

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

```
#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;
}
```

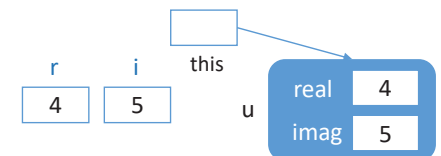


```
class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```

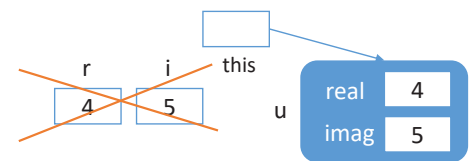
```
class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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);
    }
};
```



```
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;
}
```

```

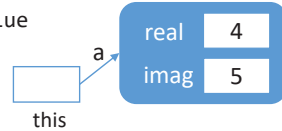
class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

    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)

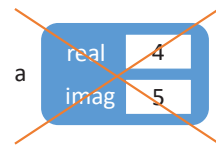
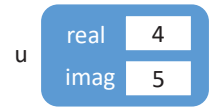
```

#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;
}

```



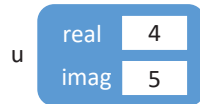
```

#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;
}

```



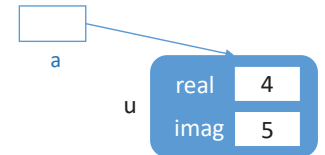
```

#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;
}

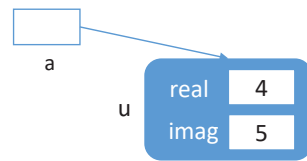
```



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

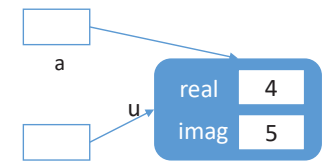
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u

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);
    }
};
```

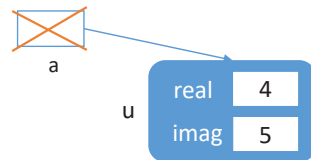


Output
(4 , 5)
(4 , 5)

```
#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"
```

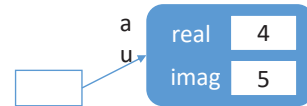
```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
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
```

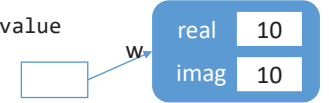
```
int main()
{
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // 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; }

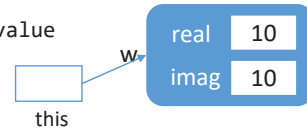
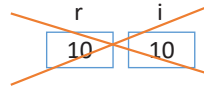
    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x) // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex&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

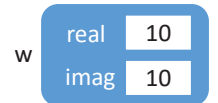
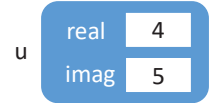


Output
(4 , 5)
(4 , 5)
(4 , 5)

```
#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;
}
```



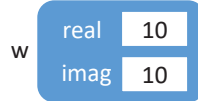
Output
(4 , 5)
(4 , 5)
(4 , 5)

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;
}
```



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

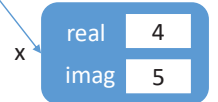
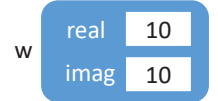
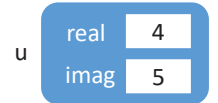
```
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }
```

```
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);
}
```

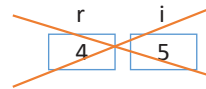
```
};
```



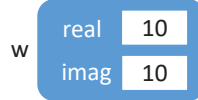
Output

```
( 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;
}
```

Output

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

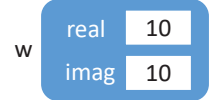
```
Complex& add3(const Complex& x)
// Return by reference
{
    real += x.real; imag += x.imag;
    return (*this);
}
```



```
#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);
    }
};

```

Diagram illustrating the state of objects and pointers during the execution of the `add1` method:

- Object `u` (initial state): `real = 4, imag = 5`
- Object `x` (initial state): `real = 10, imag = 10`
- Object `w` (initial state): `real = 10, imag = 10`
- Object `x` (after `add1`): `real = 4, imag = 5` (copy of `u`)
- Object `w` (after `add1`): `real = 10, imag = 10`
- Object `u` (after `add1`): `real = 4, imag = 5`

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);
    }
};

