Object-Oriented Programming Using C++

Polymorphism and Virtual Function

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Polymorphism

- polymorphism occurs when there is a hierarchy of classes and they are related by inheritance
- Polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function

Example: Polymorphism - Wrong Attempt

Polymorphism in the Shape class

```
class Shape {
   protected:
      int width, height;
   public:
      Shape( int a = 0, int b = 0){
         width = a;
         height = b;
      }
      int area() {
         cout << "Parent class area :" <<endl;
         return 0;
      }
};
class Rectangle: public Shape {
   public:
      Rectangle( int a = 0, int b = 0):Shape(a, b) { }
      int area () {
         cout << "Rectangle class area :" <<endl;
         return (width * height);
      }
};

class Triangle: public Shape {
   public:
      Triangle( int a = 0, int b = 0):Shape(a, b) { }
      int area () {
         cout << "Triangle class area :" <<endl;
         return (width * height / 2);
      }
};
```

Example: Polymorphism - Wrong Attempt

Polymorphism in the Shape class

```
// Main function for the program
int main() {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);

   // store the address of Rectangle
   shape = &rec;

   // call rectangle area.
   shape->area();

   // store the address of Triangle
   shape = &tri;

   // call triangle area.
   shape->area();

   return 0;
}
```

Example: Polymorphism - Wrong Attempt

Output

```
Parent class area :
Parent class area :
```

Static Linkage

- The call of the function area() is being set once by the compiler as the version defined in the base class
- This is called static resolution of the function call, or static linkage - the function call is fixed before the program is executed
- This is also sometimes called early binding because the area() function is set during the compilation of the program

Virtual Function

- A virtual function is a function in a base class that is declared using the keyword virtual
- Defining in a base class a virtual function, with another version in a derived class, signals to the compiler that we don't want static linkage for this function.
- The selection of the function to be called at any given point in the program to be based on the kind of object for which it is called - dynamic linkage or late binding

Example: Polymorphism using Virtual Function

Virtual function in the Shape class

```
class Shape {
   protected:
      int width, height;
   public:
      Shape( int a = 0, int b = 0) {
         width = a;
         height = b;
      }
      virtual int area() {
         cout << "Parent class area :" <<endl;
         return 0;
      }
};
class Rectangle: public Shape {
   public:
      Rectangle( int a = 0, int b = 0):Shape(a, b) { }
      int area () {
         cout << "Rectangle class area :" <<endl;
         return (width * height);
      }
};
class Triangle: public Shape {
   public:
      Triangle( int a = 0, int b = 0):Shape(a, b) { }
      int area () {
         cout << "Triangle class area :" <<endl;
         return (width * height / 2);
      }
};
```

Example: Polymorphism using Virtual Function

Virtual function in the Shape class

```
// Main function for the program
int main() {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);

   // store the address of Rectangle
   shape = &rec;

   // call rectangle area.
   shape->area();

   // store the address of Triangle
   shape = &tri;

   // call triangle area.
   shape->area();

   return 0;
}
```

Example: Polymorphism using Virtual Function

Output

```
Rectangle class area
Triangle class area
```

Pure Virtual Function

- You may want to include a virtual function in a base class so that it may be redefined in a derived class to suit the objects of that class
- there is no meaningful definition you could give for the function in the base class.

Example: Pure Virtual Function

Pure Virtual function in the Shape class

```
class Shape {
   protected:
      int width, height;

   public:
      Shape(int a = 0, int b = 0) {
         width = a;
         height = b;
      }

      // pure virtual function
      virtual int area() = 0;
};
```

Object-Oriented Programming Using C++

Polymorphism and Virtual Function

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur