Virtual Functions:

A virtual function is a member function of the base class, that is overridden in derived class. The classes that have virtual functions are called polymorphic classes.

The compiler binds virtual function at runtime, hence called runtime polymorphism. Use of virtual function allows the program to decide at runtime which function is to be called based on the type of the object pointed by the pointer.

In C++, the member function of a class is selected at runtime using virtual function. The function in the base class is overridden by the function with the same name of the derived class.

C++ virtual function : Syntax

```
class class_name
{
        public:
        virtual return func_name( args.. )
        {
                //function definition
        }
 }
```

Why do we need virtual functions?
Virtual functions are needed for many reasons, among them to eliminate ambiguity is one.

```
#include <iostream>
using namespace std;
class Animal
{
        public:
                void my_features()
                {
                        cout << "I am an animal.";
                }
};
```

```
class Mammal : public Animal
{
        public: void my_features()
        {
                cout << "\nI am a mammal.";
        }
};
class Reptile : public Animal
{
        public: void my_features()
                {
                        cout << "\nI am a reptile.";
                }
};
```

```
int main()
{
        Animal *obj1 = new Animal;
        Mammal *obj2 = new Mammal;
        Reptile *obj3 = new Reptile;
        obj1->my_features();
        obj2->my_features();
        obj3->my_features();
        return 0;
}
```

Output:

I am an animal.
I am a mammal.
I am a reptile.

Pure Virtual Functions and Abstract Classes in C++

Sometimes implementation of all function cannot be provided in a base class because we don't know the implementation. Such a class is called abstract class.
For example, let Shape be a base class. We cannot provide implementation of function draw() in Shape, but we know every derived class must have implementation of draw().
Similarly an Animal class doesn't have implementation of move() (assuming that all animals move), but all animals must know how to move. We cannot create objects of abstract classes.
A pure virtual function (or abstract function) in C++ is a virtual function for which we don't have implementation, we only declare it.
A pure virtual function is declared by assigning 0 in declaration.
See the following example.

```
// An abstract class
class Test
{
    // Data members of class
public:
    // Pure Virtual Function
    virtual void show() = 0;

   /* Other members */
};
```

A complete example:

A pure virtual function is implemented by classes which are derived from a Abstract class. Following is a simple example to demonstrate the same.

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

class Base
{
   int x;
public:
    virtual void fun() = 0;
    int getX() { return x; }
};
```

```
// This class inherits from Base and implements fun()
class Derived: public Base
{
    int y;
public:
    void fun() { cout << "fun() called"; }
};

int main(void)
{
    Derived d;
    d.fun();
    return 0;
}
```

Output:

```
fun() called
```

Some Interesting Facts:

1) A class is abstract if it has at least one pure virtual function.
2) We can have pointers and references of abstract class type.
3) If we do not override the pure virtual function in derived class, then derived class also becomes abstract class.
4) An abstract class can have constructors.

Restrictions for using abstract classes:

Abstract classes cannot be used for:

1. Variables or member data
2. Argument Types
3. Functions return types
4. Types of explicit conversions