```

Diagram illustrating the state of objects and pointers during the execution of the `add1` method:

- Object `u` (initial state): `real = 4, imag = 5`
- Object `x` (initial state): `real = 10, imag = 10`
- Object `w` (initial state): `real = 10, imag = 10`
- Object `x` (after `add1`): `real = 14, imag = 5` (copy of `u`)
- Object `w` (after `add1`): `real = 10, imag = 10`
- Object `u` (after `add1`): `real = 4, imag = 5`

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);
    }
};

```

Diagram illustrating the state of objects and pointers during the execution of the `add1` method:

- Object `u` (initial state): `real = 4, imag = 5`
- Object `x` (initial state): `real = 10, imag = 10`
- Object `w` (initial state): `real = 10, imag = 10`
- Object `x` (after `add1`): `real = 14, imag = 15` (copy of `u`)
- Object `w` (after `add1`): `real = 10, imag = 10`
- Object `u` (after `add1`): `real = 4, imag = 5`

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);
    }
};

```

Diagram illustrating the state of objects and pointers during the execution of the `add1` method:

- Object `u` (initial state): `real = 4, imag = 5`
- Object `x` (initial state): `real = 10, imag = 10`
- Object `w` (initial state): `real = 10, imag = 10`
- Object `x` (after `add1`): `real = 14, imag = 15` (copy of `u`)
- Object `w` (after `add1`): `real = 10, imag = 10`
- Object `u` (after `add1`): `real = 4, imag = 5`

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(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	4
imag	5

temp

real	14
imag	15

u

real	4
imag	5

w

real	10
imag	10

This is temp

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

```
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "(" << real << " , " << imag << ")" << endl; }

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

u

real	4
imag	5

w

real	10
imag	10

x

real	14
imag	15

temp

real	14
imag	15

Output

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

this

temp

```
#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(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

x

real	14
imag	15

temp

real	14
imag	15

u

real	4
imag	5

w

real	10
imag	10

This is temp

```
#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;
}
```

x

real	14
imag	15

y

real	
imag	

u

real	4
imag	5

w

real	10
imag	10

```

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);
    }
};

```

Diagram illustrating memory state for the first version of the code:

- Variable `u` (local to `main`) contains `real: 4, imag: 5`.
- Variable `w` (local to `main`) contains `real: 10, imag: 10`.
- Variable `x` (local to `main`) contains `real: 14, imag: 15`.
- Variable `y` (local to `main`) contains `real: 4, imag: 5`.
- The `this` pointer in `u` points to `y`.

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);
    }
};

```

Diagram illustrating memory state for the second version of the code (with `r` and `i` crossed out):

- Variable `u` (local to `main`) contains `real: 4, imag: 5`.
- Variable `w` (local to `main`) contains `real: 10, imag: 10`.
- Variable `x` (local to `main`) contains `real: 14, imag: 15`.
- Variable `y` (local to `main`) contains `real: 4, imag: 5`.
- The `this` pointer in `u` points to `y`.

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;
}

```

Diagram illustrating memory state for the test program:

- Variable `u` (local to `main`) contains `real: 4, imag: 5`.
- Variable `w` (local to `main`) contains `real: 10, imag: 10`.
- Variable `x` (local to `main`) contains `real: 14, imag: 15`.
- Variable `y` (local to `main`) contains `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);
    }
};

```

Diagram illustrating memory state for the second version of the code (with `r` and `i` crossed out):

- Variable `u` (local to `main`) contains `real: 4, imag: 5`.
- Variable `w` (local to `main`) contains `real: 10, imag: 10`.
- Variable `x` (local to `main`) contains `real: 14, imag: 15`.
- Variable `y` (local to `main`) contains `real: 4, imag: 5`.
- The `this` pointer in `u` points to `y`.

Output: `(4 , 5)`, `(4 , 5)`, `(4 , 5)`, `(14 , 15)`

