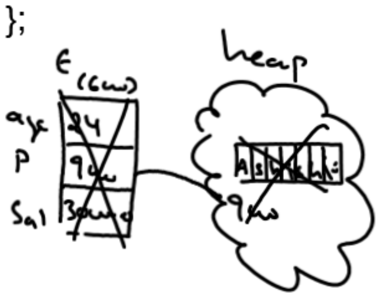
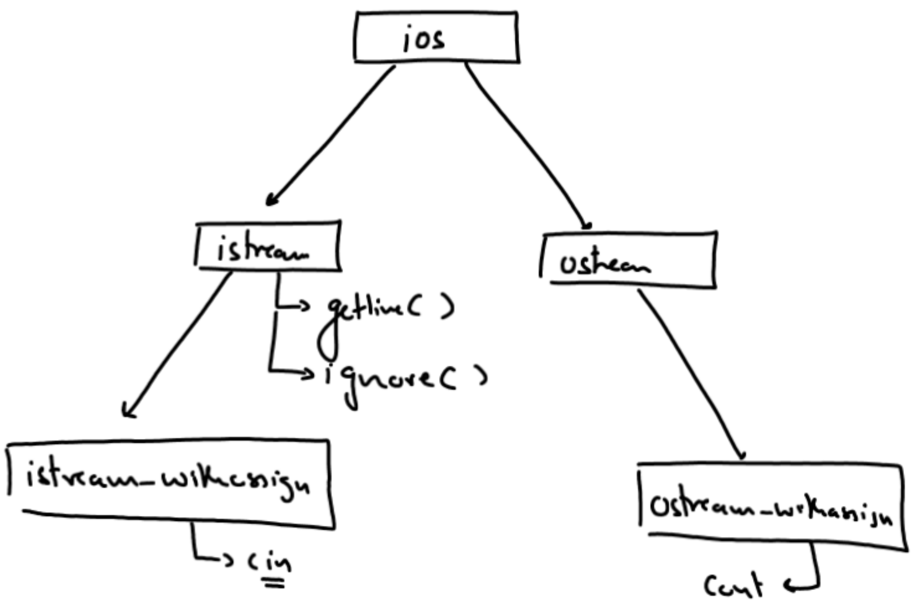
```
#include <iostream.h>
#include <conio.h>
#include <alloc.h>
#------ <string.h>
class Emp
{
   int age;
   char *p;
   float sal;
public:
         Emp();
         void show();
         ~Emp();
};
```



```
 Emp::Emp()
 {

   cout<<"Enter your age and sal:"
✓  cin>>age>>sal;
   char temp[20];
   cout<<"Enter your name:";
   cin.ignore();
   cin.getline(temp,20);
   int x=strlen(temp);
✓  p=(char*)malloc((x+1)*sizeof(char));
   strcpy(p,temp);
}
void Emp::show()
{
  cout<<age<<","<<p<<","<<sal;
}
Emp::~Emp()
{
  free(p);
}
```

```
int main()
{
  Emp E;

  E.show();

  getch();

 return 0;
}
```



Constructor
=========
1. They are special member function of a class having the same name as that of the class.

2. They are implictly called by the C++ compiler as soon as the object of a class gets created.

3. They are automatically call in the order in which objects are created

4. They can be parametrized.

5. Constructor can be overlaoded so a class can have multiple constructors.

Destructor
=========
1. A destructor is also a special member function of a class having the same name as that of the class but prefixed with tilde (~).

2. They also are implicitly called by the C+ compiler but just before the object is to be destroyed.

3. A desctructor is also automatically called but in reverse order of creation of the object.

4. They can't accept any parameter.

5. We can't overload a desctructor so a class can have only one destructor.

Constructor	Destructor
=========	=========
6. By default the C++ compiler provides two constructors if we don't create our own constructor and they are default constructor and default copy constructor.	6. By default the C++ compiler provides only one desctructor if we don't create our own destructor and it is called as default destructor.
7. Constructor can't be declared as static.	7. A destructor also can't be declared as static
8. Constructor can't be declared as const.	8. A destructor also can't be declared as const.
9. Constructor is not inherited.	9. A desctructor also is not inherited.
10. A constructor can't be declared as virtual.	10. We can declare destructor as virtual.