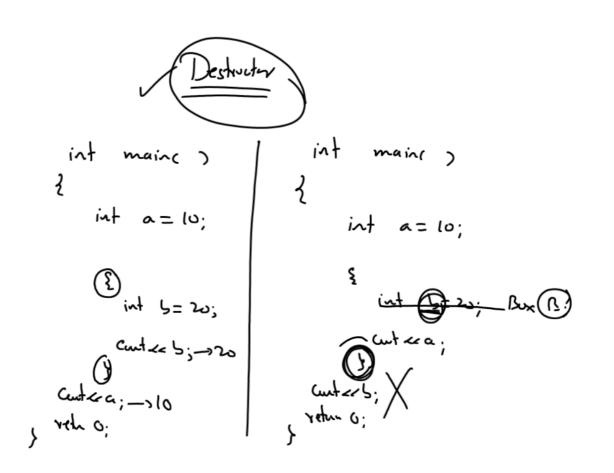
```
class Student
                                       void Student::show()
 int roll;
                                      cout<<roll<<","<<grade<<","<<per;
 char grade;
 float per;
public:
       Student(int,char,float);
       void get();
                                         int main()
       void show();
};
                                            Student S(10,'A',65.3);
Student::Student(int r,char g,float p)
                                          ³ Student P;
                                            P.get();
  roll=r;
                           enci.
                                           S.show();
  grade=g;
                                           P.show();
  per=p;
                                           return 0;
void Student::get()
cout<<"Enter roll,grade and per:";
cin>>roll>>grade>>per;
```

## Solution 1: Use Constructor Overloading

```
void Student::get()
class Student
                                                  {
 int roll;
                                                  cout < < "Enter roll, grade and per:";
 char grade;
                                                  cin>>roll>>grade>>per;
 float per;
public:
       Student(int,char,float);
                                                  void Student::show()
       void get();
                                                  cout<<roll<<","<<grade<<","<<per<<endl;
       void show();
       Student();
                                                  int main()
};
Student::Student()
                                                    Student S(10,'A',65.3);
{
                                                    Student P;
                                                    P.get();
                                                    S.show();
Student::Student(int r,char g,float p)
                                                    P.show();
                                                    return 0;
  roll=r:
                                                  }
  grade=g;
  per=p;
}
```

## **Solution 2: Use Default Parameterized Constructor**

```
class Student
                                                                  void Student::show()
  int roll;
  char grade;
                                                                  cout<<roll<<","<<grade<<","<<per
  float per;
public:
                                                                  int main()
       Student(int=0,char=' ',float=0.0);
                                                                    Student S(10,'A',65.3);
       void get();
                                                                    Student P;
       void show();
                                                                    P.get();
                                                                    S.show();
};
                                                                    P.show();
                                                                    return 0;
Student::Student(int r,char g,float p)
                                                                  }
  roll=r;
  grade=g;
  per=p;
void Student::get()
cout < < "Enter roll, grade and per:";
cin>>roll>>grade>>per;
}
```



## Destructor

## =======

- 1. In C++ just like we have constructor, similarly we also have the concept of Destructor .
- 2. A destructor is another special member function of the class having same name as that of the class but prefixed with the symbol of Tiled(~).
- 3. This means that if the class name is Box, the name of the constructor will be

Box() and the name of the destructor will be ~Box().

4. Now, whenever the C++ compiler will decide to destroy any object of the class then just before removing it from memory it will automatically call the destructor funtion present in the class.

- 5. So we can say that in C++ every object in its entire life time always automatically calls at least two member functions called
- 1. Constructor and
- 2. Destructor. The constructor function is called immediately after the object gets created and the destructor funtion gets called just before the object is to be destroy.
- 6. Moreover both these functions are implicitly called by the C++ compiler.
- 7. Also if in the class we don't declare any destructor ourselves then the C++ compiler automatically inserts a destructor in our class called as default destructor having an empty body.

```
class Student
                                          void Student::show()
 int roll;
                                          cout<<roll<<","<<grade<<","<<per<<endl;
 char grade;
 float per;
public:
                                          Student::~Student()
       Student();
       void show();
                                            cout < < "Destructor called..." < < endl;
       ~Student();
};
                                          int main()
Student::Student()
                                            Student S;
  cout<<"Constructor called..."<<endl;
                                            Student P;
  cout<<"Enter roll,grade and per:";
                                            S.show();
  cin>>roll>>grade>>per;
                                           P.show();
                                            return 0;
                                          }
```