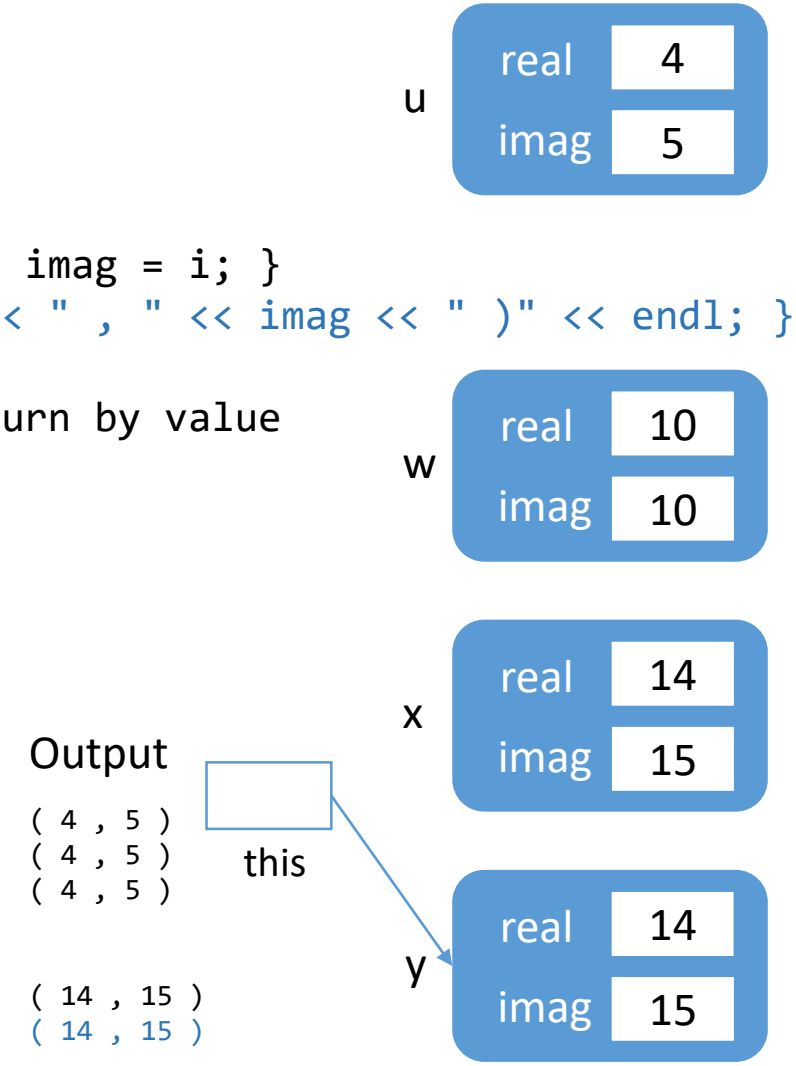
x

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);
    }
};
```



```
#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	14
imag	15

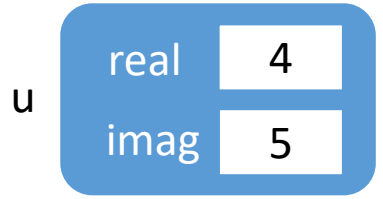
y

real	14
imag	15

z

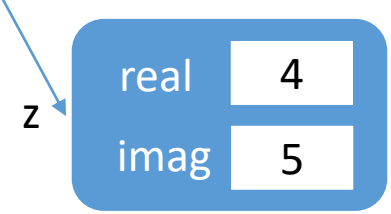
real	
imag	

```
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);
    }
};
```

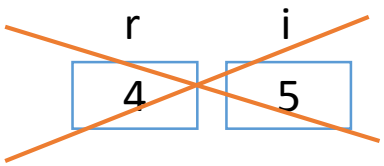


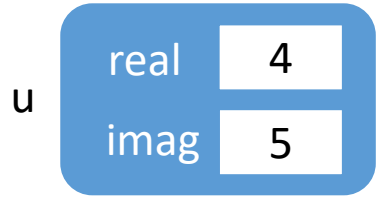
Output

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

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



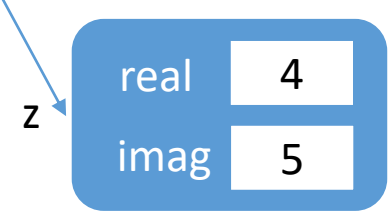
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```





Output

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

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



```
#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	14
imag	15

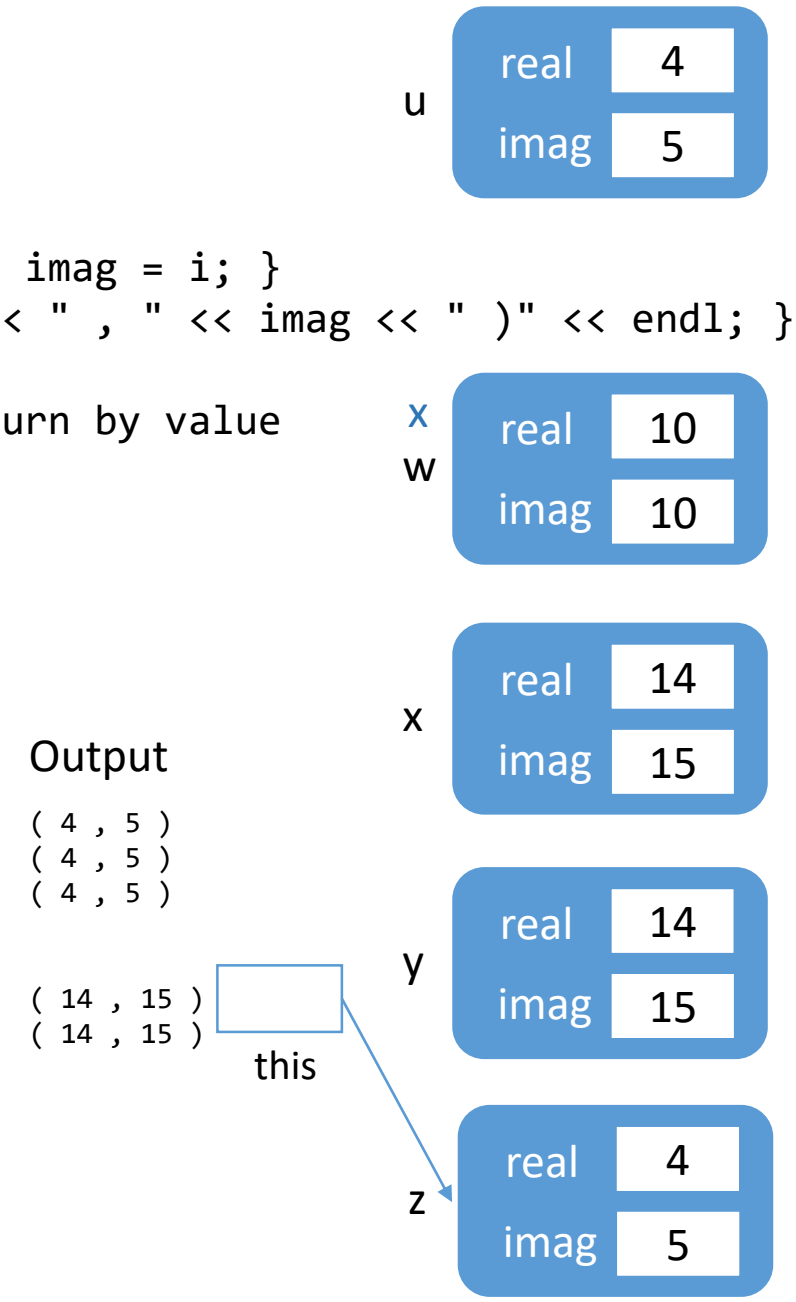
y

real	14
imag	15

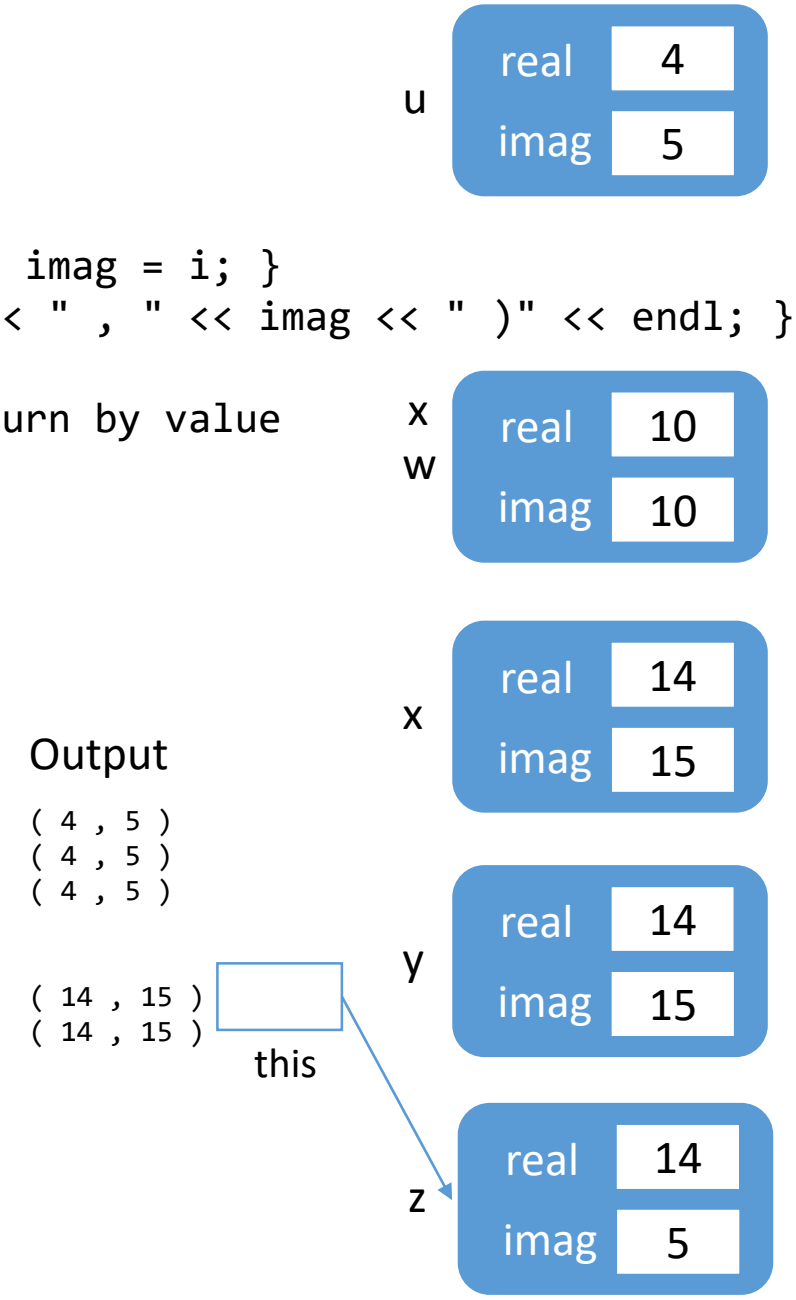
z

real	4
imag	5

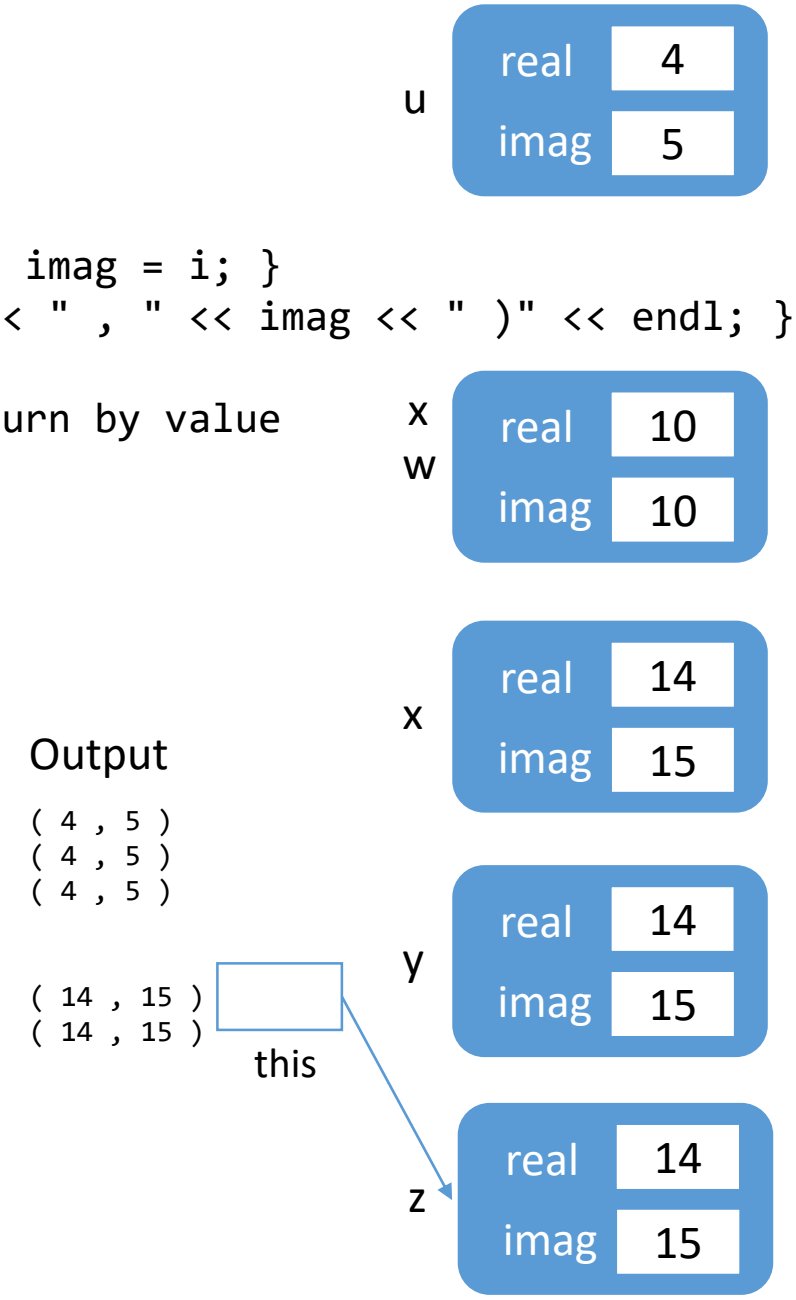
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



