Day 10

27 March 2025 08:24 AM

→ Operator overloading :→

int + int

float + float

char + char

Complex c_1, c_2

$c_1 + c_2 \rightarrow c_1.sum(c_2)$ ✗

□ + □

* one operand must be user defined (object)
* Never change precedence & associativity
* :: , . , -> , * , sizeof( )

  * → deference

Unary operators

+ − ++ −− sizeof typecasting

+ → sign op ✗

− → sign change

+5

−5

int a = 5;

a = −a;

a | 5 | −5

−(+5)

−5

```
#include<iostream>
using namespace std;
class data{
    int a;
    public:
        data(int a1=0)
        {
            a=a1;
        }
        data signchange()
        {
            data temp;
            temp.a =  a*-1;
            return temp;
        }
```

```
        data operator-()
        {
            data temp;
            temp.a =  a*-1;
            return temp;
        }
        void output()
        {
            cout<<a<<endl;
        }
};
int main()
{
    data d1=10,d2;
    d1.output();
    // d2=d1.signchange();
    d2 = -d1;
    d2.output();
    d1.output();
    return 0;
}
```

++

int a = 5;

a 6

b a = ++a;

a = a+1

b = ++a

```
#include<iostream>
using namespace std;
class data{
    int a;
    public:
        data(int a1=0)
        {
            a=a1;
        }
        data increment()
        {
            data temp;
            a = a+1;
            temp.a = a;
            return temp;
        }
        data operator ++()      //pre
        {
            data temp;
            a = a+1;
            temp.a = a;
            return temp;
        }
        data operator ++(int)      //post
        {
            data temp;
            temp.a = a;
            a = a+1;
            return temp;
        }
        void output()
        {
            cout<<a<<endl;
```

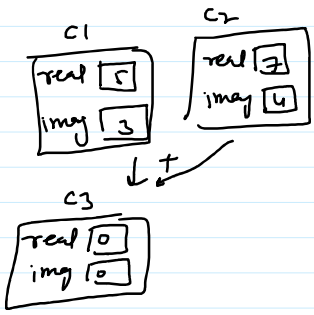
```
        }
};
int main()
{
    data d1=10,d2;
    d1.output();        //10
    // d2=d1.increment();
    d2 = ++d1;

    d2.output();        //11
    d1.output();        //11
    d2 = d1++;
    d2.output();        //11
    d1.output();        //12
    return 0;
}
```

+ — * / %

Complex c1(5,3), c2(7,4), c3;

c3 = c1 + c2;

c3.output(); → (12+7i)




```
#include<iostream>
using namespace std;
class complex{
    public:
        complex(int r=0, int i=0)
        {
            real = r;
            imag = i;
        }
        void output()
        {
            if(imag >= 0)
                cout<<real<<"+"<<imag<<"i"<<endl;
            else
                cout<<real<<imag<<"i"<<endl;
        }
        complex sum(complex obj)
        {
            complex temp;
            temp.real = real + obj.real;
            temp.imag = imag + obj.imag;
            return temp;
        }
        complex operator +(complex &obj)
        {
            complex temp;
            temp.real = real + obj.real;
            temp.imag = imag + obj.imag;
            return temp;
        }
    private:
        int real,imag;
};
int main()
{
    complex c1(5,7),c2(4,3);
    c1.output();
    c2.output();
    complex c3;
```

member func of class Complex

this [c1]

obj

temp

calling obj

```
    complex c1(5,7),c2(4,3);
    c1.output();
    c2.output();
    complex c3;
    // c3 = c1.sum(c2);
    c3 = c1+c2;
    c3.output();
    return 0;
}
```

calling obj

$c_3 = c_1 + c_2$ ← Argument

$c_3 = c_1 * c_2;$

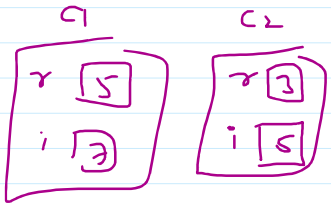
c_3.output()

```
#include<iostream>
using namespace std;
class complex{
    public:
        complex(int r=0, int i=0)
        {
            real = r;
            imag = i;
        }
        void input()
        {
            cout<<"Enter real:";
            cin>>real;
            cout<<"Enter Imag:";
            cin>>imag;
        }
        void output()
        {
            if(imag >= 0)
                cout<<real<<"+"<<imag<<"i"<<end
l;
            else
                cout<<real<<imag<<"i"<<endl;
        }
        complex sum(complex obj)
        {
            complex temp;
            temp.real = real + obj.real;
            temp.imag = imag + obj.imag;
            return temp;
        }
        complex operator +(complex &obj)
        {
            complex temp;
            temp.real = real + obj.real;
            temp.imag = imag + obj.imag;
            return temp;
        }
        complex operator*(complex &obj)
        {
            complex temp;
            temp.real = real*obj.real - imag *
obj.imag;
            temp.imag = real * obj.imag + imag
* obj.real;
            return temp;
        }
    private:
        int real,imag;
};
```

```
int main()
{
    complex c1,c2;
    c1.input();
    c2.input();
    c1.output();
    c2.output();
    complex c3;
    // c3 = c1.sum(c2);
    c3 = c1+c2;
    c3.output();
    c3 = c1*c2+c3;
    c3.output();
    return 0;
}
```

