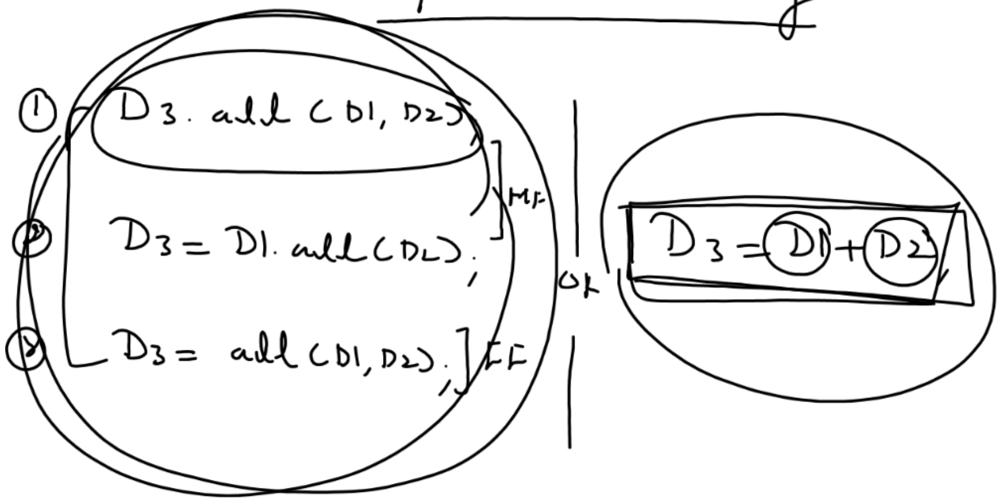
Operator Overloading



```
① D3.add(D1,D2)
② D3=D1.add(D2);   ] MF
③ D3= add(D1,D2);  ] FF
```

```
D3=D1+D2
```

Operator Overloading is a technique using which a programmer can redefine the existing operators of C++ lang in such a way that these operators can be used to work upon objects of user defined types much in the same way like they work upon variables of primitive data types.

Benefit ?
======
By overloading operators we can simplify the call to the function i.e. when we overload operators we actually create functions but these functions can be used as operators which can make the process of giving function calls simpler

Techniques Of Oveloading Operators
============================

As member fn

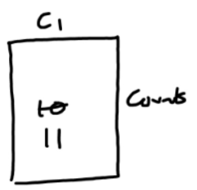
<return type> operator <op-symb> (arg);

As friend function

friend <ret type> operator <op-symb> (arg);

A ProgramTo Overload Unary Operatot ++ (Pre increment version) As Member Function Of The Class

```
#include <iostream>
using namespace std;
class Counter
{
   int count;
public:
   Counter(int c)
   {
      count=c;
   }
   Counter()
   {
      count=0;
   }
   void show()
   {
      cout<<"Count:"<<count<<endl;
                        10
   }
   void operator++();
};
void Counter::operator++()
{
   ++count;
}
```



Count++;
or
Count = Count+1.
or
Count += 1;

C1.operator++();

```
int main()
{
   Counter C1(10);
   C1.show();      ⟶ 10
   ++C1;
   C1.show();      ⟶ 11
   return 0;
}
```

```
#include <iostream>
using namespace std;
class Counter
{
   int count;
public:
   Counter(int c)
   {
      count=c;
   }
   Counter()
   {
      count=0;
   }
   void show()
   {
      cout<<"Count:"<<count<<endl;
   }
   void operator++();
};
void Counter::operator++()
{
   ++count;
}
```

Error!

```
int main()
{
   Counter C1(10);
   Counter C2;
   C1.show();
   C2.show();
   C2=++C1;

    C1.show();
    C2.show();
    return 0;
}
```

$\underbrace{C2 = C1.op++();}_{void}$

```
#include <iostream>
using namespace std;
class Counter
{
   int count;
public:
   Counter(int c)
   {
      count=c;
   }
   Counter()
   {
      count=0;
   }
   void show()
   {
      cout<<"Count:"<<count<<endl;
   }
   Counter operator++();
};
```

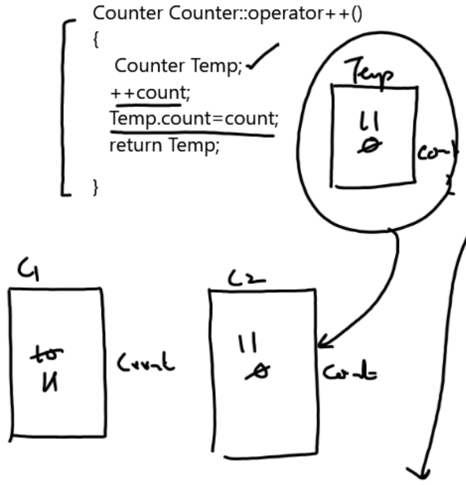
Improved version of previous code

```
Counter Counter::operator++()
{
   Counter Temp;
   ++count;
   Temp.count=count;
   return Temp;

}
```

```
int main()
{
 Counter C1(10);
 Counter C2;
  C1.show();    10
  C2.show();    0
  C2=++C1;

   C1.show();
   C2.show();
   return 0;
}
```



C2 = C1.op++();

x = ++y (x)

X c2 = ++c1 (c2)