

DR. B. R . AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY  
JALANDHAR, PANJAB



LAB-6  
OBJECT ORRIENTED PROGRAMMING

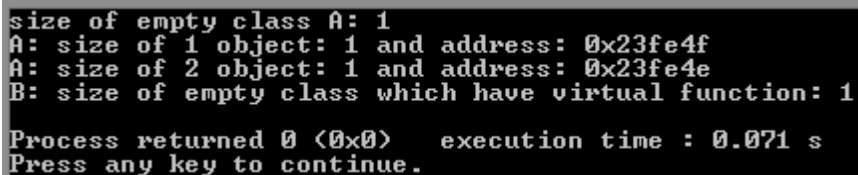
SUBMITTED BY:-

NAME :- Yash Sharma  
ROLL NO:-20103165  
BATCH:- CSE Section-2  
GROUP:- G-3

SUBMITTED TO:- Dr Nonita Sharma

**Prog 1: Demonstrate the concept of empty class. Show the size of empty class. Create two objects of the class. Show their addresses and size.**

```
#include<bits/stdc++.h>
using namespace std;
class A{
};
class B{
void display(){
}
};
int main(){
A a, b;
cout<<"size of empty class A: "<<sizeof(A)<<endl;
cout<<"A: size of 1 object: "<<sizeof(a)<<" and address: "<<&a<<endl;
cout<<"A: size of 2 object: "<<sizeof(b)<<" and address: "<<&b<<endl;
cout<<"B: size of empty class which have virtual function: "<<sizeof(B)<<endl;
return 0;
}
```



The screenshot shows the output of the C++ program. It displays the size of class A as 1, the size of object a as 1 with address 0x23fe4f, the size of object b as 1 with address 0x23fe4e, and the size of class B as 1. It also shows the process returned 0 and execution time of 0.071 s.

```
size of empty class A: 1
A: size of 1 object: 1 and address: 0x23fe4f
A: size of 2 object: 1 and address: 0x23fe4e
B: size of empty class which have virtual function: 1
Process returned 0 (0x0)   execution time : 0.071 s
Press any key to continue.
```

**Prog 2: Create a outer class contains another inner class and its definition, object of inner class and calls its function. [ Nested Class]**

```
#include<iostream>
using namespace std;
class A{
public:
int a;
class B{
public:
int b;
void get_data(){
printf("Enter element: ");
cin>>b;
}
void display(){
cout<<"data: "<<b;
}
};
};
int main(){
A :: B b;
b.get_data();
b.display();
return 0;
}
#include<bits/stdc++.h>
using namespace std;
class A{
```

```

public:
int a;
void get_data(){
cin>>a;
}
void display(){
cout<<"Data: "<<a;
}
};
class B{
public:
A a1;
};
int main(){
B b1;
cout<<"Enter element: ";
b1.a1.get_data();
b1.a1.display();
return 0;
}

```

```

Enter element: 12
data: 12
Process returned 0 (0x0)   execution time : 6.833 s
Press any key to continue.

```

### **Prog 3: Demonstrate the concept of object composition wherein one object contains object of another class.**

```

#include<bits/stdc++.h>
using namespace std;
class A{
public:
int a;
void get_data(){
cin>>a;
}
void display(){
cout<<"Data: "<<a;
}
};
class B{
public:
A a1;
};
int main(){
B b1;
cout<<"Enter element: ";
b1.a1.get_data();
b1.a1.display();
return 0;
}

```

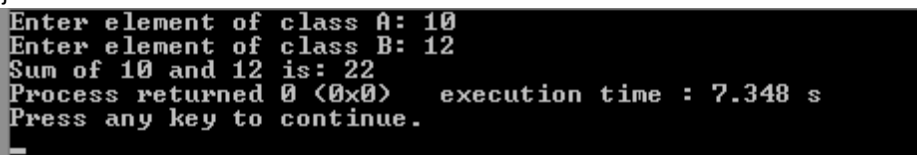
```

Enter element: 12
Data: 12
Process returned 0 (0x0)   execution time : 3.180 s
Press any key to continue.

