- Solution

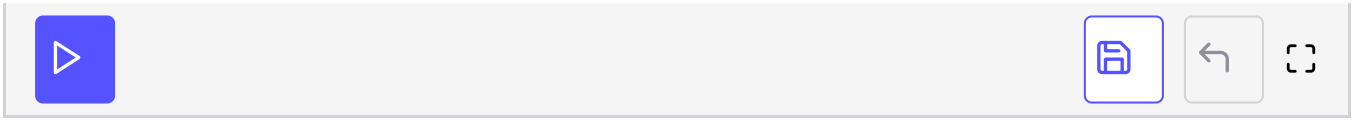
Let's look at the solution of the problem discussed in the previous lesson.

WE'LL COVER THE FOLLOWING

Solution

```
#include <iostream>

class Sort{
public:
  virtual void processData() final {
    readData();
    sortData();
    writeData();
  }
private:
  virtual void readData(){}
  virtual void sortData()= 0;
  virtual void writeData(){}
};


class QuickSort: public Sort{
private:
  void readData() override {
    std::cout << "readData" << std::endl;
  }
  void sortData() override {
    std::cout <<  "sortData" << std::endl;
  }
  void writeData() override {
    std::cout << "writeData" << std::endl;
  }
};

class BubbleSort: public Sort{
private:

  void sortData() override {
    std::cout <<  "sortData" << std::endl;
  }
```

```
};


int main(){

  std::cout << std::endl;

  Sort* sort = new QuickSort;
  sort->processData();

  std::cout << std::endl;

}
```



Explanation

- We have implemented three classes named `Sort`, `QuickSort`, and `BubbleSort`.
- We have created three private `virtual` methods and a public `virtual final` method `processData` in the `Sort` class which calls the three private methods.
- The method `SortData` is pure virtual and defined in class `Sort`.
- The classes `QuickSort` and `BubbleSort` publically inherit from the `Sort` class.
- We have overridden the methods of the `Sort` class in `QuickSort`.
- We have overridden the method `sortData` of `Sort` class in `BubbleSort`.
- By using a pointer to the Base class, we can access the overridden methods of the derived class.

In the next lesson, we'll discuss multiple inheritance in detail.