```
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);
    }
};
```

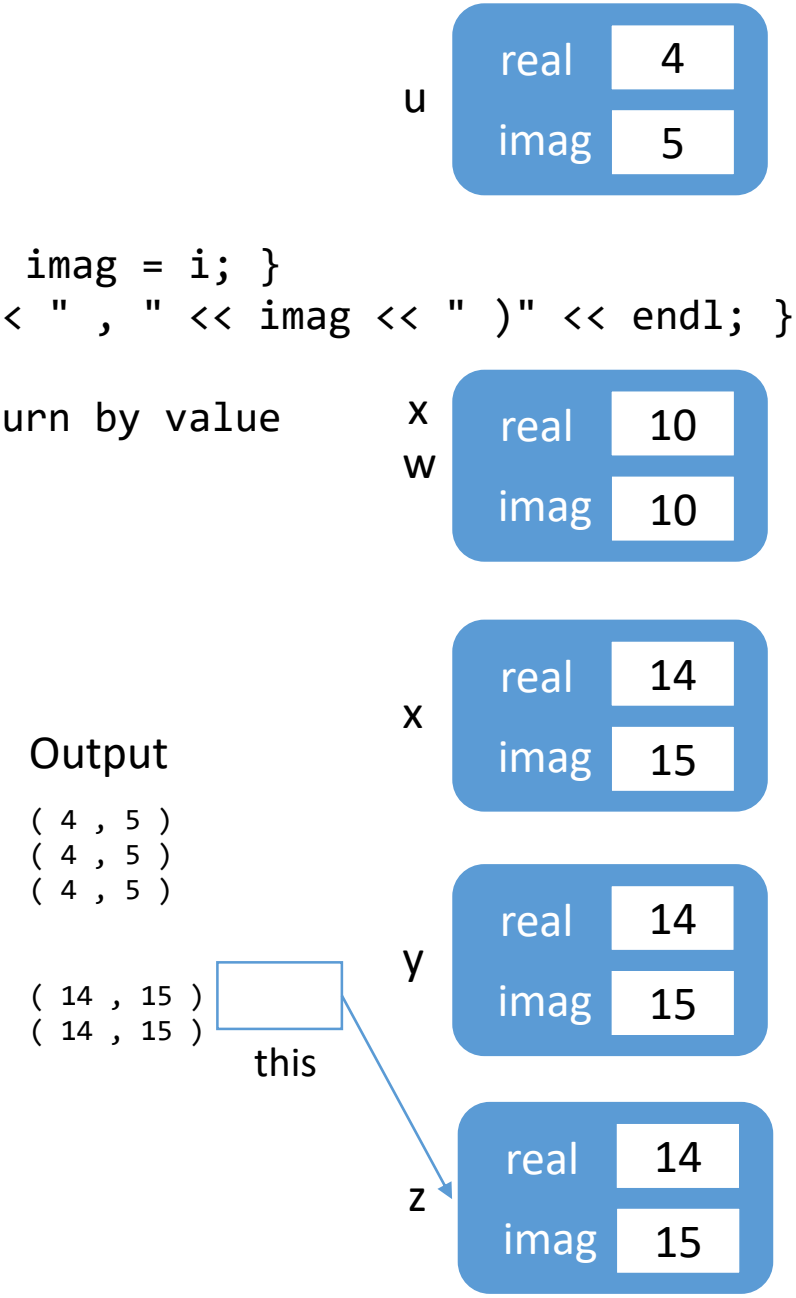


```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



```
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 z,
Returning it by reference, i.e. returning z



```
#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 paramete This is z ing 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	14
imag	15

