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 分别简述 C++ 中单继承和多继承的概念

单继承指一个派生类只有一个直接基类。
多继承指一个派生类可以有两个或两个以上的直接基类。

1.2 说明 C++ 代码复用中，继承和聚集这两种方式的区别？

继承与封装存在矛盾，但集聚则没有。继承可以实现子类型，但是具有集聚关系的两个类不具备子类型关系。

1.3 简述 C++ 中虚基类的作用

解决重复继承问题带来的空间浪费与冗余的一致性检查。

2 编程题

2.1 阅读代码

我预测的输出结果是
string1
string2
string3
string4
destructor of D
destructor of C
destructor of B
destructor of A
实际输出结果与预测一致。

2.2 Phone

```
#include <iostream>
#include <cstdio>
#include <Windows.h>
#include <cstring>
#include<time.h>
using std::cout;
using std::endl;
using std::cin;
#pragma warning(disable:4996)
class Screen
{
public:
    void display(char* message, int len);
    Screen(int l, int w);
private:
    int length;
    int width;
};
int code[200];

class Mainboard
{
public:
    Mainboard();
    Mainboard(int d);
    void encode(char* message, int* code, int len);
    void decode(char* message, int* code, int len);
    int decoding(char s);
    int encoding(char s);
private:
    int delay;
};

class TestPhone
{
public:
    TestPhone();
    void sendMessage();
```

```
39          void sendMessage(char* message);
40          void receiveMessage();
41          void inputAndDisplay();
42          void inputAndDisplay(char* message);
43      protected:
44          Mainboard mainboard;
45          Screen screen;
46          int code_len;
47      };
48
49      class ReleasePhone:public TestPhone
50      {
51      public:
52          void dateAndTime();
53      };
54
55      Screen::Screen(int l, int w)
56      {
57          length = l;
58          width = w;
59      }
60
61      void Screen::display(char* message, int len)
62      {
63          len--;
64          char* s = new char[width+1];
65          s[width] = '\0';
66          int outp_cont = len / width * width == len ? len / width : len / width + 1;
67          for (int i = 0; i < outp_cont; i++)
68          {
69              if (i % length == 0 && i >= length)
70              {
71                  printf("\n");
72              }
73              if (i < outp_cont - 1)
74              {
75                  strncpy(s, &message[i * width], width);
76                  printf("%s\n", s);
77              }
```

```
78              else
79              {
80                  strncpy(s, &message[i * width], len-i*width);
81                  s[len - i * width] = '\0';
82                  printf("%s", s);
83              }
84          }
85      }
86
87      Mainboard::Mainboard()
88      {
89          delay = 0;
90      }
91      Mainboard::Mainboard(int d)
92      {
93          delay = d;
94      }
95      int Mainboard::encoding(char s)
96      {
97          if (s >= 'a' && s <= 'z')
98          {
99              return s - 'a';
100         }
101         if (s == '□')
102         {
103             return 26;
104         }
105         if (s == '\0')
106         {
107             return -1;
108         }
109         return s;
110     }
111
112     int Mainboard::decoding(char s)
113     {
114         if (s >= 0 && s <= 25)
115         {
116             return s + 'a';
```

```
117         }
118         if (s == 26)
119         {
120             return '□';
121         }
122         if (s == -1)
123         {
124             return '\0';
125         }
126         return s;
127     }
128
129     void Mainboard::encode(char* message, int* code, int len)
130     {
131         for (int i = 0;i < len; i++)
132         {
133             code[i] = encoding(message[i]);
134             Sleep(delay);
135         }
136     }
137
138     void Mainboard::decode(char* message, int* code, int len)
139     {
140         for (int i = 0; i < len; i++)
141         {
142             message[i] = decoding(code[i]);
143             Sleep(delay);
144         }
145     }
146
147     TestPhone::TestPhone()
148         :mainboard(1), screen(8,12)
149     {
150         code_len = 0;
151     }
152
153     void TestPhone::sendMessage()
154     {
155         char s[200];
```

```
156         cin.get(s, 200);
157         code_len = strlen(s)+1;
158         mainboard.encode(s, code, code_len);
159     }
160
161     void TestPhone::receiveMessage()
162     {
163         char s[200];
164         mainboard.decode(s, code, code_len);
165         screen.display(s, code_len);
166     }
167
168     void TestPhone::inputAndDisplay()
169     {
170         sendMessage();
171         receiveMessage();
172     }
173
174     void TestPhone::sendMessage(char* message)
175     {
176         code_len = strlen(message) + 1;
177         mainboard.encode(message, code, code_len);
178     }
179
180     void TestPhone::inputAndDisplay(char* message)
181     {
182         sendMessage(message);
183         receiveMessage();
184     }
185
186     void ReleasePhone::dateAndTime()
187     {
188         char system_time[200];
189         time_t now_time = time(NULL);
190         strcpy(system_time, asctime(localtime(&now_time)));
191         system_time[strlen(system_time) - 1] = '\0';
192         inputAndDisplay(system_time);
193     }
```