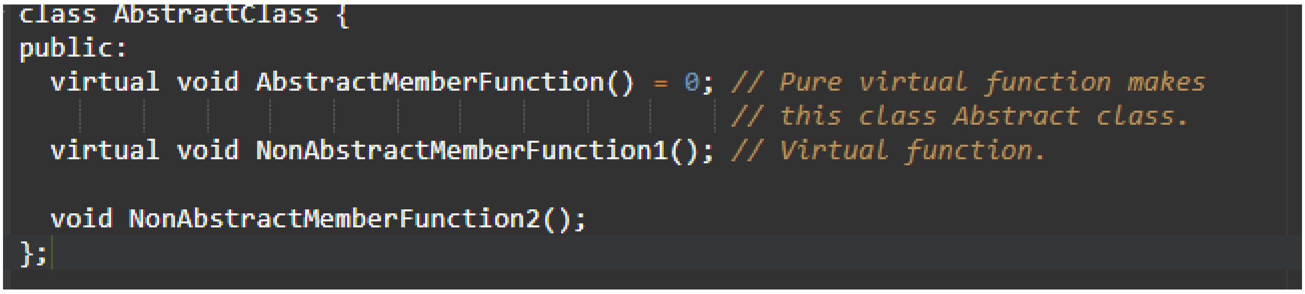
What is abstract class?

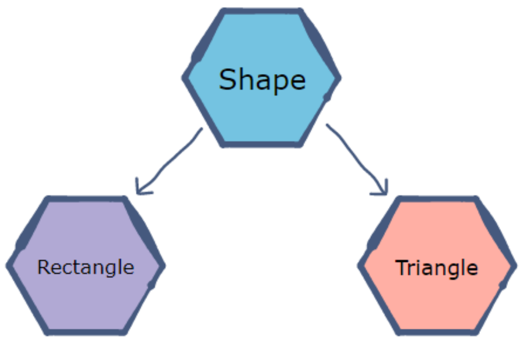
abstract class in C++ is a class that has at least *one* pure virtual function (i.e., a function that has no definition). The classes inheriting the abstract class *must* provide a definition for the pure virtual function; otherwise, the subclass would become an abstract class itself.

We **cannot create objects of an abstract class**. However, we can derive classes from them, and use their data members and member functions (except pure virtual functions).



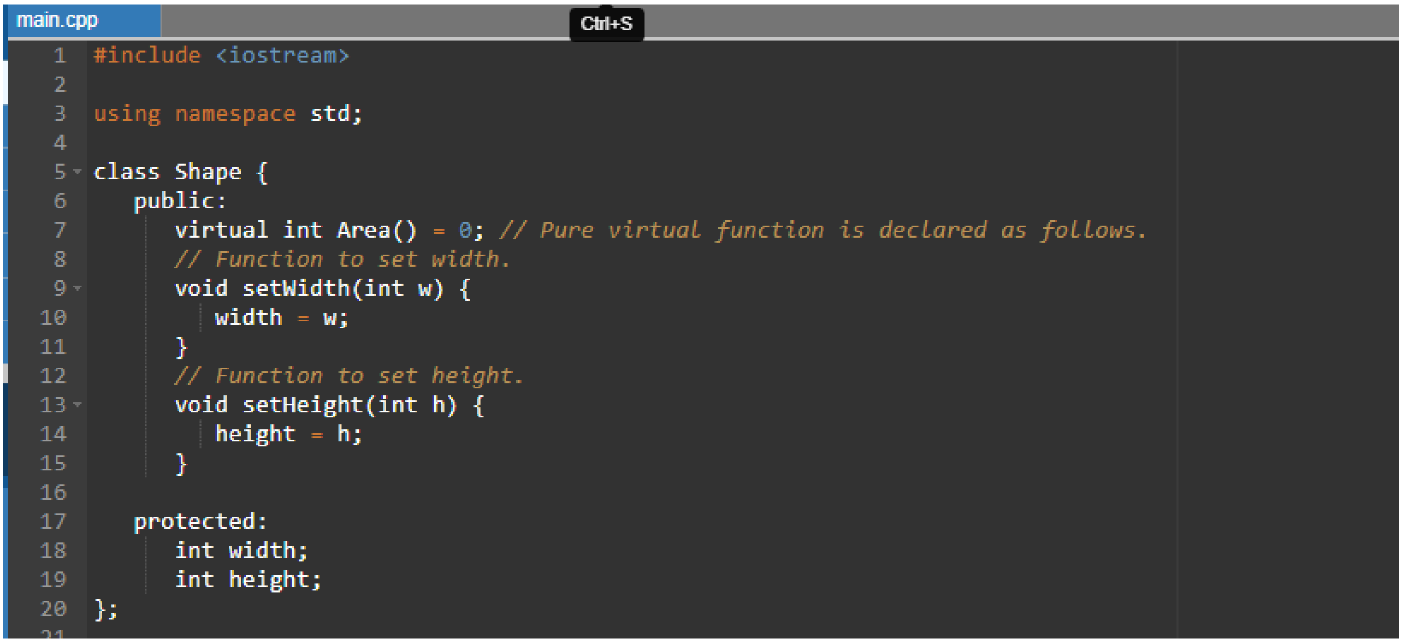
```
class AbstractClass {
public:
  virtual void AbstractMemberFunction() = 0; // Pure virtual function makes
                                             // this class Abstract class.
  virtual void NonAbstractMemberFunction1(); // Virtual function.

  void NonAbstractMemberFunction2();
};
```