```

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

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

```

Diagram illustrating the state of variables and memory:

- Variable **u** (Complex object): real = 4, imag = 5
- Variable **x** (Complex object): real = 10, imag = 10
- Variable **w** (Complex object): real = 10, imag = 10
- Variable **x** (Complex object): real = 14, imag = 15
- Variable **y** (Complex object): real = 14, imag = 5

Output:

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
( 14 , 15 )

```

Diagram showing the 'this' pointer pointing to the object represented by **x** (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);
    }
};

```

Diagram illustrating the state of variables and memory:

- Variable **u** (Complex object): real = 4, imag = 5
- Variable **x** (Complex object): real = 10, imag = 10
- Variable **w** (Complex object): real = 10, imag = 10
- Variable **x** (Complex object): real = 14, imag = 15
- Variable **y** (Complex object): real = 14, imag = 15

Output:

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
( 14 , 15 )

```

Diagram showing the 'this' pointer pointing to the object represented by **x** (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);
    }
};

```

Diagram illustrating the state of variables and memory:

- Variable **u** (Complex object): real = 4, imag = 5
- Variable **x** (Complex object): real = 10, imag = 10
- Variable **w** (Complex object): real = 10, imag = 10
- Variable **x** (Complex object): real = 14, imag = 15
- Variable **y** (Complex object): real = 14, imag = 15

Output:

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
( 14 , 15 )

```

Diagram showing the 'this' pointer pointing to the object represented by **x** (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;
}

```

Diagram illustrating the state of variables and memory:

- Variable **u** (Complex object): real = 4, imag = 5
- Variable **w** (Complex object): real = 10, imag = 10
- Variable **x** (Complex object): real = 14, imag = 15
- Variable **y** (Complex object): real = 14, imag = 15

Diagram showing the 'this' pointer pointing to the object represented by **x** (real=14, imag=15).

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

```

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

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

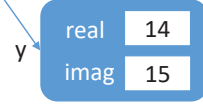
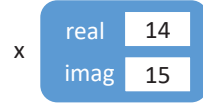
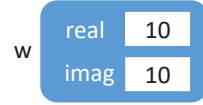
```

Output

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

(14 , 15)
(14 , 15)

this



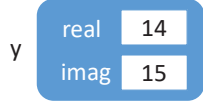
```

#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);
    }
};

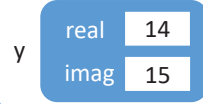
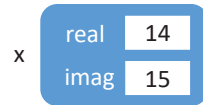
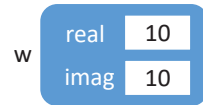
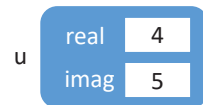
```

Output

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

(14 , 15)
(14 , 15)

this



```

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

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

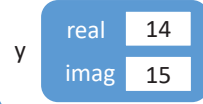
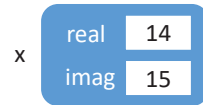
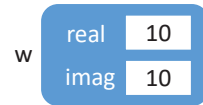
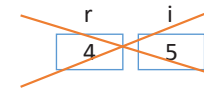
```

Output

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

(14 , 15)
(14 , 15)

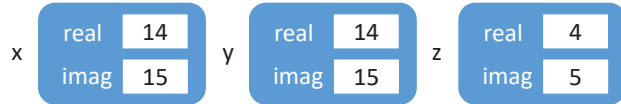
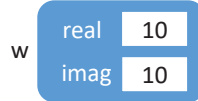
this



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

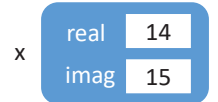
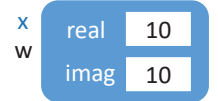
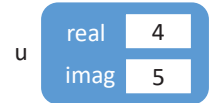
```
void f(const Complex a) { a.print(); } // const Complex a = u
void g(const Complex* a) { a->print(); } // const Complex* a = &u
void h(const Complex& a) { a.print(); } // const Complex& a = u
```

```
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;
}
```

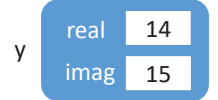


```
class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

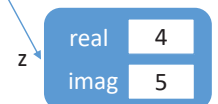
    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x)
    // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex&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



