概念题

1. 简述 C++中虚函数的概念，并说明虚函数有哪些作用。
   虚函数是指加了关键词 virtual 的成员函数。
   作用：实现消息的动态绑定，指出基类中可以被派生类重定义的成员函数。

2. 说明 C++中静态绑定和动态绑定的区别，在哪些情况下会发生动态绑定?
   静态绑定：向基类的指针或引用所指向的对象发送消息，调用基类成员函数处理。
   动态绑定：根据实际引用或指向的对象类型决定调用基类还是派生类成员函数。
   当基类中成员函数被定义为虚函数时:向基类的指针或引用实际指向或引用的是派生类的对象，且派生类重新定义了这个成员函数，则调用派生类重新定义的函数。

3. 简述 C++中抽象类的概念及作用。
   包含纯虚函数的类称为抽象类。
   抽象类的作用是为派生类提供基本框架和公共的对外接口。

编程题

1. 写出以下两段程序的输出，给出相应的说明并动手验证其正确性。

   1) 100 //int bar(char x)为静态绑定, Base* pObj 调用的是 int Base::bar(char x)
      50 //virtual int bar(int x)为动态绑定, Base* pObj 调用的是 int Derived::bar(int x)

   2) 5 //调用 void A::print()
      E //调用 void B::print()
      E //virtual void print()const 为动态绑定, A* p 指向的 B d2 调用 void B::print()
      5 //调用 void A::print()
      E // virtual void print()const 为动态绑定, A* d2 引用的 B d2 调用 void B::print()

2. 由 Animal 类派生出三个类，创建若干个对象并输出相关信息、叫声、体重之和。

```
class Animal {
protected:
    char name[10];
    int weight;
    static int total_weight;
public:
    Animal(const char* n, int w = 0) {
        strcpy(name, n);
        weight = w;
        total_weight += weight;
    }
    virtual void sound() = 0;
    void show() {
```

```
        cout << name << " weights " << weight << " kg." << endl;
    }
    ~Animal() {
        total_weight -= weight;
    }
    static int get_total() {
        return total_weight;
    }
};

int Animal::total_weight = 0;

class Dog :public Animal {
public:
    Dog(const char* n, int w = 0) : Animal(n, w) {}
    void sound() {
        cout << name << ": \"woof woof woof!\"" << endl;
    }
};

class Cat :public Animal {
public:
    Cat(const char* n, int w = 0) : Animal(n, w) {}
    void sound() {
        cout << name << ": \"mew~\"" << endl;
    }
};

class Cow :public Animal {
public:
    Cow(const char* n, int w = 0) : Animal(n, w) {}
    void sound() {
        cout << name << ": \"moooooooo\"" << endl;
    }
};

int main() {
    Dog Damo("Damon", 20);
    Dog Gra("Graham", 18);
    Dog Alex("Alex", 22);
    Dog Dave("Dave", 19);
    Cat Lili("Liam", 20);
    Cat Noel("Noel", 20);
    Cow Just("Justine", 18);
```

```
    Cow Don("Donna", 17);
    Cow Ann("Annie", 15);
    Damo.show();
    Damo.sound();
    Gra.show();
    Gra.sound();
    Alex.show();
    Alex.sound();
    Dave.show();
    Dave.sound();
    Lili.show();
    Lili.sound();
    Noel.show();
    Noel.sound();
    Just.show();
    Just.sound();
    Don.show();
    Don.sound();
    Ann.show();
    Ann.sound();
    cout << "The animals weigh " << Animal::get_total() << " kg in total."<< endl;
}
```

3. 创建三个学生类的对象，调用系统 display 接口输出所有学生姓名、平均成绩、绩点。

```
class Student {
protected:
    char name[10];
    double politics;
    double english;
    double average;
    double score_s;
public:
    Student(const char* n) {
        strcpy(name, n);
        politics = 0;
        english = 0;
        average = 0;
        score_s = 0;
    }
    void put_politics(double s) {
        politics = s;
    }
    void put_english(double s) {
```

```
        english = s;
    }
    virtual void score() {
        average = (politics + english) / 2.0;
        score_s =  average / 20.0;
    }
    void display() {
        score();
        cout << "Student " << name << "\'s average is " << average << "." << endl;
        cout << "Student " << name << "\'s score is " << score_s << "." << endl;
    }
};

class ComputerStudent :public Student {
protected:
    double programming;
public:
    ComputerStudent(const char* n) :Student(n) {
        programming = 0;
    }
    void put_programming(double s) {
        programming = s;
    }
    void score() {
        average = (politics + english + programming) / 3.0;
        score_s = programming / 20.0;
    }
};

class AIStudent :public ComputerStudent {
    double machine_learning;
public:
    AIStudent(const char* n) :ComputerStudent(n) {
        machine_learning = 0;
    }
    void put_machine(double s) {
        machine_learning = s;
    }
    void score() {
        average = (politics + english + programming + machine_learning) / 4.0;
        score_s = machine_learning / 20.0;
    }
};
```

```
void display(Student* stu) {
    stu->display();
}

int main() {
    Student* Students[4];

    Student John("John");
    Students[0] = &John;
    John.put_english(88);
    John.put_politics(93);

    ComputerStudent Paul("Paul");
    Students[1] = &Paul;
    Paul.put_english(85);
    Paul.put_politics(90);
    Paul.put_programming(96);

    ComputerStudent George("George");
    Students[2] = &George;
    George.put_english(91);
    George.put_politics(88);
    George.put_programming(87);

    AIStudent Ringo("Ringo");
    Students[3] = &Ringo;
    Ringo.put_english(90);
    Ringo.put_politics(91);
    Ringo.put_programming(95);
    Ringo.put_machine(86);

    for (int i = 0; i < 4; i++)
        display(Students[i]);
    return 0;
}
```