Consider an example of a `Shape` base class with sub-classes (`Triangle` and `Rectangle`) that inherit the `Shape` class.

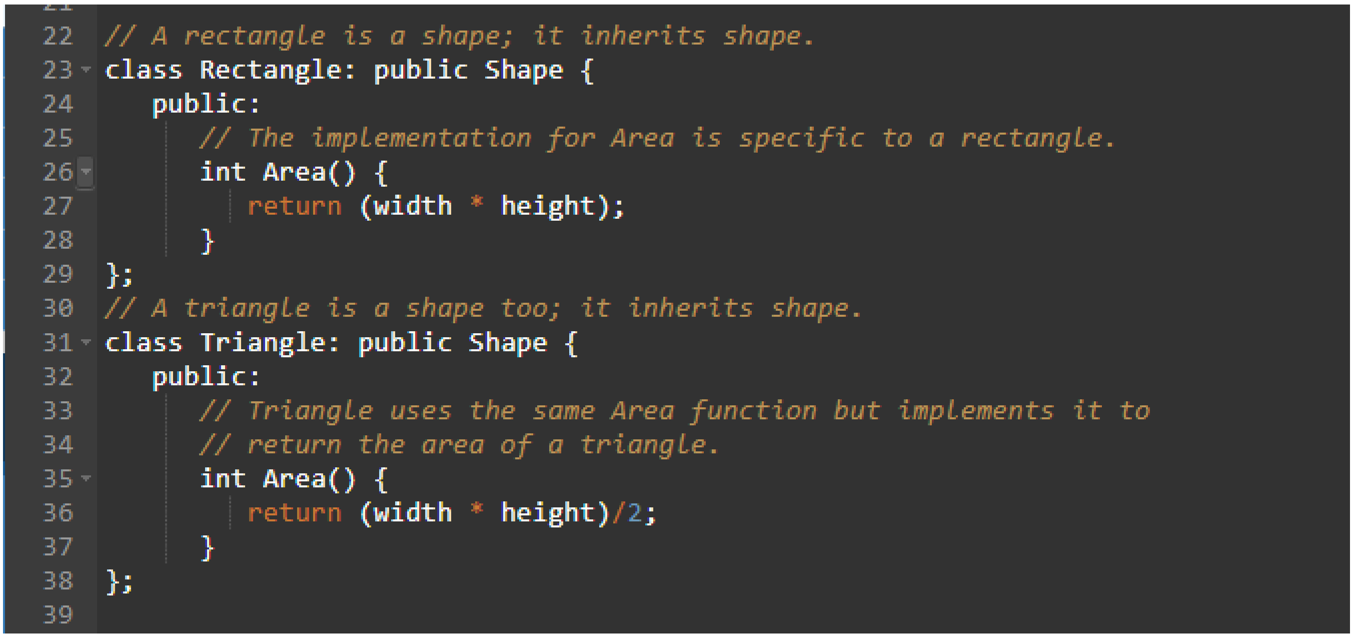


Now, suppose we need a function to return the area of a shape. The function will be declared in the Shape class; however, it cannot be defined there as the formula for the area is different for each shape. A non-specific shape does not have an area, but rectangles and triangles do. Therefore, the pure virtual function for calculating area will be implemented differently by each sub-class.