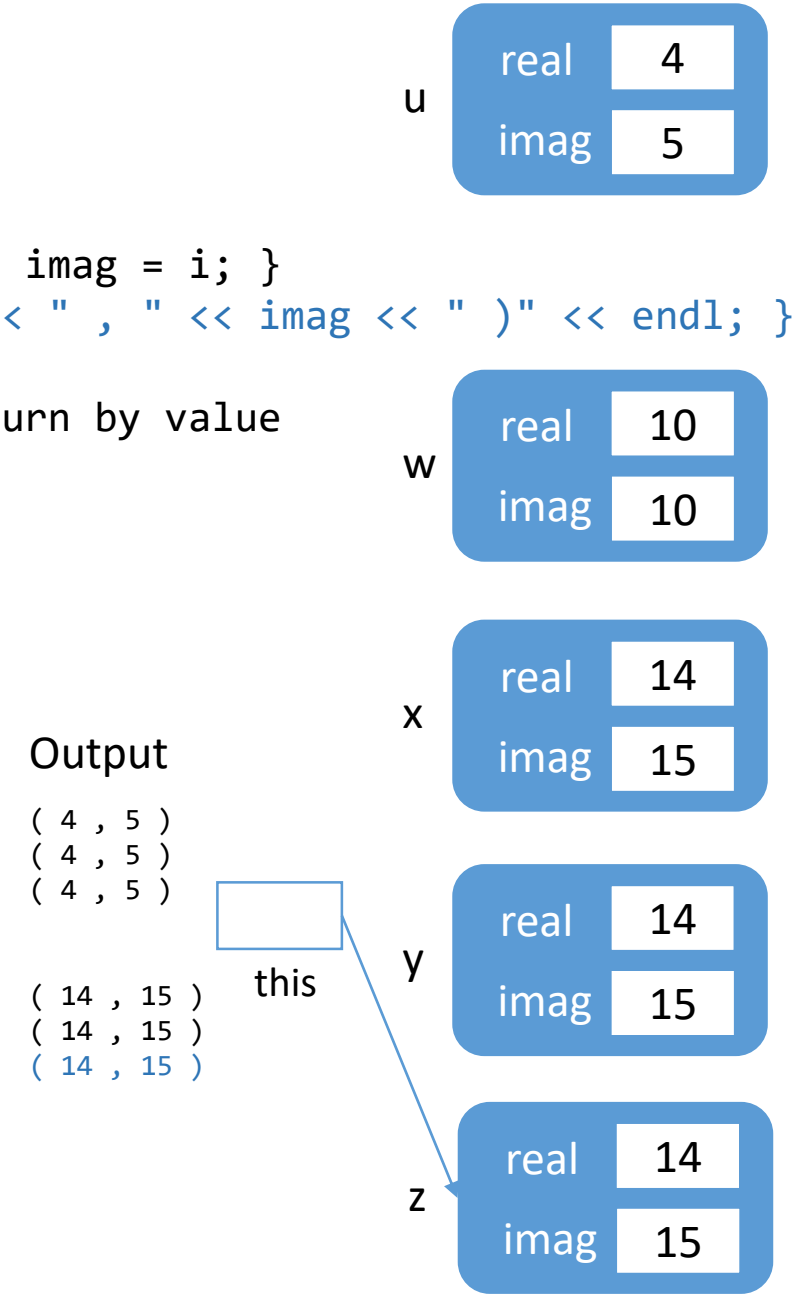
y

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);
    }
};
```



```
#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

u x y z

```
#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.

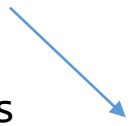
u x y z

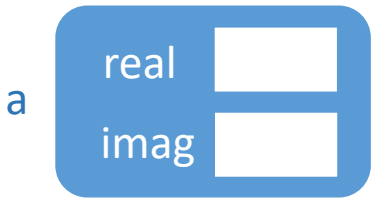
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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

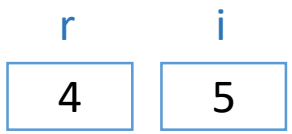
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

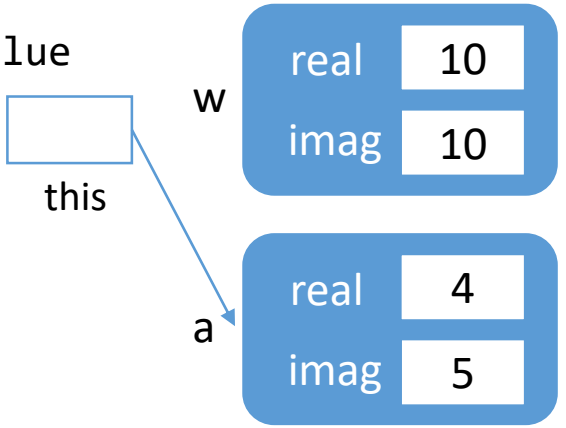
Cursor here





```
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);
    }
};
```

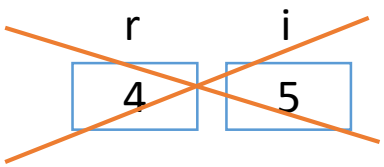


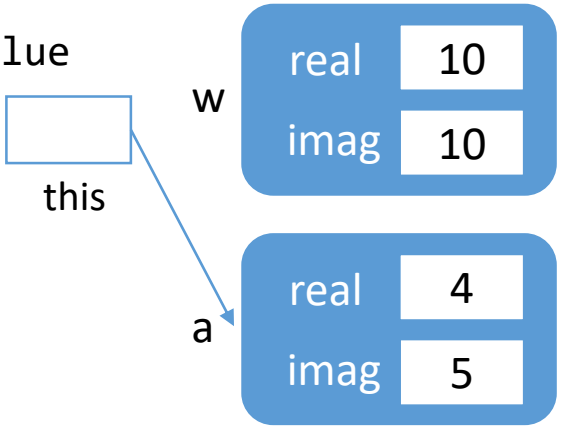


Output

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

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

```
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

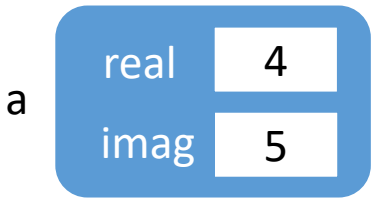
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

u x y z

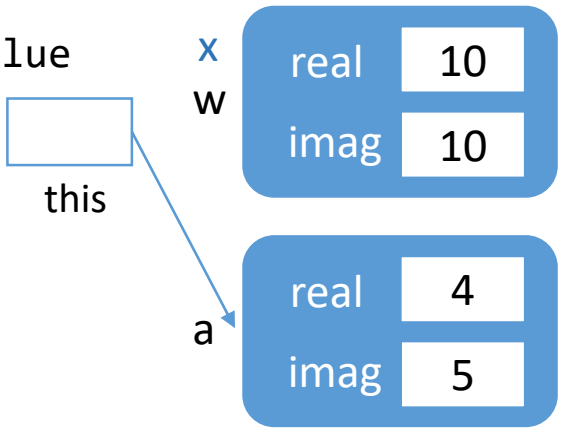
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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 )
```



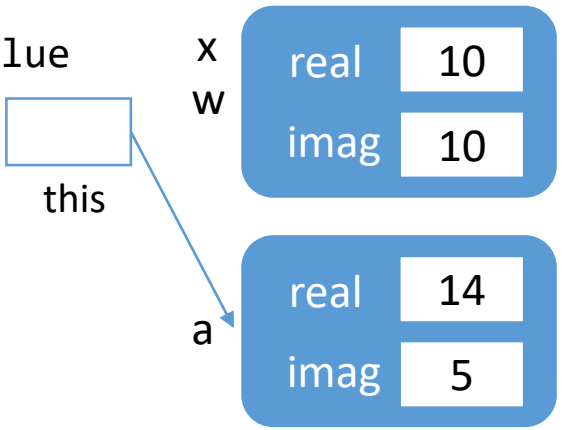
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```

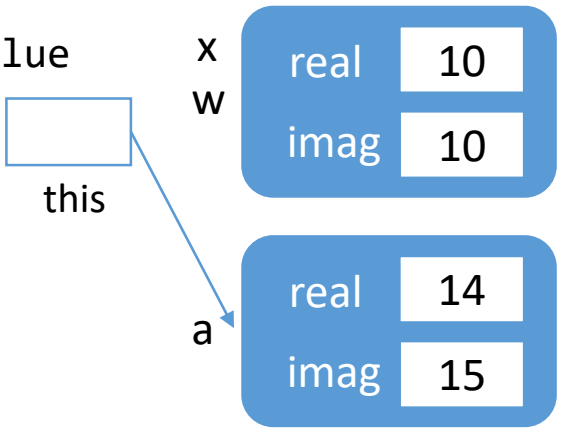


Output

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

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

u x y z

```
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);
    }
};
```



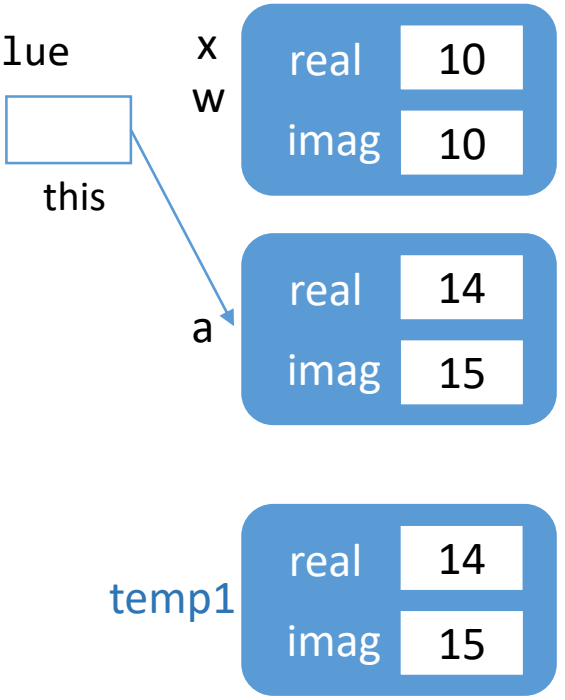
Output

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

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

u x y z

```
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



Output

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

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

u x y z

```
#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); co    dl << 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 )
```

This is temp1

w

real	10
imag	10

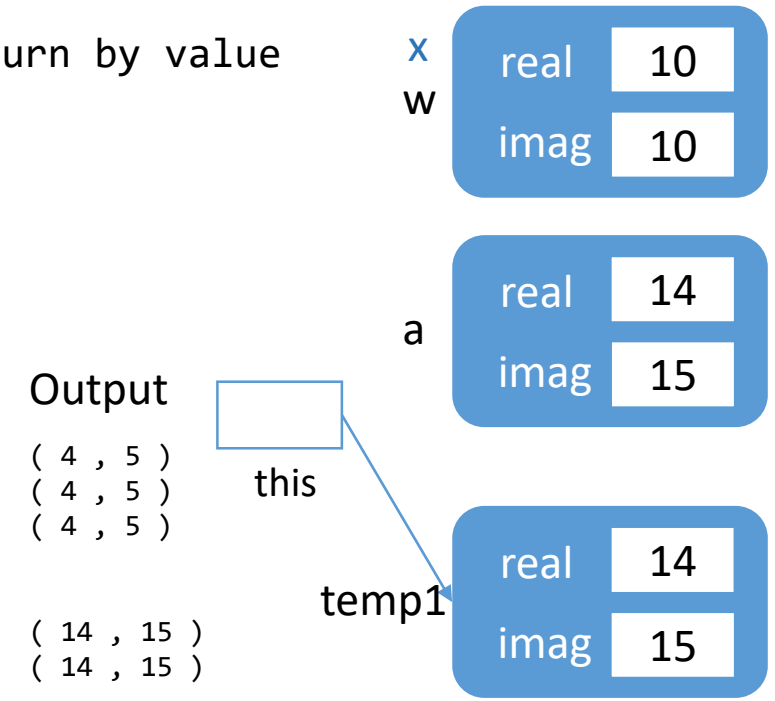
a

real	14
imag	15

temp1

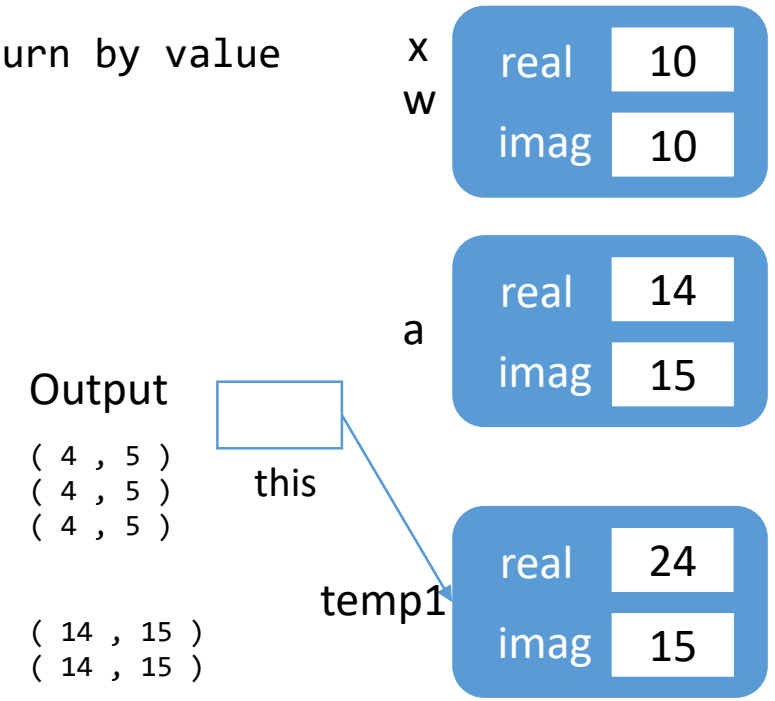
real	14
imag	15

```
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);
    }
};
```