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

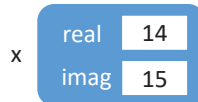
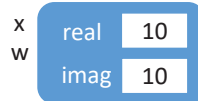
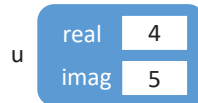


(14 , 15)
(14 , 15)

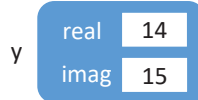


```
class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

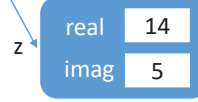
    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x)
    // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex&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



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

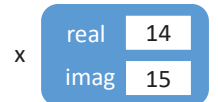
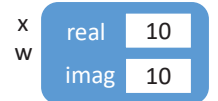
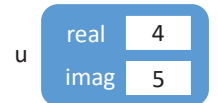


(14 , 15)
(14 , 15)

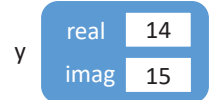


```
class Complex /* File: complex.h */
{
private:
    float real; float imag;
public:
    Complex(float r, float i) { real = r; imag = i; }
    void print() { cout << "( " << real << " , " << imag << " )" << endl; }

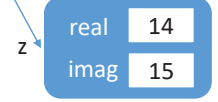
    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x)
    // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex&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



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



(14 , 15)
(14 , 15)



```

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

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

```

Output

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )

```

Diagram illustrating the state of variables u, w, x, y, and z. Each variable is represented by a box with 'real' and 'imag' components. u and w are initialized to (4, 5). x, y, and z are initialized to (10, 10). After the execution of the code, x, y, and z all contain the value (14, 15). A callout box points to the 'return (*this);' line in the add3 method, stating: '*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 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;
}

```

Diagram illustrating the state of variables u, w, x, y, and z. Each variable is represented by a box with 'real' and 'imag' components. u and w are initialized to (4, 5). x, y, and z are initialized to (10, 10). After the execution of the code, x, y, and z all contain the value (14, 15). A callout box points to the 'z.add3(w)' line in the main function, stating: 'This is 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 )

```

Diagram illustrating the state of variables u, w, x, y, and z. Each variable is represented by a box with 'real' and 'imag' components. u and w are initialized to (4, 5). x, y, and z are initialized to (10, 10). After the execution of the code, x, y, and z all contain the value (14, 15). A callout box points to the 'return (*this);' line in the add3 method, stating: '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;
}

```

Diagram illustrating the state of variables u, w, x, y, and z. Each variable is represented by a box with 'real' and 'imag' components. u and w are initialized to (4, 5). x, y, and z are initialized to (10, 10). After the execution of the code, x, y, and z all contain the value (14, 15). A callout box points to the 'z.add3(w)' line in the main function, stating: 'Cursor here'.

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"
```

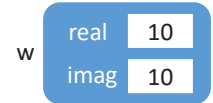
```
void f(const Complex a) { a.print(); }
void g(const Complex* a) { a->print(); }
void h(const Complex& a) { a.print(); }
```

```
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



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; }

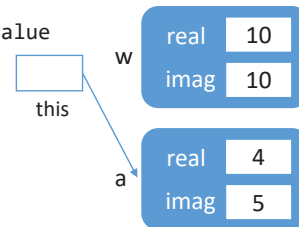
    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x)
    // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex&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



Output

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

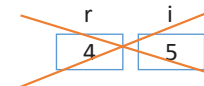
```
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
```



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; }

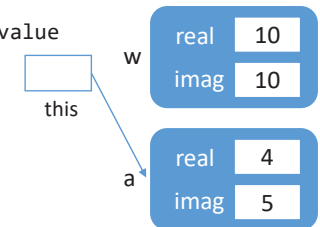
    Complex add1(const Complex& x) // Return by value
    {
        real += x.real; imag += x.imag;
        return (*this);
    }
    Complex* add2(const Complex& x)
    // Return by value using pointer
    {
        real += x.real; imag += x.imag;
        return this;
    }
    Complex& add3(const Complex&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



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()
{
```

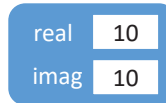
```
    // Check the parameter passing methods
    Complex u(4, 5); f(u); g(&u); h(u);
    // Check the parameter returning methods
    Complex w(10, 10); cout << endl << endl;
    Complex x(4, 5); (x.add1(w)).print(); // Complex temp = *this = x
    Complex y(4, 5); (y.add2(w))->print(); // Complex* 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

