Задача 1.

Сделаем 5 наследников от класса Animal — по одному на каждое животное. Перегрузим у каждого из них виртуальную функцию say(). Напечатаем в функции то, какой звук должно издавать конкретное животное.

Далее необходимо лишь добавить каждого животного в массив.

```
#include <iostream>
#include <vector>
class Animal {
public:
    virtual void say() {}
};
class Wolf : public Animal {
public:
    virtual void say() {
        std::cout << "Woooo!" << std::endl;
    }
};
class Raccoon : public Animal {
public:
    virtual void say()  {
        std::cout << "Trill trill!" << std::endl;
    }
};
class Duck : public Animal {
public:
    virtual void say()  {
        std::cout << "Quack quack!" << std::endl;
    }
};
class Dolphin : public Animal {
public:
    virtual void say()  {
        std::cout << "Click click!" << std::endl;
    }
};
class Moose : public Animal {
public:
    virtual void say()  {
        std::cout << "Bellow bellow!" << std::endl;
    }
};
int main() {
    std::vector<Animal*> zoo;
    zoo.push_back(new Duck());
    zoo.push_back(new Dolphin());
    zoo.push_back(new Moose());
    zoo.push_back(new Raccoon());
    zoo.push_back(new Wolf());
    for(Animal * animal: zoo) {
        animal->say();
    }
    return 0;
}
```

Задача 2.

Создадим два наследника Callback — один для печати, другой для двойного применения. В обоих перегрузим виртуальную функцию exec.

В Printer будет просто принимать значение, которое нам будет выдавать Computator и просто будем выводить его.

Для двойного вычисления функции, нам потребуется сам объект Computator. Добавим во второй класс указатель на этот объект и запишем в него настоящий объект Computator с помощью конструктора. Далее для вывода создадим экземпляр уже созданного нами класса Printer для последующего вывода. После запустим вычисления передав Printer.

После этого остается создать соотвествующие объекты (и не забыть передать DoubledCompute указатель на оригинальный объект Computor). И далее мы запускаем вычисления передав вначале просто Printer, а потом DoubleCompute.

```
#include <iostream>
#include <cmath>

class Callback {
public:
    virtual void exec(double data) {}
};

class Computator {
public:
    void compute(double data, Callback * callback) {
        double result;
        result = exp(tan(fabs(sin(data))));
        callback->exec(result);
    }
};

class Printer : public Callback {
public:
    virtual void exec(double data)  {
        std::cout << data << std::endl;
    }
};

class DoubledCompute : public Callback {
public:
    DoubledCompute(Computator * c) {
        computator = c;
    }
    virtual void exec(double data)  {
        Callback * pr = new Printer();
        computator->compute(data, pr);
        delete pr;
    }
private:
    Computator * computator;
};

int main() {
      // создание основного объекта для вычислений
    Computator * computator = new Computator();
      // создание callback для вывода на экран результата
    Callback * printer = new Printer();
```

```
    // создание объекта для двойного вычисления
    // в качестве объекта для второго вычисления функции
    // мы передаем в конструктор все тот же исходный Computator
  Callback * doubled = new DoubledCompute(computator);
  double num;
  std::cin >> num;
    // запускаем вычисления у исходного объекта, передав
    // печатающий Callback
  computator->compute(num, printer);
    // запускаем вычисления у исходного объекта, передав
    // Callback, который запустит вычисления у исходного объекта еще раз
    // и после этого напечатает результат
  computator->compute(num, doubled);
  return 0;
}
```