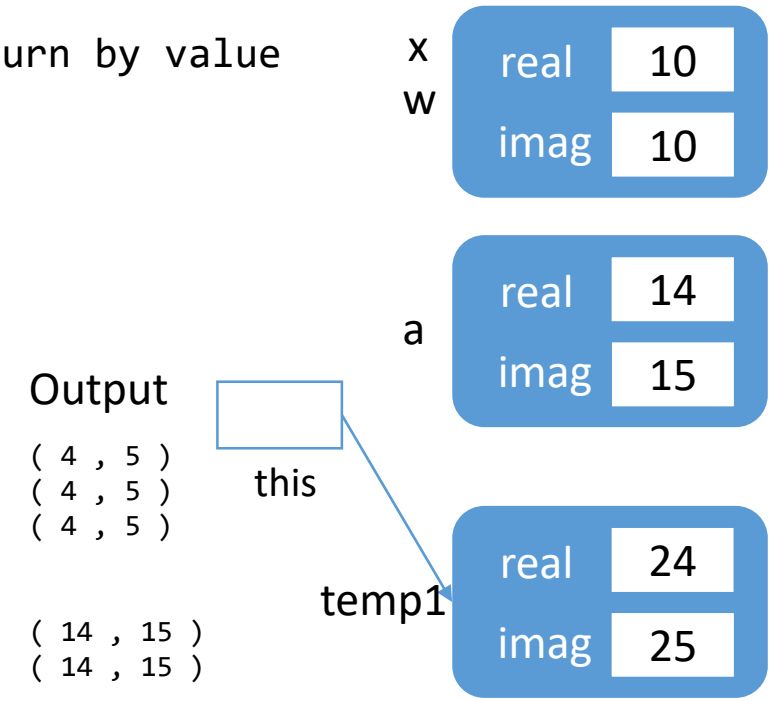
u x y z

```
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 x y z

```
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);
    }
};
```



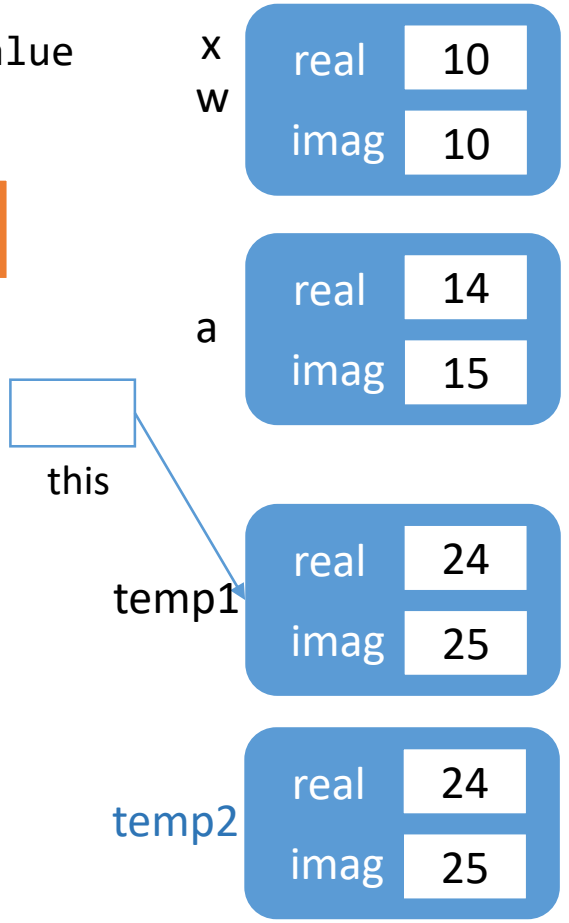
u x y z

```
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

Output

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

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



u x y z

```
#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;
  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 )
```

This is temp2

w

real	10
imag	10

a

real	14
imag	15

temp1

real	24
imag	25

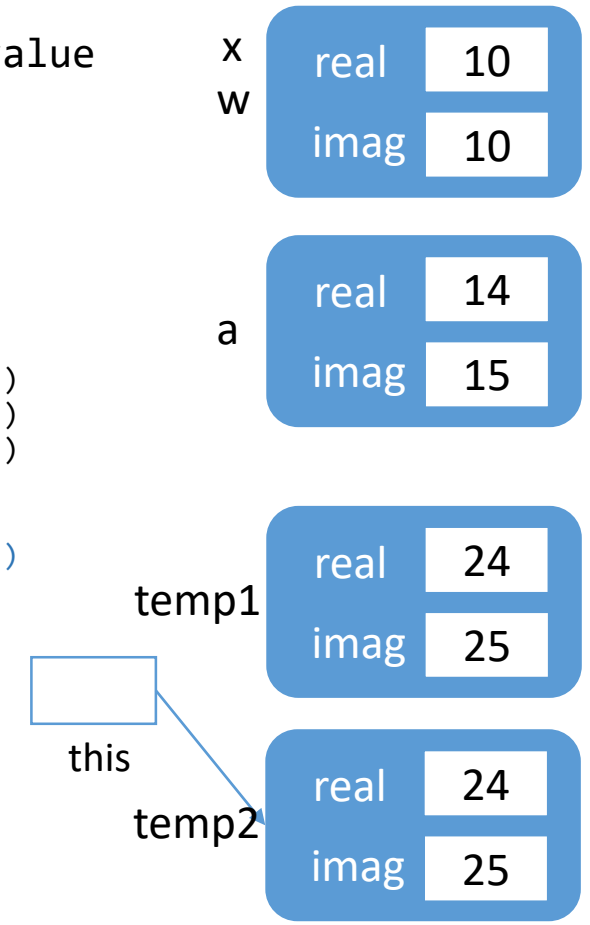
temp2

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 )
```



```
#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;
  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 )
```

This is temp1

This is temp2

w

real	10
imag	10

a

real	14
imag	15

temp1

real	24
imag	25

temp2

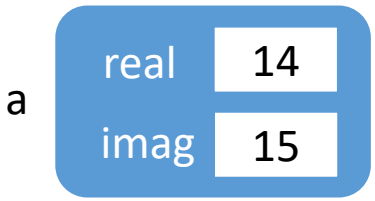
real	24
imag	25

u x y z

```
#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

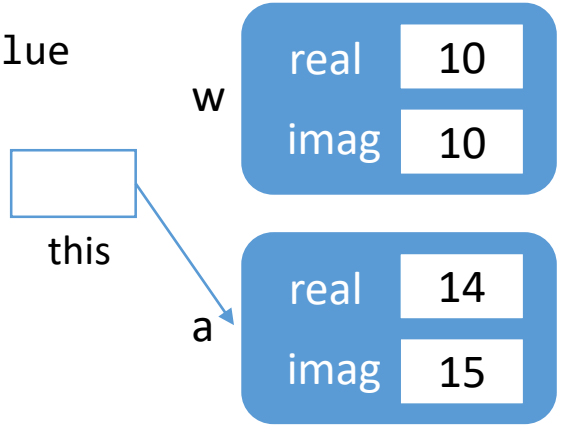
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

