**LAB 10:TO ILLUSTRATE THE CONCEPT OF PURE VIRTUAL FUNCTIONS IN C++.WAP OF YOUR CHOICE.**

**OBJECTIVE:**

1.To understand the concept of abstract classes in C++.

2.To learn about pure virtual functions and their significance.

3.To create an abstract class with one or more pure virtual functions.

4.To implement derived classes that inherit from the abstract class and provide concrete implementations for the pure virtual functions.

**THEORY:** In the word of object-oriented programming (OOP), C++ offers powerful tools to design and implement complex software systems. One fundamental feature that C++ provides for managing class hierarchies and enforcing a specific structure in derived classes is the concept of abstract classes and pure virtual functions. This lab aims to provide a comprehensive understanding of abstract classes and pure virtual functions in C++ by delving into their definitions, applications, and practical usage through suitable examples. Abstract Classes: In the world of C++, an abstract class serves as a blueprint for other classes. It provides a common interface for its derived classes, while preventing instances of the abstract class from being created. Abstract classes often contain a mix of concrete (implemented) methods and pure virtual functions. Pure Virtual Functions: A pure virtual function is a special function that is declared in an abstract class but has no implementation. It serves as a contract that mandates all derived classes to provide their own implementations for this function.

**Key Concepts:**

**Abstract Class:** An abstract class is a class that cannot be instantiated, and it typically contains one or more pure virtual functions. It acts as a template for derived classes, defining a common interface.

**Pure Virtual Function:** A pure virtual function is a virtual function declared in an abstract class but has no implementation. It is marked with the virtual keyword and the = 0 assignment, making it mandatory for derived classes to provide an implementation.

**Polymorphism:** Through the use of abstract classes and pure virtual functions, we achieve polymorphism, allowing objects of different derived classes to be treated uniformly through pointers or references to the abstract base class.

**//SOURCE CODE:**

#include <iostream>

using namespace std;

// Abstract class with a pure virtual function

class Colour {

public:

virtual void print() = 0; // Pure virtual function

};

// Concrete class derived from the abstract class

class Brown : public Colour{

public:

void print() override {

cout << "My favourite colour is brown." << endl;

}

};

int main() {

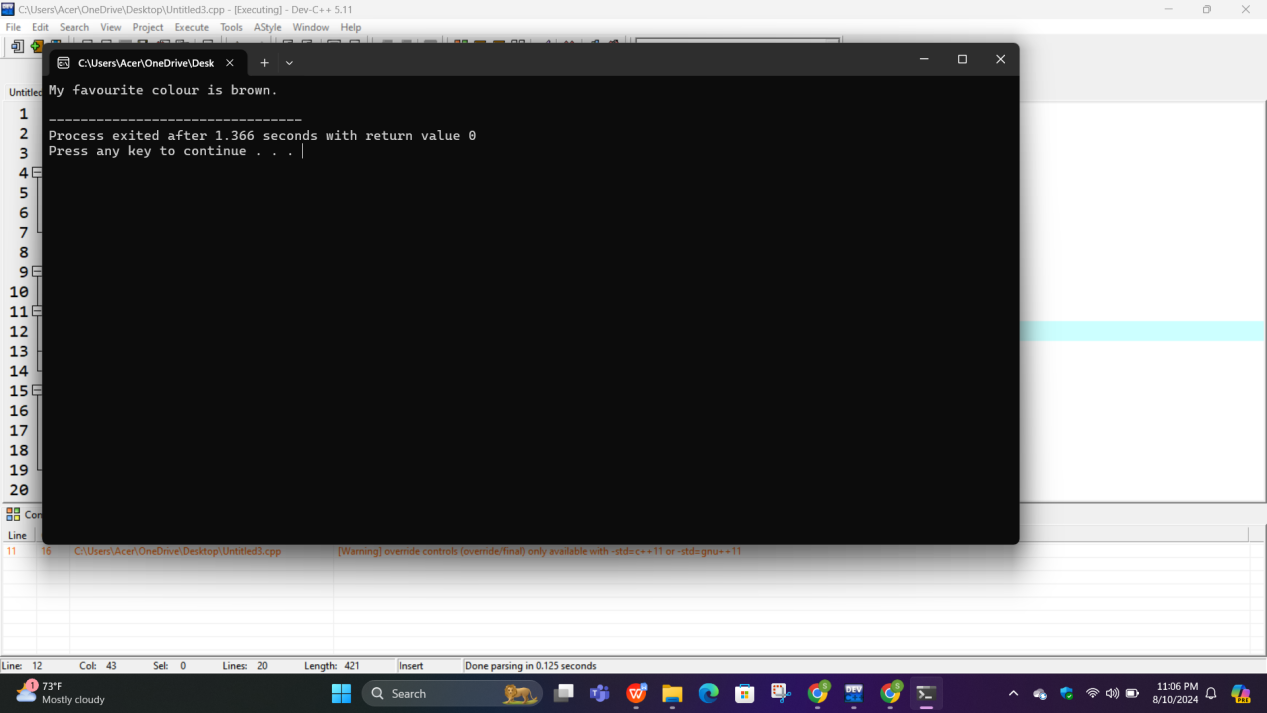
Brown b;

b.print();

return 0;

}

**OUTPUT:**

****

**CONCLUSION:**

In this C++ program, we've effectively demonstrated the concepts of abstract classes and pure virtual functions. Abstract classes and pure virtual functions prove invaluable for enforcing consistent interfaces and facilitating code reuse and extensibility in complex software systems