>, <, >=, <=, ==, !=

c1 > c2



5 + 7i 3 + 4i

5 + 3i 4 + 5i

```
#include<iostream>
using namespace std;
class complex{
    public:
        complex(int r=0, int i=0)
        {
            real = r;
            imag = i;
        }
        void input()
        {
            cout<<"Enter real:";
            cin>>real;
            cout<<"Enter Imag:";
            cin>>imag;
        }
        void output()
        {
            if(imag >= 0)
                cout<<real<<"+"<<imag<<"i"<<endl;
            else
                cout<<real<<imag<<"i"<<endl;
        }
        complex sum(complex obj)
        {
```

```
            complex temp;
            temp.real = real + obj.real;
            temp.imag = imag + obj.imag;
            return temp;
        }
        complex operator +(complex &obj)
        {
            complex temp;
            temp.real = real + obj.real;
            temp.imag = imag + obj.imag;
            return temp;
        }
        complex operator*(complex &obj)
        {
            complex temp;
            temp.real = real*obj.real - imag * obj.imag;
            temp.imag = real * obj.imag + imag * obj.real;
            return temp;
        }
        bool operator>(complex &obj)
        {
            if(real > obj.real and imag >obj.imag)
                return true;
            else
                return false;
        }
        bool operator<(complex &obj)
        {
            if(real < obj.real and imag < obj.imag)
                return true;
            else
                return false;
        }
        bool operator==(complex &obj)
        {
            if(real == obj.real and imag == obj.imag)
                return true;
            else
                return false;
        }
        bool operator >=(complex &obj)
        {
            if(real > obj.real && imag>obj.imag || real==obj.real && imag== obj.imag)
                return true;
            return false;
        }
        bool operator <=(complex &obj)
        {
            if(real < obj.real && imag < obj.imag || real==obj.real && imag== obj.imag)
                return true;
            return false;
        }
    private:
        int real,imag;
};
int main()
{
    complex c1,c2;
    c1.input();
    c2.input();
    c1.output();
    c2.output();
```

```
    complex c3;
    // c3 = c1.sum(c2);
    c3 = c1+c2;
    c3.output();
    c3 = c1*c2+c3;
    c3.output();
    if(c1>c2)
        cout<<"C1 is greater";
    else
        cout<<"C1 is not greater";
    return 0;
}
```

- 44 - 11 !

✓ c1 > c2 && c3 > c4

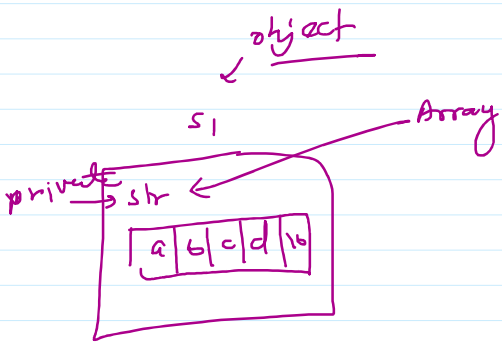
[ ] → Subscript operator

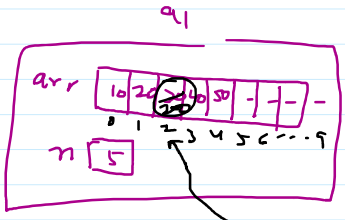
object

string s1 = "abcd";

cout<< s1[0]; → a

s1.~~str~~[0]





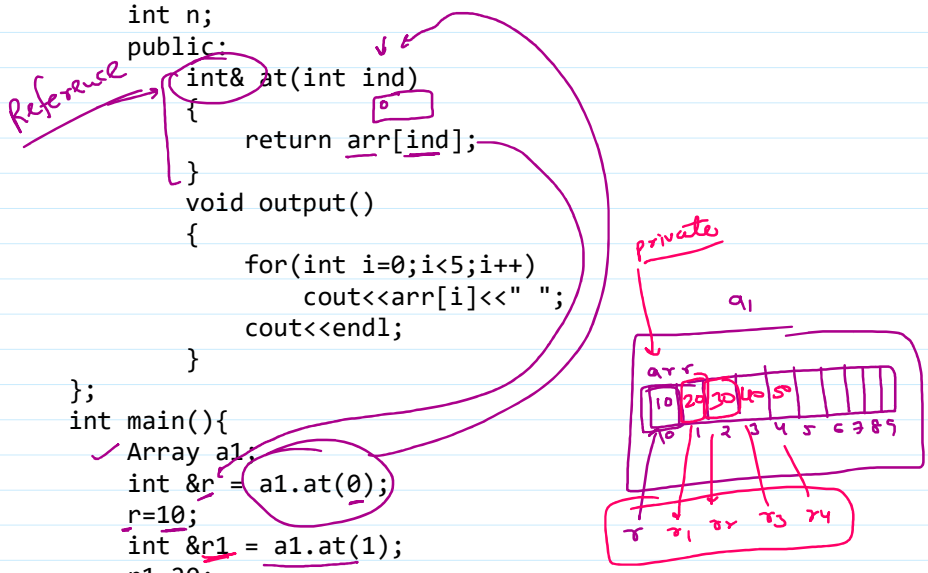

```
class Array
{
    int arr[10];
    int n;

    int& at(int ind)
    {
        return arr[ind];
    }
};
```

```
int main ()
{
    Array a1;
    a1.input();
    cout<< a1[0]; → Error
    a1.~~arr~~[0]
    int& x = a1.at(2)
    cout<<x; → 30
    x=200;
```