( 24 , 25 )
```



```
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 )
```



u x y z

```
#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

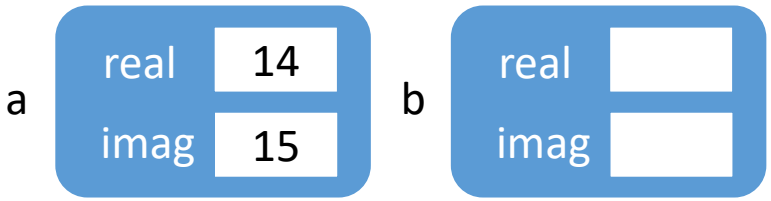
real	14
imag	15

u x y z

```
#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

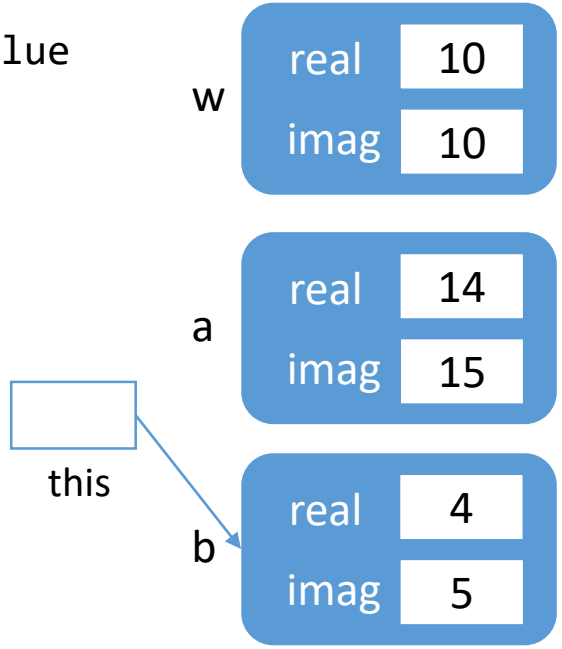
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



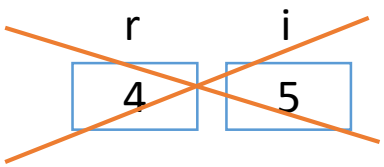
```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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 )
```

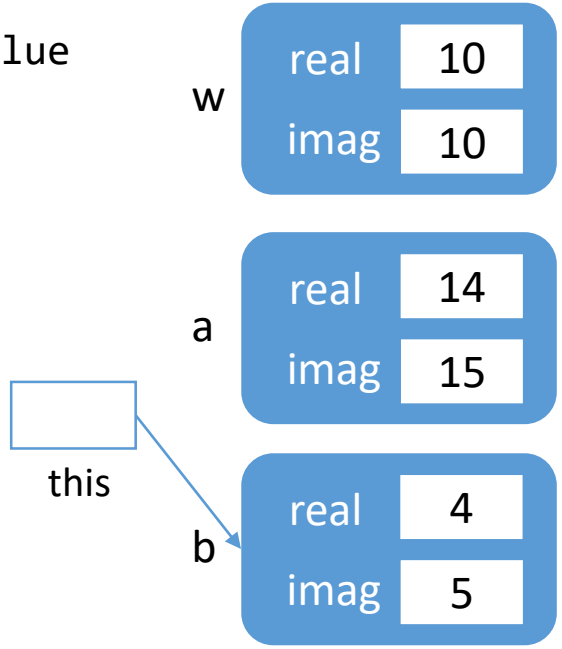


```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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 )
```

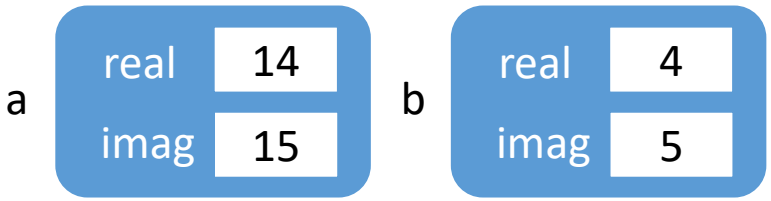


u x y z

```
#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

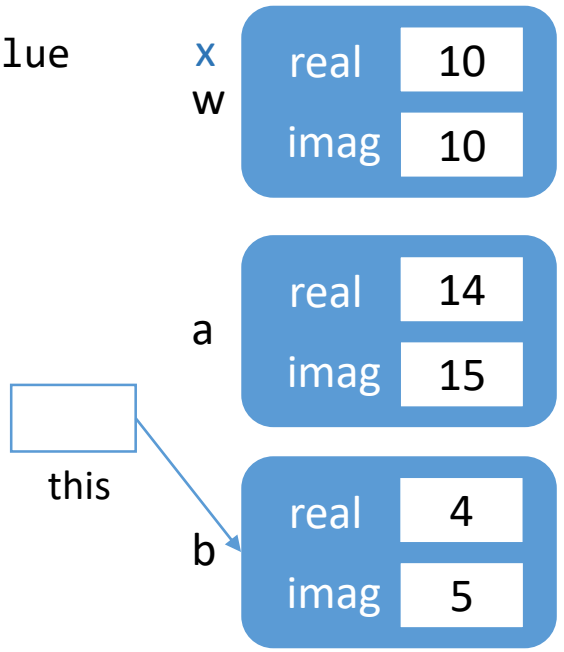
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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

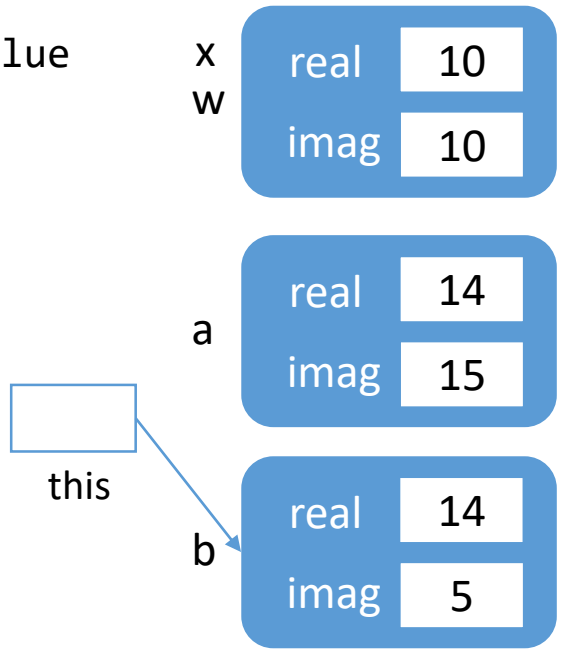
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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

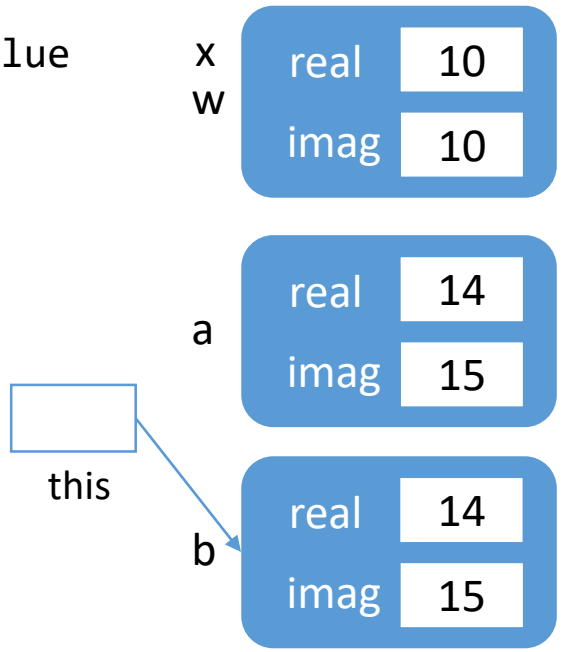
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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

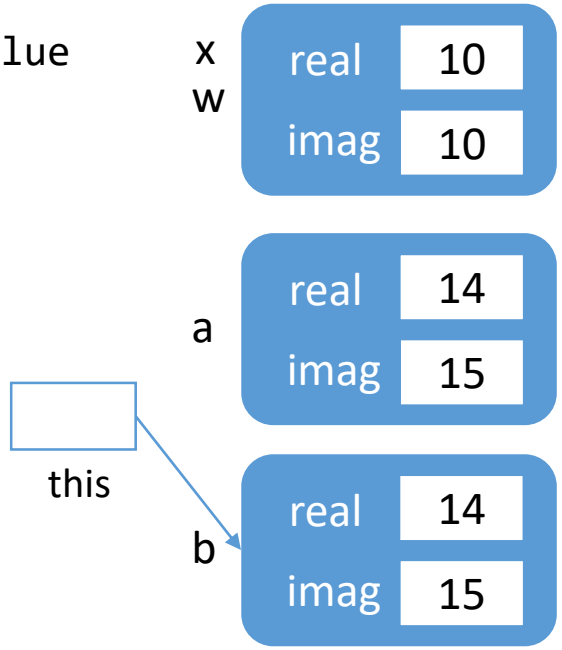
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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 )
```



u x y z

```
#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); c                    ndl;
  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

Output

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

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

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

w

real	10
imag	10

a

real	14
imag	15

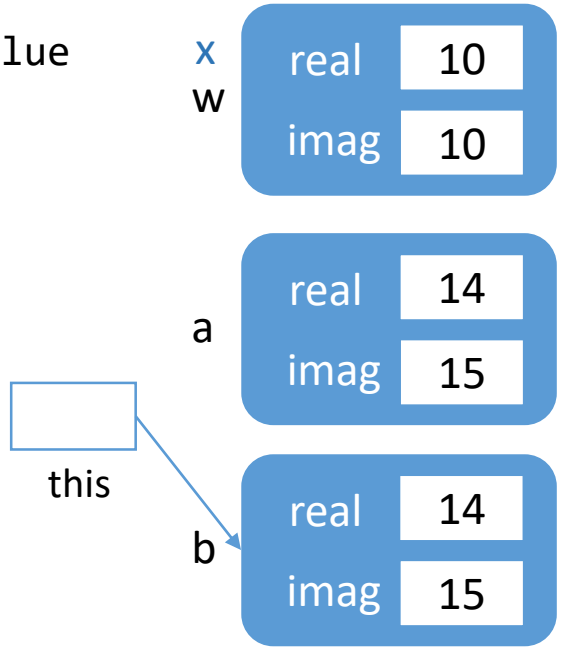
b

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

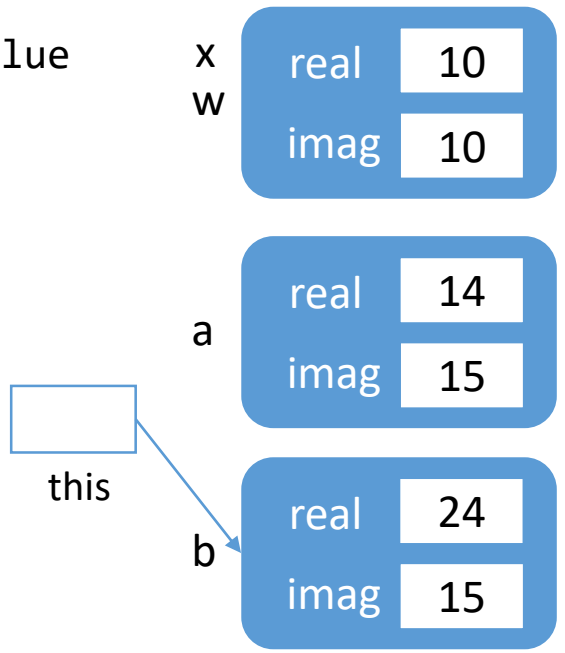
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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