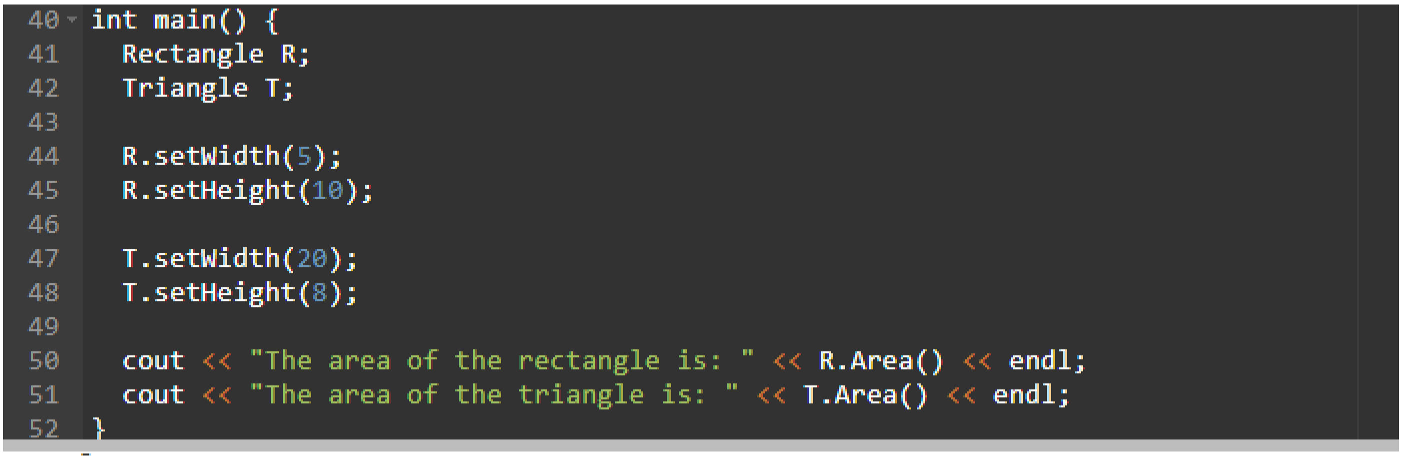
The following code snippet implements the abstract Shape class along with its sub-classes:



```cpp
#include <iostream>

using namespace std;

class Shape {
   public:
      virtual int Area() = 0; // Pure virtual function is declared as follows.
      // Function to set width.
      void setWidth(int w) {
         width = w;
      }
      // Function to set height.
      void setHeight(int h) {
         height = h;
      }

   protected:
      int width;
      int height;
};
```



```cpp
// A rectangle is a shape; it inherits shape.
class Rectangle: public Shape {
   public:
      // The implementation for Area is specific to a rectangle.
      int Area() {
         return (width * height);
      }
};
// A triangle is a shape too; it inherits shape.
class Triangle: public Shape {
   public:
      // Triangle uses the same Area function but implements it to
      // return the area of a triangle.
      int Area() {
         return (width * height)/2;
      }
};

```



```cpp
int main() {
  Rectangle R;
  Triangle T;

  R.setWidth(5);
  R.setHeight(10);

  T.setWidth(20);
  T.setHeight(8);

  cout << "The area of the rectangle is: " << R.Area() << endl;
  cout << "The area of the triangle is: " << T.Area() << endl;
}
```

Note: The return type of the virtual function must be consistent throughout all of its implementing classes.

Why can't we create Object of an Abstract Class?

When we create a pure virtual function in Abstract class, we reserve a slot for a function in the **VTABLE**(, but doesn't put any address in that slot. Hence the VTABLE will be incomplete.

As the VTABLE for Abstract class is incomplete, hence the compiler will not let the creation of object for such class and will display an errror message whenever you try to do so.

Rules of Abstract Class

1) As we have seen that any class that has a **pure virtual function** is an abstract class.
2) We cannot create the instance of abstract class. For example: If I have written this line `Shape obj;` in the above program, it would have caused compilation error.
3) We can create pointer and reference of base abstract class points to the instance of child class. For example, this is valid:

```
Shape *obj = new Dog();
obj->sound();
```

4) Abstract class can have constructors.
5) If the derived class does not implement the pure virtual function of parent class then the derived class becomes abstract.