Operator Overloading

+ << = ()
- >> []

```
Point operator-(Point &p1, Point &p2){
Point temp();
temp.x = p1.x-p2.x;
temp.y = p1.y-p2.y;
return temp;
}
```

```
ostream &operator>>(istream &in, Employee &e
out<<e1.id;
out<<e1.name;
out<<e1.salary;
return out;
}
```

```
void operator=(Point &ref)// this->p2
{
      *x = *ref.x;
      *y = *ref.y;
}
```

```
Point p1(2,3);
Point p2;
p2=p1;
```

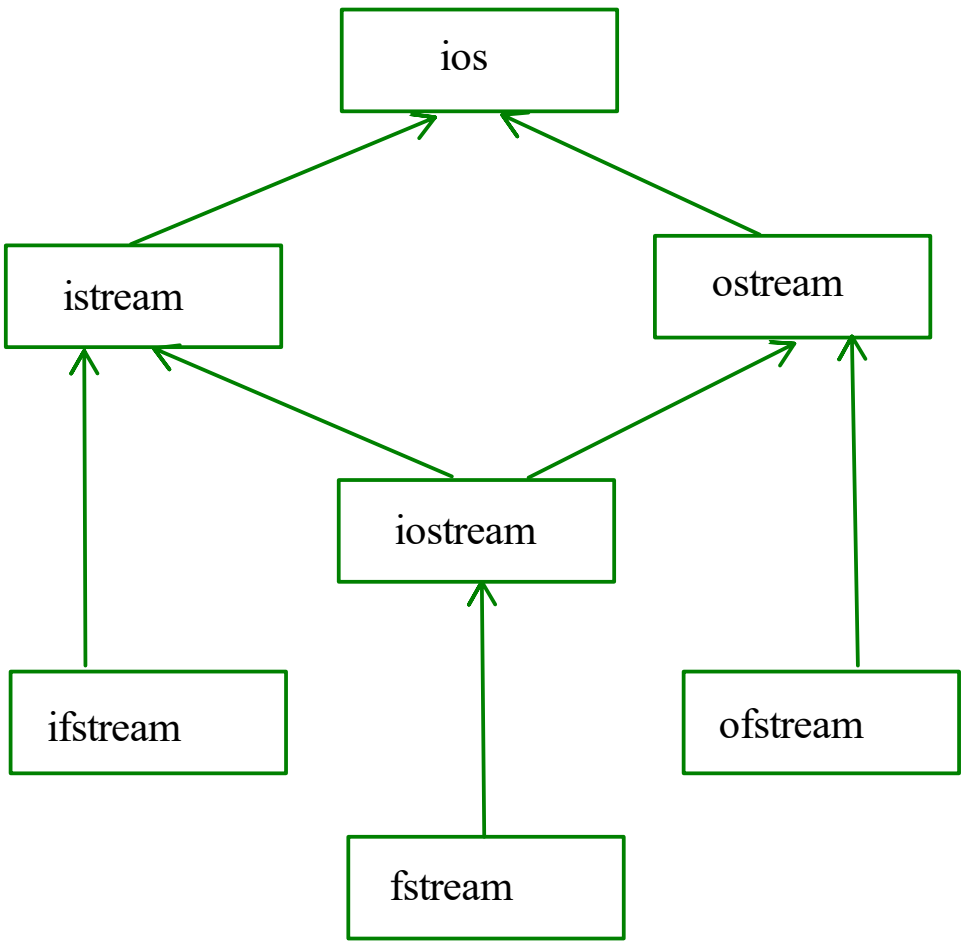
```
int arr[5];
arr[0];

Array a1;
a1[ ];
```

```
int& operator[](int index){
      return ptr[index];
}
```

```
cout<<a1[1];
int val = a1[1];
a1[1]=20;
```

Console I/O
File I/O



1,
anil,
10000

```
string line = "1,anil,10000";
stringstream data(line);
string id;
string name;
string salary;
getline(data , id , ',' );
getline(data, name , ',');
getline(data,salary, ',');
```

1 , anil , 10000

```
void f1(stringstream s1){
}

string name = "sunbeam"
stringstream s1(name);
f1(name);
```

```
int min = 120;


Time t1(120);
t1.display();
```

Employee -> employee.txt  vector<Employee *> empList;
Students -> students.txt vector<Student*> studentList;

menu driven
case1:
addemployee

Person.txt
E,1,anil,10000
S,1,mukesh,50

copy ctor

200 200

```
int main(){
vector<Employe> v1;
vector<Employee*> v2;
}

load(){
Employee e1;
v1.push_back(e1);

Employee *e = new Employee();
v2.push_back(e); //

}
```