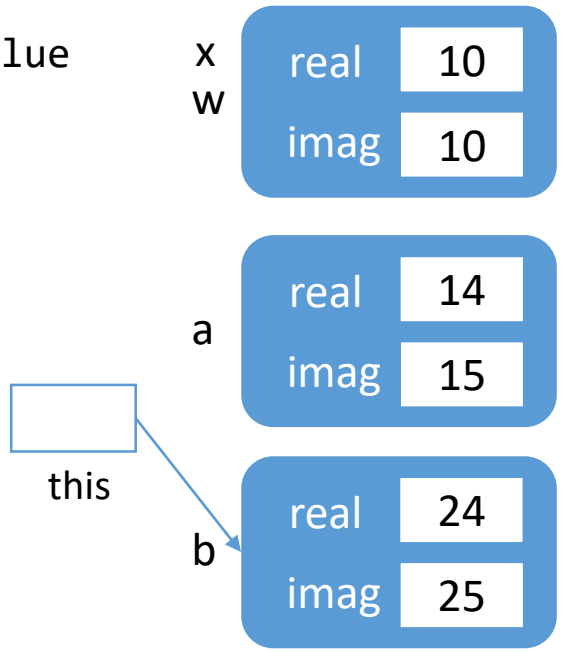
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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

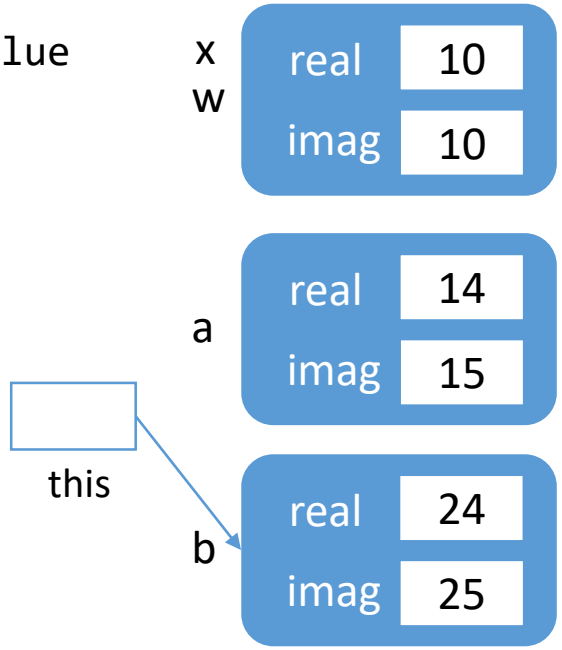
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

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



```
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 )
```



u x y z

```
#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); c                    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

Output

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

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

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

w

real	10
imag	10

a

real	14
imag	15

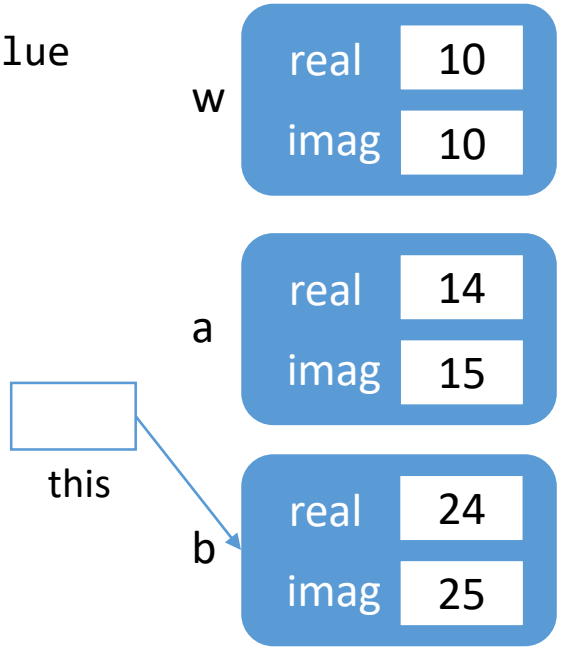
b

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 )
```

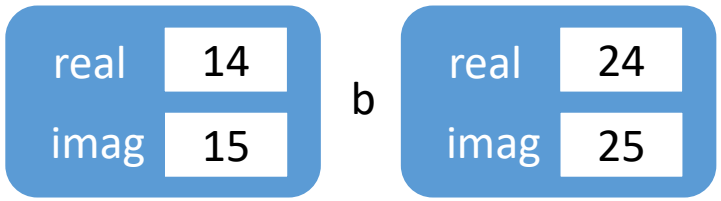


u x y z

```
#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

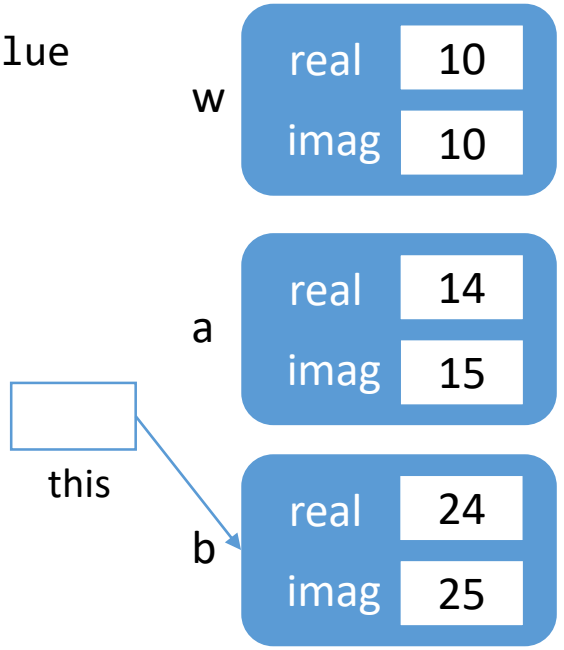
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

( 24 , 25 )
( 14 , 15 )

( 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 )

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



u x y z

```
#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 )

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

Cursor here

w

real	10
imag	10

a

real	14
imag	15

b

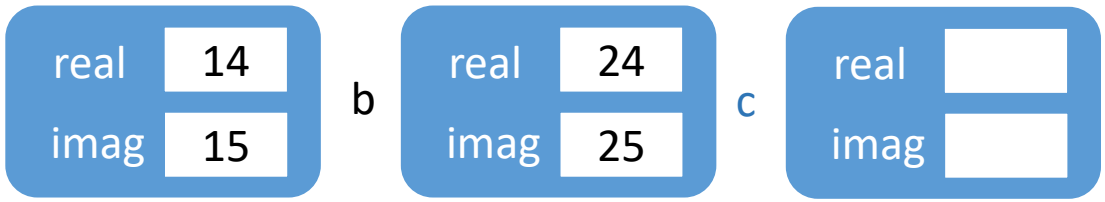
real	24
imag	25

u x y z

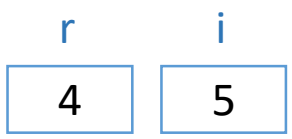
```
#include <iostream> /* File: complex-test.cpp */
using namespace std;
#include "complex.h"

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 )

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

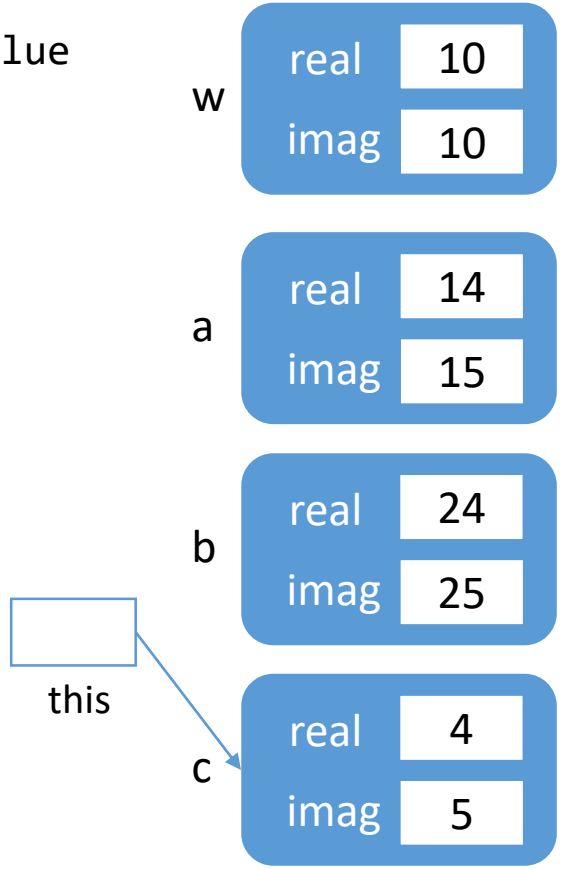


```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```

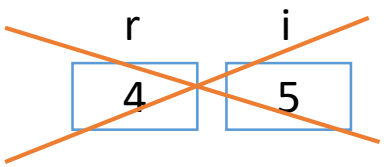


Output

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

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

( 24 , 25 )
( 14 , 15 )

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

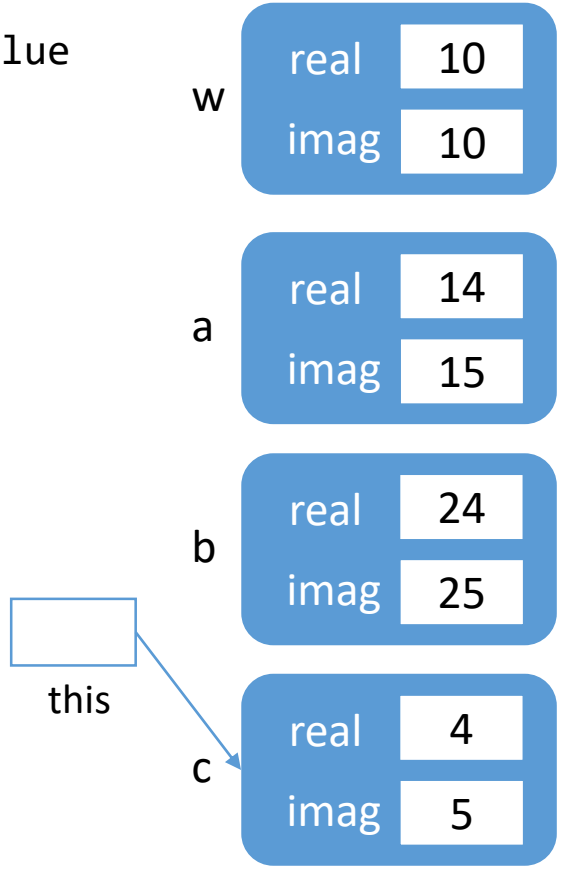


```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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 )

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



u x y z

