https://youtu.be/2krvZ3-INUk

Virtual Functions

Virtual functions are a polymorphic feature. These functions are declared (and possibly defined) in a base class, and can be overridden by derived classes.

This approach declares an **interface** at the base level, but delegates the implementation of the interface to the derived classes.

In this exercise, `class Shape` is the base class. Geometrical shapes possess both an area and a perimeter. `Area()` and `Perimeter()` should be virtual functions of the base class interface. Append `= 0` to each of these functions in order to declare them to be "pure" virtual functions.

A **pure virtual function** is a **virtual function** that the base class **declares** but does not **define**.

A pure virtual function has the side effect of making its class **abstract**. This means that the class cannot be instantiated. Instead, only classes that derive from the abstract class and override the pure virtual function can be instantiated.

```
class Shape {
    public:
        Shape() {}
        virtual double Area() const = 0;
        virtual double Perimeter() const = 0;
};
```

Virtual functions can be defined by derived classes, but this is not required. However, if we mark the virtual function with `= 0` in the base class, then we are declaring the function to be a pure virtual function. This means that the base class does not define this function. A derived class must define this function, or else the derived class will be abstract.

Instructions

1. Create base class called `Shape`.
2. Define pure virtual functions (`= 0`) for the base class.
3. Write the derived classes.
    - Inherit from `class Shape`.
    - Override the pure virtual functions from the base class.
4. Test in `main()`