value (30)

```
#include<iostream>
using namespace std;
class Array{
```

```
    int arr[10];
    int n;
    public:
        int& at(int ind)
        {
            return arr[ind];
        }
        void output()
        {
            for(int i=0;i<5;i++)
                cout<<arr[i]<<" ";
            cout<<endl;
        }
};
int main(){
    Array a1;
    int &r = a1.at(0);
    r=10;
    int &r1 = a1.at(1);
    r1=20;
    int &r2 = a1.at(2);
    r2=30;
    int &r3 = a1.at(3);
    r3=40;
    int &r4 = a1.at(4);
    r4=50;
    a1.output();
    return 0;
}
```




```
#include<iostream>
using namespace std;
class Array{
    int arr[10];
    int n;
    public:
        Array(int n1=0)
        {
            n=n1;
        }
        int& at(int ind)
        {
            return arr[ind];
        }
        int& operator[](int ind)
        {
            return arr[ind];
        }
        void output()
        {
            for(int i=0;i<n;i++)
                cout<<arr[i]<<" ";
            cout<<endl;
        }
};
int main(){
    int n;
    cout<<"Enter number of elements:";
    cin>>n;
    Array a1=n;
    for(int i=0;i<n;i++)
    {
        // int &r = a1.at(i);
        cout<<"Enter value of "<<i+1<<" element:";
        cin>>a1[i];
    }
    for(int i=0;i<n;i++)
    {
        cout<<a1[i]<<" ";
    }
```

```
    return 0;
}
```

Type Casting :-

int a = 7.5;

int double

primitive → old

int
float
char

int a = int(7.5)

Explicit

data
int a
float b

✓ object to primitive

data d_1(10, 7.5)

int x = int(d_1)

d_1: a [10], b [7.5]

x [10]

float y = float(d_1);

y [7.5]

primitive to object

int x = 10;

float y = 7.5;

✓ data d_1 = data (x); int

data d_1 = x

constructor

```
#include<iostream>
using namespace std;
class Data
{
    int a;
    float b;
    public:
        Data()
        {
        }
        explicit Data(int a1)
        {
            a=a1;
        }
        explicit Data(float b1)
        {
            b=b1;
        }
        explicit Data(int a1, float b1)
        {
            a=a1;
            b=b1;
        }
        void output()
        {
            cout<<a<<'\t'<<b<<endl;
        }
        explicit operator int()
        {
            return a;
        }
        explicit operator float()
        {
            return b;
        }
};
int main()
```

```
{
    Data d1=Data(10);
    d1.output();
    Data d2=Data(7.5f);
    d2.output();
    Data d3=Data(10,7.5);
    d3.output();
    int x = int(d1);
    cout<<x<<endl;
    float y = float(d2);
    cout<<y<<endl;
}
```

ostream

<< → Insertion

operator<< (int a) ≡ printf("%d", a);

data d1 (10, 7.5);

d1: a [10], b [7.5]

(float)

X cout<<d1;

~~d1 << cout~~

operator <<

~~<< (data a)~~

```
#include<iostream>
using namespace std;
class Data
{
    int a;
    float b;
    public:
        Data(int a1=0, float b1=0)
        {
            a=a1;
            b=b1;
        }
        friend istream& operator>>(istream& ,
Data&);
        friend ostream& operator<<(ostream&,
Data&);
};
ostream& operator<<(ostream &out, Data &obj)
{
    out<<obj.a<<'\t'<<obj.b<<endl;
    return cout;
}
istream& operator>>(istream& in, Data &obj)
{
    cout<<"Enter Int:";
    cin>>obj.a;
    cout<<"Enter Float:";
    cin>>obj.b;
    return cin;
}
int main()
{
```

```
        Data d1;
        Data d2;
        Data d3;
        cin>>d1>>d2>>d3;
        cout<<d1<<d2<<d3;
    }
```