```

## Prog 4: Create a friend function that adds the two integer members of different classes

```
#include<bits/stdc++.h>
using namespace std;
class A;
class B{
public:
int a;
void get_data(){
cin>>a;
}
friend int sum(B &, A &);
};
class A{
public:
int a;
void get_data(){
cin>>a;
}
friend int sum(B &, A &);
};
int sum(B &b,A &a ){
return b.a + a.a;
}
int main(int argc, char const *argv[]){
A a1;
B b1;
cout<<"Enter element of class A: ";
a1.get_data ();
cout<<"Enter element of class B: ";
b1.get_data();
cout<<"Sum of "<<a1.a <<" and "<<b1.a<<" is: "<<sum(b1, a1);
return 0;
}
```



```
Enter element of class A: 10
Enter element of class B: 12
Sum of 10 and 12 is: 22
Process returned 0 (0x0)   execution time : 7.348 s
Press any key to continue.
```

## Prog 5: Show the concept of friend class.

```
#include<bits/stdc++.h>
using namespace std;
class A{
public:
int b;
void display(){
cout<<"Data of class A that assign by class B: "<<b;
}
friend class B;
};
class B{
public:
void get_data(A &a){
printf("Enter element for assign class A: ");
cin>>a.b;
```

```

}
void display(A &a){
cout<<"Element of A print using friend class B: "<<a.b<<endl;
}
};
int main()
{
A a;
B b;
cout<<"Get and set of class A using friend class B: \n";
b.get_data(a);
b.display(a);
cout<<"Using class A: ";
a.display();
return 0;
}

```

```

Get and set of class A using friend class B:
Enter element for assign class A: 10
Element of A print using friend class B: 10
Using class A: Data of class A that assign by class B: 10
Process returned 0 (0x0)   execution time : 3.864 s
Press any key to continue.

```

## Prog 6: Demonstrate the concept of deep copy and shallow copy.

### Shallow copy:-

```

#include<bits/stdc++.h>
using namespace std;
class A{
private:
int q, r;
int *p;
public:
A(){
p = new int;
}
void getdata(int a, int b, int c){
*p = a;
q = b;
r = c;
}
void display(){
cout<<"P = "<<*p<<"\nQ = "<<q<<"\nR = "<<r<<endl;
}
void deepcopy(A &a){
*p = *a.p;
q = a.q;
r = a.r;
}
};
int main(){
A a;
cout<<"Shallow copy: \n";
a.getdata(1,2,3);
cout<<"a: \n";

```

```

a.display();
A b = a;
cout<<"b: \n";
b.display();
b.getdata(4,5,6);
cout<<"b: Using shallow copy: \n";
b.display();
cout<<"a: \n";
a.display();
return 0;
}

```

```

Shallow copy:
a:
P = 1
Q =2
R= 3
b:
P = 1
Q =2
R= 3
b: Using shallow copy:
P = 4
Q =5
R= 6
a:
P = 4
Q =2
R= 3

Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.

```

## Deep Copy:-

```

#include<iostream>
using namespace std;
class A{
private:
int q, r;
int *p;
public:
A(){
p = new int;
}
void getdata(int a, int b, int c){
*p = a;
q = b;
r = c;
}
void display(){
cout<<"P= "<<*p<<"\nQ ="<<q<<" \nR ="<<r<<endl;
}
void deepcopy(A &a){
p = new int;
*p = *(a.p);
q =a.q;
r = a.r;
}
};
int main(){
A a;

```

```
cout<<"Deep copy: \n";
a.getdata(1,2,3);
cout<<"a: \n";
a.display();
A b;
cout<<"b: using deep copy: \n";
b.deepcopy(a);
b.display();
cout<<"b: \n";
b.getdata(4,5,6);
b.display();
cout<<"a: \n";
a.display();
return 0;
}
```

```
Deep copy:
a:
P= 1
Q =2
R = 3
b: using deep copy:
P= 1
Q =2
R = 3
b:
P= 4
Q =5
R = 6
a:
P= 1
Q =2
R = 3

Process returned 0 (0x0)   execution time : 0.024 s
Press any key to continue.
```