Virtual Functions

- Virtual function also plays an important role in Object Oriented Programming.
- Virtual function uses the concept of late binding.
- In this, pointer plays a main role in it.
- A virtual function is a member function which is defined in base class and again redefined in the derived class. i.e. virtual function are defined both in the base and derived class with same name.
- function are declared with a "virtual" keyword in base class.

Syntax:-

```
class classname
{
public:
virtual returntype function_name(arguments
...
};
```

Rules of Virtual function

Virtual functions must be members of some class.

Virtual functions cannot be static members.

They are accessed through object pointers.

They can be friend of another class.

A virtual function must be defined in the base class, even though it is not used.

A virtual function re-defined in derived class without using virtual keyword.

Program for virtual function

```
#include<iostream.h>
#include<conio.h>
 class base
  {
  public:
  virtual void print()
   {
```

```
    cout<< "base function";
  }
};
  class derived : public base
  {
  public:
  void print()
  {
  cout<< "derived function";
  }
};
  void main()
  {
  clrscr();
  derived d;
  base* b = &d;
  b-> print();
  getch();
  }
```

Output :- derived function.

Pure Virtual Function.

A pure virtual is a virtual function with no body [ε/]. So the body of the virtual function in the base class can be removed.

A Pure virtual function is a type of function, which has only a function declaration

Another name of pure virtual function is empty virtual function.

Here base class becomes an abstract class.

It is declared by assigning 0 in the declaration.

Syntax :- virtual returntype functionname() = 0;

For example :- virtual void display() = 0;