OOPS

Creating Class

```cpp
#include<iostream>
using namespace std;

class Student{


    string Name;

    int Age;
    bool Gender;

    public:
    Student(){
        cout<<"Default Constructor called with no data members"<<endl;
    }

    Student(string name,int age,bool gender){
        Name=name;
        Age=age;
        Gender=gender;
    } //parameterised Constructor

    Student(Student &a){
        Name=a.Name;
        Age=a.Age;
        Gender=a.Gender;
        cout<<"Copy Constructor"<<endl;
    }
    void setName(string s){
        Name=s;
    }
    void setAge(int a){
        Age=a;
    }
    void setGender(bool g){
       Gender=g;
    }

    void display(){
        cout<<"\n";
        cout<<"Name Entered: ";
        cout<<Name<<endl;
        cout<<"Age Entered: ";
        cout<<Age<<endl;
        cout<<"Gender Entered: ";
        cout<<Gender<<endl;
        cout<<"\n";

    }
};



int main(){
//     Student arr[3];
//    for(int i=0;i<3;i++){
//     string s;
//     int age;
//     bool gender;
//     cout<<"Name: ";
//     cin>>s;
//     arr[i].setName(s);
//     cout<<"Age: ";
//     cin>>age;
//     arr[i].setAge(age);
//     cout<<"Gender: ";
//     cin>>gender;
//     arr[i].setGender(gender);
//    }
//     for(int j=0;j<3;j++){
//         arr[j].display();
//         cout<<"\n";
//     }
    Student a("Arpana",21,0);
    a.display();
    Student b("Gaurav",20,1);
    b.display();
    Student c=a;
    c.display();
    Student d(b);
    d.display();
    return 0;
}
```

Output

```
Name Entered: Arpana

Age Entered: 21

Gender Entered: 0
```

```
Name Entered: Gaurav

Age Entered: 20

Gender Entered: 1

Copy Constructor

Name Entered: Arpana

Age Entered: 21

Gender Entered: 0

Copy Constructor

Name Entered: Gaurav

Age Entered: 20

Gender Entered: 1
```

OPERATOR OVERLOADING

Adding two complex no’s.

```
#include<iostream>
using namespace std;

class Complex{
    private:
    int real,imag;
    public:
    Complex(int r,int i){
        real=r;
        imag=i;
    }
    Complex(){}
    Complex operator +(Complex obj){
        Complex tem;
        tem.real=real+obj.real;
        tem.imag=imag+obj.imag;
        return tem;
    }
    void display(){
        cout<<real<<" +i"<<imag;
    }
};
int main(){
    Complex c1(2,30),c2(1,6);
    Complex c3=c1+c2;
    c3.display();
    return 0;
}
```

Output: 3 +i36

OVERRIDING

Binds the address of the base class to point to child class at run time

```
#include<iostream>
using namespace std;

class Base{
    public:
     void print(){ // dynamically binds the address of child class to the base
class while runtime.
        cout<<"Print function of Base class"<<endl;
    }
    void display(){
        cout<<"Display function of Base class"<<endl;
    }
};
class Child:public Base{
    public:
   virtual void print(){
        cout<<"Print function of Child class"<<endl;
    }
    virtual void display(){
        cout<<"Display function of Child class"<<endl;
    }
};

int main(){
    Base *base;
    Child child;
    base=&child;
    base->display();
    base->print();
    return 0;
}
```

Output

Display function of Base class

Print function of Base class