```
#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 )

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

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

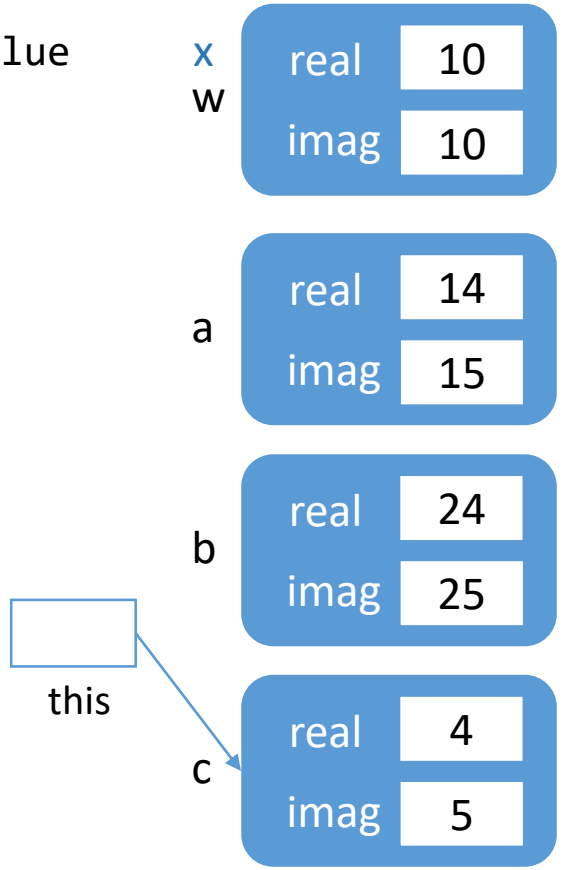
c

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

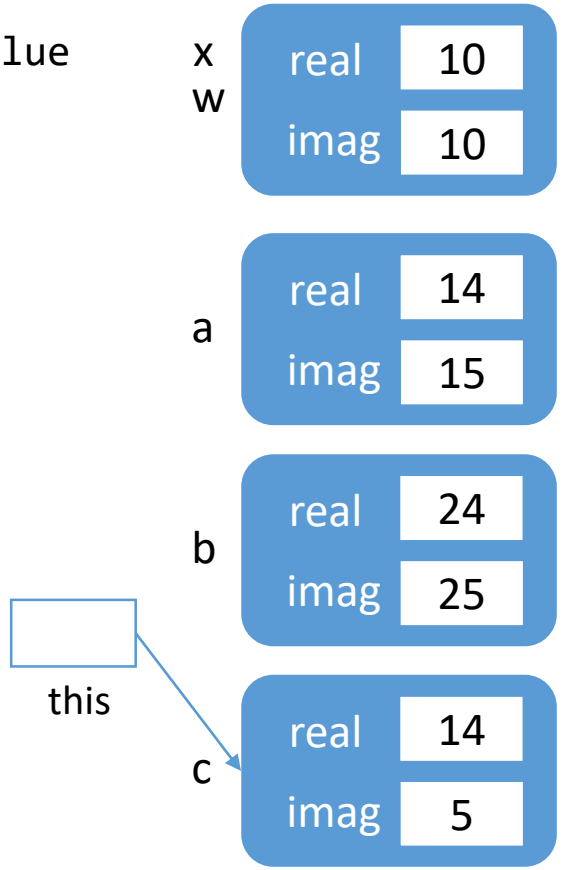
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

( 24 , 25 )
( 14 , 15 )

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



```
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

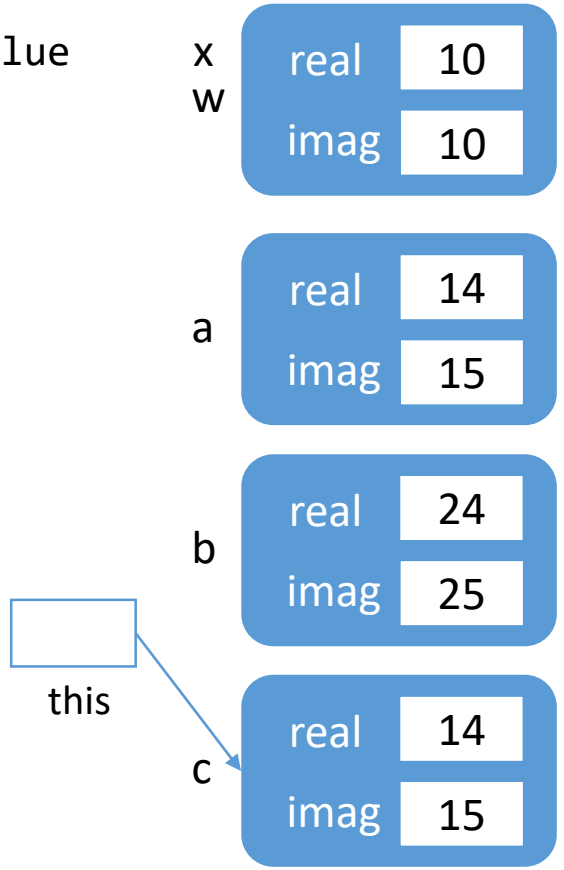
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

( 24 , 25 )
( 14 , 15 )

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



```
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 )
```



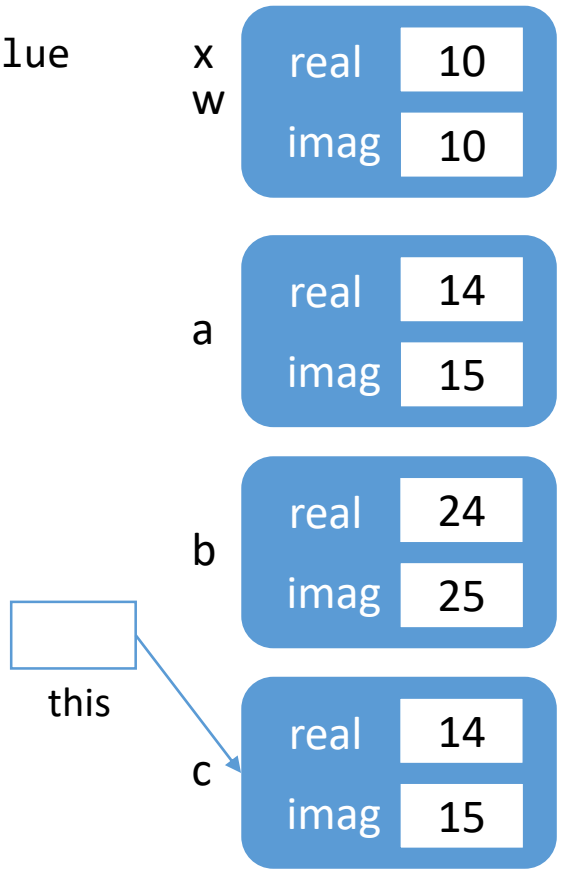
u x y z

```
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 c,
Returning it by reference, i.e. returning c

Output

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

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

( 24 , 25 )
( 14 , 15 )

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



u x y z

```
#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;
}
```

This is c

Output

```
( 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

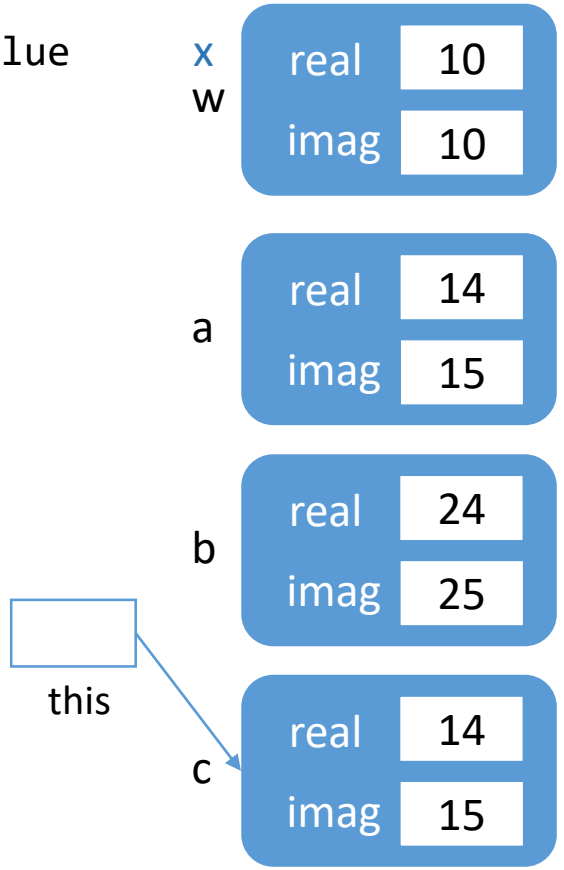
c

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 )
```

