高级程序设计 **2020 春**

实验报告

指导老师：陈家骏 黄书剑 姓名：王晨渊, 学号：181220057

1 概念题

1.1 简述 C++ 中虚函数的概念，并说明虚函数有哪些作用。

虚函数是指声明时增加了 virtual 关键词的成员函数，一方面可以实现消息的动态绑定，另一方面可以指出基类中可以被派生类重定义的成员函数

1.2 说明 C++ 中静态绑定与动态绑定的区别，在哪些情况下会发生动态绑定？

静态绑定在编译时刻根据指针或引用的静态类型确定成员函数属于某个类，动态绑定在运行时刻根据指针或引用类型变量实际指向或引用的对象类型来确定成员函数属于某个类。

只有通过基类的指针或引用访问基类的虚函数时才进行动态绑定。基类的构造函数、析构函数中对虚函数的调用不进行动态绑定。

1.3 简述 C++ 中抽象类的概念及作用

抽象类是包含纯虚函数的类。抽象类的作用是为派生类提供一个基本框架和一个公共的对外接口。

2 编程题

2.1 阅读代码

2.1.1

我预测的输出结果是 100\n50\n，实际输出结果与预测一致。

2.1.2

我预测的输出结果是 5\nE\nE\n5\nE, 实际输出结果与预测一致

2.2 Anmial

```
1  #include <string>
2  #include <iostream>
```

```
3   using std::string;
4   using std::cout;
5   using std::endl;
6
7   class Animal
8   {
9   public:
10      virtual void sound()=0;
11      void show();
12      Animal();
13      Animal(string name, double weight);
14  private:
15      string name;
16      double weight;
17  };
18
19
20  class Dog:public Animal
21  {
22  public:
23      Dog(string name, double weight)
24          : Animal(name,weight)
25      {
26      }
27      void sound()
28      {
29          cout<< "woof woof woof!"<<endl;
30      }
31  };
32
33  class Cat:public Animal
34  {
35  public:
36      Cat(string name, double weight)
37          : Animal(name,weight)
38      {
39      }
40      void sound()
41      {
```

```
42              cout<< "mew~"<<endl;
43          }
44      };
45
46      class Cow:public Animal
47      {
48      public:
49          Cow(string name, double weight)
50              : Animal(name,weight)
51          {
52          }
53          void sound()
54          {
55              cout<< "moooooooooo"<<endl;
56          }
57      };
58
59      void Animal::show()
60      {
61          cout<<"Name:"<<name<<endl;
62          cout<<"Weight:"<<weight<<endl;
63      }
64
65      Animal::Animal(string n, double w)
66      {
67          name = n;
68          weight = w;
69      }
70
71      Animal::Animal()
72      {
73          name = "Not initialized!";
74          weight = 0;
75      }
```

2.3 Student manager

```
1       #include <iostream>
2       #include <string>
```

```
3   using std::cout;
4   using std::endl;
5   using std::string;
6
7   enum {POLITICS,ENGLISH,SOFTWARE,ML};
8
9   class Student
10  {
11  public:
12      Student(const string& name, double* scores);
13      virtual void score(double *score_list);
14      void display_name();
15  protected:
16      string name;
17      double politics;
18      double english;
19  };
20
21  class ComputerStudent:public Student
22  {
23  public:
24      ComputerStudent(const string& name, double *scores);
25      void score(double *score_list);
26  protected:
27      double software;
28  };
29
30
31  class AIStudent:public ComputerStudent
32  {
33  public:
34      AIStudent(const string& name, double *scores);
35      void score(double *score_list);
36  protected:
37      double machine_learing;
38  };
39
40
41  Student::Student(const string& n, double *scores)
```

```
42      {
43          name = n;
44          politics = scores[POLITICS];
45          english = scores[ENGLISH];
46      }
47
48
49      void Student::display_name()
50      {
51          cout<<name<<endl;
52      }
53
54      void Student::score(double *score_list)
55      {
56          score_list[0] = (english+politics)/2;
57          score_list[1] = score_list[0]/20;
58      }
59
60
61      ComputerStudent::ComputerStudent(const string& n, double *scores)
62          : Student(n,scores)
63      {
64          software =scores[SOFTWARE];
65      }
66
67      void ComputerStudent::score(double *score_list)
68      {
69          Student::score(score_list);
70          score_list[0] = (score_list[0]*2+software)/3;
71          score_list[1] = software/20;
72      }
73
74      AIStudent::AIStudent(const string& n, double *scores)
75          : ComputerStudent(n,scores)
76      {
77          machine_learing = scores[ML];
78      }
79
80
```

```
81      void AIStudent::score(double *score_list)
82      {
83          ComputerStudent::score(score_list);
84          score_list[0] = (score_list[0]*3+machine_learing)/4;
85          score_list[1] = machine_learing/20;
86      }
87
88
89      void display(Student* stu)
90      {
91          double scores_list[2]={0,0};
92          stu->display_name();
93          stu->score(scores_list);
94          cout<<"Average score:"<<scores_list[0]<<"GPA:"<<scores_list[1]<<endl;
95      }
```