C++ abstract class and pure virtual function

## **C++ abstract class and pure virtual function : Introduction**

**C++ abstract class** is a class designed for the role of the base class having at least one pure virtual function. A class without a **pure virtual function** cannot be termed as an abstract base class in C++. Abstract classes are used as a framework upon which new subclasses are derived.

Similarly, a virtual function without definition or hull definition is called a pure virtual function.

Example:

virtual return\_type func\_name() = 0;

A class having a pure virtual function cannot be instantiated i.e the object of abstract classes cannot be created. However, a pointer to the abstract base class or abstract class can be created.  They only serve as the foundation to derive subclasses.

Another important thing about pure virtual function and abstract class is that the pure virtual function must be overridden in derived class. Else the pure inherited pure virtual function remains same in all derived classes. Hence pure abstract classes and pure virtual function allow a programmer to build the implementation in stages.

## **C++ abstract class : syntax and structure**

//declaring abstract base class

class base\_class

{

virtual return\_type func\_name() = 0; //pure virtual function

}

As illustrated above, an abstract class must have a virtual function with null definition i.e **pure virtual function**.

### **Example: Simple illustration of abstract class and pure virtual function in C++**

//pure\_virtual\_func.cpp

#include <iostream>

using namespace std;

class base\_class

{

public:

virtual void display() = 0;

};

class derived\_class : public base\_class

{

public:

void display()

{

cout<<"This is simple illustration of abstract class and pure virtual function";

}

};

int main()

{

derived\_class obj;

obj.display();

return 0;

}

**Output**

This is simple illustration of abstract class and pure virtual function