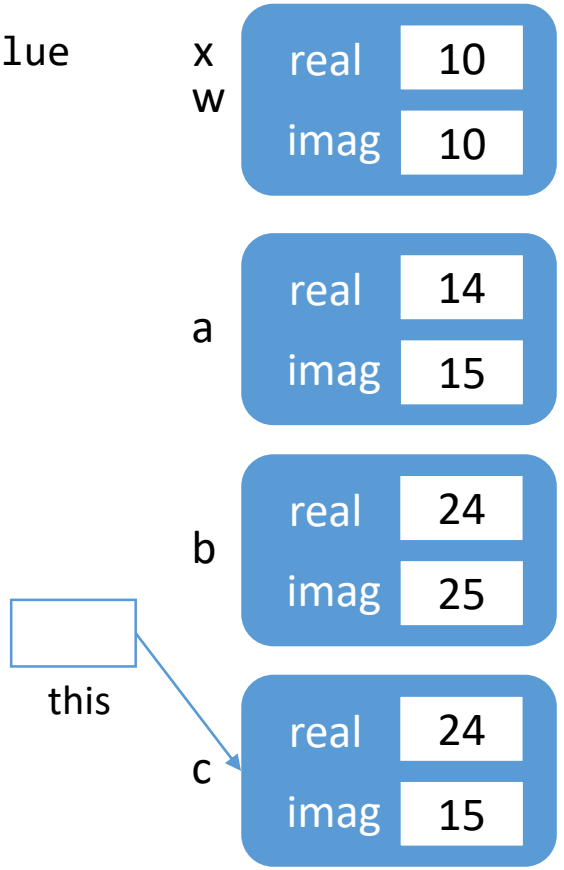


u x y z

```
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

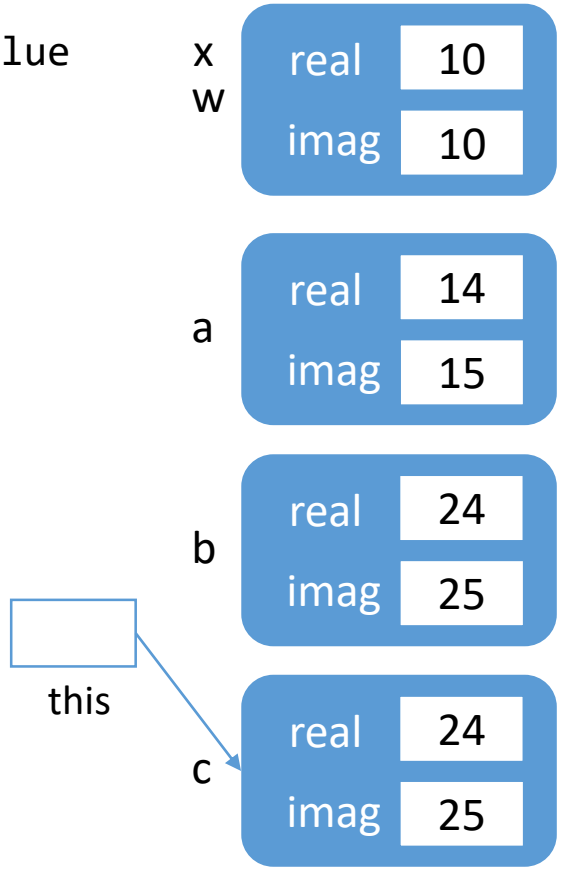
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )

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

( 24 , 25 )
( 14 , 15 )

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



```
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 )
```

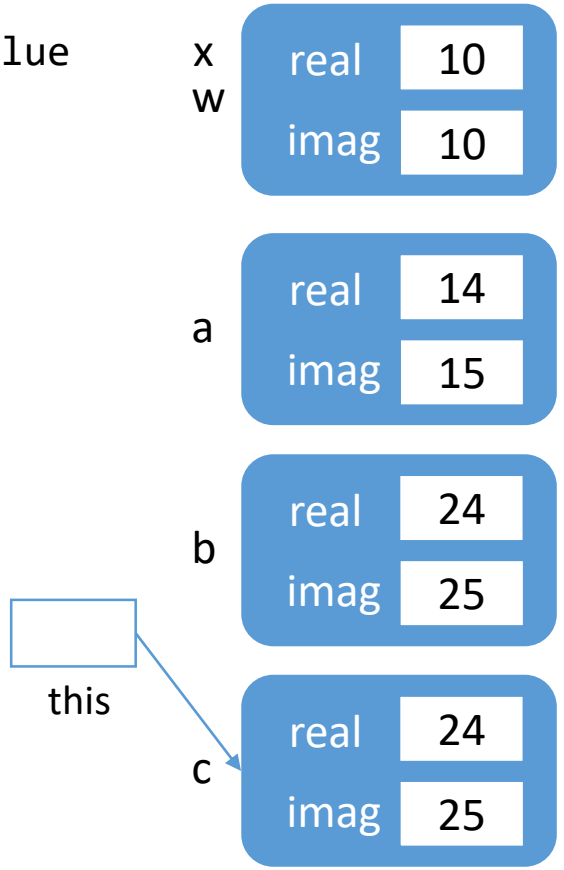


```
class Complex /* File: complex.h */
{
  private:
    float real; float imag;
  public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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 c,
Returning it by reference, i.e. returning c

Output

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

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

( 24 , 25 )
( 14 , 15 )

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



u x y z

```
#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;
}
```

This is c

Output

```
( 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

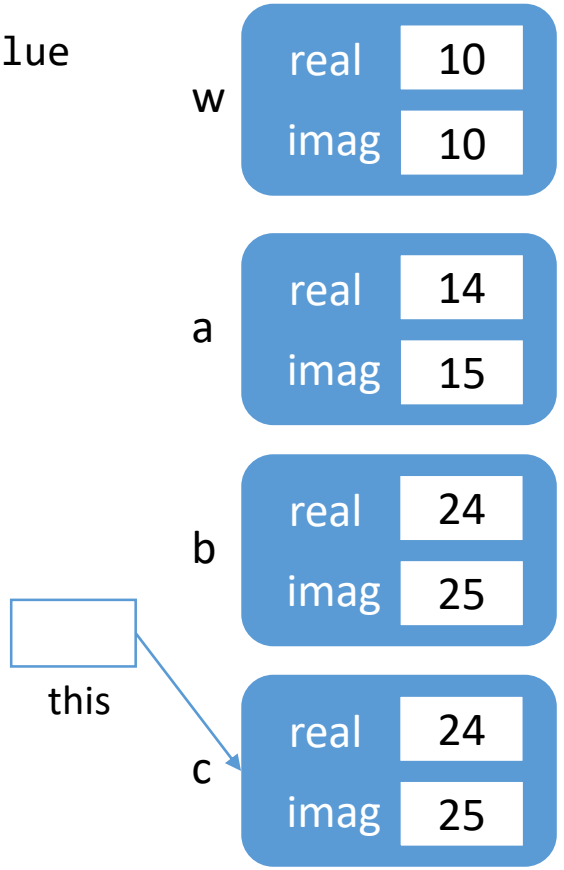
c

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 )
```



u x y z

```
#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 )

( 24 , 25 )
( 24 , 25 )

( 24 , 25 )
```

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

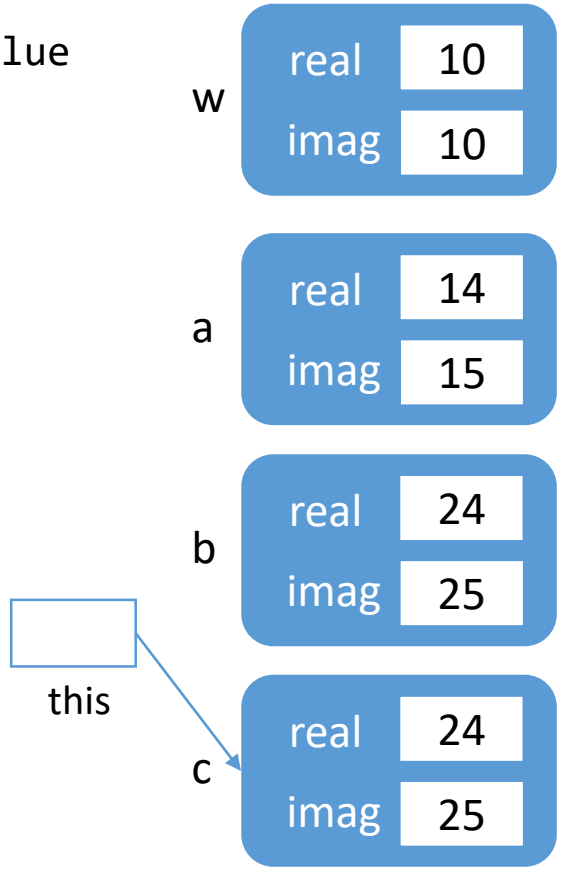
c

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 )
```

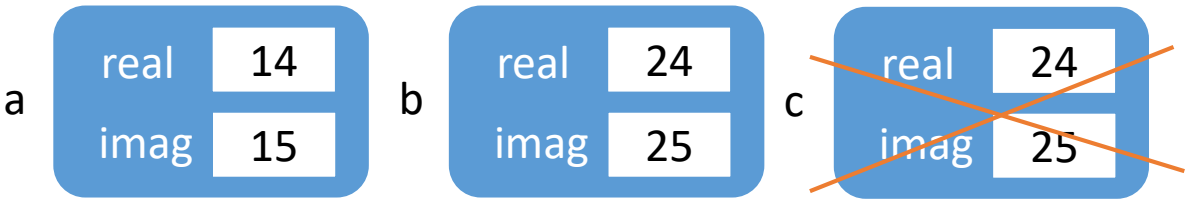


u x y z

```
#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 )

( 24 , 25 )
( 24 , 25 )

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



u x y z

```
#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 )

( 24 , 25 )
( 24 , 25 )

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

w

real	10
imag	10

a

real	14
imag	15

b

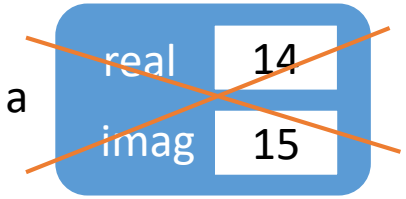
real	24
imag	25

u x y z

```
#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 )

( 24 , 25 )
( 24 , 25 )

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



u x y z

```
#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 )

( 24 , 25 )
( 24 , 25 )

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

w

real	10
imag	10

u x y

```
#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 )

( 24 , 25 )
( 24 , 25 )

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

w

real	10
imag	10

u x

```
#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 )

( 24 , 25 )
( 24 , 25 )

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

w

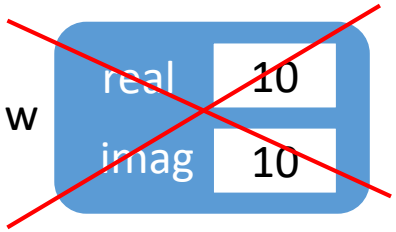
real	10
imag	10

u

```
#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 )

( 24 , 25 )
( 24 , 25 )

( 24 , 25 )
( 24 , 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 )

( 24 , 25 )
( 14 , 15 )

( 24 , 25 )
( 24 , 25 )

( 24 , 25 )
( 24 , 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 )

( 24 , 25 )
( 14 , 15 )

( 24 , 25 )
( 24 , 25 )

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

Done!!! :)