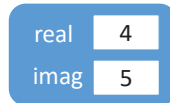
```
( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
```

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

w



a



```
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
```

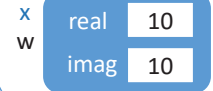
```
{
    real += x.real; imag += x.imag;
    return (*this);
}
```

};

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 )
```

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);
}
```

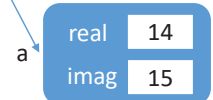
};

Output

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

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

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);
    }
};

```

Output

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

(14 , 15)
(14 , 15)

Diagram illustrating the state of variables x, w, a, and temp1. x and w are Complex objects with real=10, imag=10. a is a Complex object with real=14, imag=15. temp1 is a Complex object with real=14, imag=15. A box labeled 'this' points to the 'return (*this);' statement in the add1 function.

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 = w.add1(w); cout << "d1 << 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)

Diagram illustrating the state of variables a and temp1. a is a Complex object with real=14, imag=15. temp1 is a Complex object with real=14, imag=15. A box labeled 'this' points to the 'return (*this);' statement in the add1 function.

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)

Diagram illustrating the state of variables x, w, a, and temp1. x and w are Complex objects with real=10, imag=10. a is a Complex object with real=14, imag=15. temp1 is a Complex object with real=14, imag=15. A box labeled 'this' points to the 'return (*this);' statement in the add1 function.

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)

Diagram illustrating the state of variables a and temp1. a is a Complex object with real=14, imag=15. temp1 is a Complex object with real=24, imag=15. A box labeled 'this' points to the 'return (*this);' statement in the add1 function.

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);
    }
};

```

Diagram illustrating the state of variables and objects during the execution of the `add1` method:

- Object x:** real = 10, imag = 10
- Object w:** real = 14, imag = 15
- Object a:** real = 14, imag = 15
- Object temp1:** real = 24, imag = 25

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);
    }
};

```

Diagram illustrating the state of variables and objects during the execution of the `add1` method:

- Object x:** real = 10, imag = 10
- Object w:** real = 14, imag = 15
- Object a:** real = 14, imag = 15
- Object temp1:** real = 24, imag = 25
- Object temp2:** real = 24, imag = 25

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); c = w.add1(w); // This is temp2
    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;
}

```

Diagram illustrating the state of variables and objects during the execution of the `main` function:

- Object w:** real = 10, imag = 10
- Object a:** real = 14, imag = 15
- Object temp1:** real = 24, imag = 25
- Object temp2:** real = 24, imag = 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);
    }
};

```

Diagram illustrating the state of variables and objects during the execution of the `main` function:

- Object w:** real = 10, imag = 10
- Object a:** real = 14, imag = 15
- Object temp1:** real = 24, imag = 25
- Object 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; // This is temp2 endl;
```

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

```
    Complex a(4, 5); a.add1(w).add1(w).print(); a.print(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

Output

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

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

(24 , 25)

w

real	10
imag	10

a

real	14
imag	15

temp1

real	24
imag	25

temp2

real	24
imag	25

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 )
```

w

real	10
imag	10

a

real	14
imag	15

this

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)

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)

w

real	10
imag	10

a

real	14
imag	15

Cursor here

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 )
```

w

real	10
imag	10

a

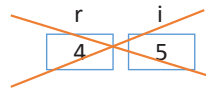
real	14
imag	15

b

real	
imag	

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 )
```

w

real	10
imag	10

a

real	14
imag	15

b

real	4
imag	5

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; }
```

r	i
4	5

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

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

Output

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

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

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

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

this

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

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

};

w

real	10
imag	10

a

real	14
imag	15

b

real	4
imag	5

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(); }
```

Output

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

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

```
int main()
```

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

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

w

real	10
imag	10

a

real	14
imag	15

b

real	4
imag	5

```

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

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

