概念题

1. 单继承：派生类只有一个直接基类；多继承：派生类有一个以上的直接基类.
2. 聚集：把一个类作为另一个类(新类)的成员对象类.
   具有聚集关系的两个类之间通常属于整体与部分的关系.
   不能实现子类型，程序中旧类的对象不能用新类的对象替代;
   继承：在定义一个新的类时，先把已有程序中的一个或多个类的功能全部包含进来，
   然后在新的类中再给出新功能的定义或对已有类的某些功能重新定义.
   代码复用通过把一个类的代码包含(复制)到另一个类(新类)中实现.
   具有继承关系的两个类之间通常属于一般与特殊的关系
   实现了子类型，程序中基类的对象可以用派生类的对象替代.
   继承与封装存在矛盾，聚集则否.
   继承的代码复用功能常常可以用聚集来实现.
   继承更容易实现子类型，具有聚集关系的两个类不具有子类型关系.
3. 解决重复继承问题，使派生类的对象中只有一个该虚基类成员或成员函数

编程题

1. 猜想以下程序输出行数和输出内容，动手验证猜想并阐述真实输出的过程.
   string1　　//创建 D 类新对象的时候，先调用 A 的构造函数
   string2　　//再调用 B 的构造函数
   string3　　//再调用 C 的构造函数
   string4　　//最后调用 D 的构造函数
   destructor of D　　//D 类对象消亡的时候，先调用 D 的析构函数
   destructor of C　　//再调用 C 的析构函数
   destructor of B　　//再调用 B 的析构函数
   destructor of A　　//最后调用 A 的析构函数

   基类的声明次序决定基类构造函数/析构函数的调用次序，故顺序为 A-B-C/C-B-A.
   虚基类的构造函数由间接包含虚基类的类直接调用，故共有四次构造函数的调用.
   虚基类的构造函数优先非虚基类的构造函数执行.
   调用的构造函数及执行次序是 A(“string1”), B(“string1”, “string2”),
   C(“string1”, “string3”), D(“string4”)，析构函数为~D(), ~C(), ~B(), ~A().

2. 设有一个测试版手机类 TestPhone，包含屏幕 Screen 和主板 Mainboard.

```
#include <iostream>
#include <Windows.h>
#include<time.h>
#pragma warning (disable:4996);
using namespace std;

class Screen {
    int length;
```

```
    int width;
public:
    Screen(int l, int w) {
        length = l;
        width = w;
    }
    void display(char* message, int len) {
        int i, j, k;
        for (i = 0; i < length; i++) {
            for (j = 0; j < width; j++) {
                k = i * width + j;
                if (k == len)
                    goto L;
                else
                    cout << message[k];
            }
            cout << endl;
        }
    L:;
        cout << endl;
    }
};

class Mainboard {
    int delay;
public:
    Mainboard(int d) {
        delay = d;
    }
    void encode(char* message, int*& code, int len) {
        code = new int[len];
        for (int i = 0; i < len; i++) {
            if (message[i] == ' ')
                code[i] = 26;
            else if (message[i] == '\0')
                code[i] = -1;
            else
                code[i] = (int)(message[i] - 97);
            Sleep(delay);
        }
    }
    void decode(char*& message, int* code, int len) {
        message = new char[len];
        for (int i = 0; i < len; i++) {
```

```
            if (code[i] == 26)
                message[i] = ' ';
            else if (code[i] == -1)
                message[i] = '\0';
            else
                message[i] = (char)(code[i] + 97);
            Sleep(delay);
        }
    }
};

class TestPhone {
    Screen* screen;
    Mainboard* board;
public:
    TestPhone(int length, int width, int delay) {
        screen = new Screen(length, width);
        board = new Mainboard(delay);
    }
    void sendMessage() {
        cout << "Please imput message: ";
        char message[1000];
        cin.getline(message, 1000);
        int* code = NULL;
        cout << "Encoding message into code..." << endl;
        board->encode(message, code, strlen(message));
        cout << "Code: " << endl;
        char char_code[1000];
        int count = 0;
        for (int i = 0; i < strlen(message); i++) {
            if (code[i] == -1) {
                char_code[count] = '-';
                char_code[count + 1] = '1';
                count += 2;
            }
            else if (code[i] > 9) {
                char_code[count] = code[i] / 10 + 48;
                char_code[count + 1] = code[i] % 10 + 48;
                count += 2;
            }
            else {
                char_code[count] = code[i] + 48;
                count += 1;
            }
```

```
            char_code[count] = ' ';
            count += 1;
        }
        char_code[count] = '\0';
        screen->display(char_code, count);
    }
    void receiveMessage() {
        int len;
        cout << "Please imput length of code: ";
        cin >> len;
        int* code = new int[len];
        cout << "Please imput code: ";
        for (int i = 0; i < len; i++)
            cin >> code[i];
        char* message = NULL;
        cout << "Decoding code into message..." << endl;
        board->decode(message, code, len);
        cout << "Message: " << endl;
        screen->display(message, len);
    }
};

class ReleasePhone :public TestPhone {
public:
    ReleasePhone(int length, int width, int delay)
        :TestPhone(length, width, delay) {};
    void dateAndTime() {
        cout << endl;
        char system_time[200];
        time_t now_time = time(NULL);
        strcpy(system_time, asctime(localtime(&now_time)));
        cout << "The time is " << system_time;
    }
};

int main()
{
    ReleasePhone Phone(15, 5, 10);
    Phone.sendMessage();
    Phone.receiveMessage();
    Phone.dateAndTime();
    return 0;
}
```