Virtual Function in C++

What is a Virtual Function in C++?

A **virtual function** in C++ is a special type of function that allows a derived class to override a function defined in its base class. When a function is declared as virtual, C++ decides which function to call at runtime based on the actual object type, not the type of the pointer or reference that is used to call the function. This behavior is known as **runtime polymorphism**.

Why Do We Need Virtual Functions?

Virtual functions are useful when you have a base class and multiple derived classes that implement the same function in different ways. By using virtual functions, you can work with pointers or references to the base class while still calling the correct function from the derived class, based on the actual object type.

How Do Virtual Functions Work?

1. **Base Class Declaration**: A function in the base class is marked as `virtual` if it is meant to be overridden by derived classes.
2. **Derived Class Override**: Each derived class can provide its own version of the virtual function.
3. **Runtime Decision**: When a virtual function is called through a base class pointer or reference, C++ determines which function to invoke at runtime, based on the actual type of the object.

Real-Life Examples of Virtual Functions

1. **GUI Frameworks** (e.g., handling different types of widgets like buttons and sliders)
2. **File Handling Systems** (e.g., reading and writing different file formats like text, binary, and images)
3. **Game Development** (e.g., implementing character actions such as attacking or defending)
4. **Vehicle Management Systems** (e.g., managing various types of vehicles with specific behaviors)
5. **Document Processing Systems** (e.g., processing different document types like Word, PDF, and Spreadsheet)
6. **Shape Drawing Applications** (e.g., drawing various shapes like circles, rectangles, and triangles)
7. **Media Players** (e.g., playing different media formats like audio, video, and images)
8. **Employee Payroll Systems** (e.g., calculating pay for different employee types like salaried, hourly, and commission-based)

9. **Robotics Control Systems** (e.g., controlling different types of robots with specialized movements)

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

class Animal {
public:
    virtual void sound() {  // Virtual function in the base class
        cout << "Animal makes a sound" << endl;
    }
};

class Dog : public Animal {
public:
    void sound() override {  // Overriding the virtual function
        cout << "Dog barks" << endl;
    }
};

class Cat : public Animal {
public:
    void sound() override {  // Overriding the virtual function
        cout << "Cat meows" << endl;
    }
};

int main() {
    Animal* animalPtr;

    Dog dog;
    Cat cat;

    animalPtr = &dog;
    animalPtr->sound();  // Output: Dog barks

    animalPtr = &cat;
    animalPtr->sound();  // Output: Cat meows

    return 0;
}
```

Explanation:

- **Virtual Function**: `sound()` is declared as a virtual function in the `Animal` base class.
- **Derived Classes**: The `Dog` and `Cat` classes override the `sound()` function to provide their own specific behavior.
- **Base Class Pointer**: When `animalPtr` points to a `Dog` object, calling `animalPtr->sound()` invokes the `Dog`'s version of `sound()`. Similarly, when it points to a `Cat` object, it invokes the `Cat`'s version.

Conclusion

Virtual functions are a powerful feature in C++ that enable runtime polymorphism. They allow you to write flexible and maintainable code, making it easier to work with different types of derived objects using a common base class interface.

Thanks for visiting!
Author:Pratham Jaiswal