```

Output

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

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

(24 , 25)
(14 , 15)

```

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

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)

x

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	15

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);

}

};

Output

(4 , 5)

(4 , 5)

(4 , 5)

(14 , 15)

(14 , 15)

(14 , 15)

(24 , 25)

(14 , 15)

x

real	10
imag	10

a

real	14
imag	15

b

real	14
imag	15

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);

}

};

Output

(4 , 5)

(4 , 5)

(4 , 5)

(14 , 15)

(14 , 15)

(14 , 15)

(24 , 25)

(14 , 15)

x

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

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);
    }
};

```

Output

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

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

(24 , 25)
(14 , 15)

x real 10
imag 10

w real 10
imag 10

a real 14
imag 15

b real 24
imag 25

this

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

real 10
imag 10

a real 14
imag 15

b real 24
imag 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)

w real 10
imag 10

a real 14
imag 15

b real 24
imag 25

this

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)

real 10
imag 10

a real 14
imag 15

b real 24
imag 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 )

```

w

a

b

this

real	10
imag	10

real	14
imag	15

real	24
imag	25

Output

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;
}

```

Cursor here

```

( 24 , 25 )
( 14 , 15 )

```

```

( 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 )

```

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	
imag	

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);
    }
};

```

r

4

i

5

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	4
imag	5

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);
    }
};

```

Diagram showing memory state after `add1` and `add2` calls. A box with `r=4` and `i=5` is crossed out. Variable `w` has `real=10, imag=10`. Variable `a` has `real=14, imag=15`. Variable `b` has `real=24, imag=25`. Variable `c` has `real=4, imag=5`. A box labeled `this` points to `c`.

Output:

```

( 4 , 5 )
( 4 , 5 )
( 4 , 5 )
( 14 , 15 )
( 14 , 15 )
( 14 , 15 )
( 24 , 25 )
( 14 , 15 )
( 24 , 25 )
( 24 , 25 )

```

Output

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;
}

```

Diagram showing memory state after `add3` call. Variable `w` has `real=10, imag=10`. Variable `a` has `real=14, imag=15`. Variable `b` has `real=24, imag=25`. Variable `c` has `real=4, imag=5`. A box labeled `this` points to `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

```

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);
    }
};

```

Diagram showing memory state after `add1` and `add2` calls. Variable `x` has `real=10, imag=10`. Variable `w` has `real=10, imag=10`. Variable `a` has `real=14, imag=15`. Variable `b` has `real=24, imag=25`. Variable `c` has `real=4, imag=5`. A box labeled `this` points to `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

```

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);
    }
};

```

Diagram showing memory state after `add3` call. Variable `x` has `real=10, imag=10`. Variable `w` has `real=10, imag=10`. Variable `a` has `real=14, imag=15`. Variable `b` has `real=24, imag=25`. Variable `c` has `real=14, imag=5`. A box labeled `this` points to `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

```

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)

x w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

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);
    }
};

```

Output

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

x w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

this

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

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)

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

This is c

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)

x w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	14
imag	15

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);
    }
};

```

Output

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

x w
real 10
imag 10

a
real 14
imag 15

b
real 24
imag 25

c
real 24
imag 15

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);
    }
};

```

Output

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

x w
real 10
imag 10

a
real 14
imag 15

b
real 24
imag 25

c
real 24
imag 25

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);
    }
};

```

Output

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

x w
real 10
imag 10

a
real 14
imag 15

b
real 24
imag 25

c
real 24
imag 25

this

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

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)

w
real 10
imag 10

a
real 14
imag 15

b
real 24
imag 25

c
real 24
imag 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

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

this

Output

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	24
imag	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

w

real	10
imag	10

a

real	14
imag	15

b

real	24
imag	25

c

real	24
imag	25

this

Output

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	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(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

a	real	14
	imag	15

b	real	24
	imag	25

w	real	10
	imag	10

Output

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

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

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

a	real	14
	imag	15

w	real	10
	imag	10

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(); cout << endl;
    Complex b(4, 5); b.add2(w)->add2(w)->print(); b.print(); cout << endl;
    Complex c(4, 5); c.add3(w).add3(w).print(); c.print();
    return 0;
}
```

w	real	10
	imag	10

Output

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

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

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

w	real	10
	imag	10

u

Output

real	10
imag	10

Output

~~w~~

Output

Output

Done!!! :)