

程设第十二次作业

20305055 熊明

1. 分析下列程序，写出运行时的输出结果：

```
1  class Date {
2  public:
3      Date(int, int, int);
4      Date(int, int);
5      Date(int);
6      Date();
7      void display();
8  private:
9      int month;
10     int day;
11     int year;
12 };
13 #include "date.h"
14 #include <iostream>
15 using namespace std;
16 Date::Date(int m, int d, int y):month(m),day(d),year(y){}
17
18 Date::Date(int m, int d) :month(m), day(d) { year = 2005; }
19
20 Date::Date(int m) :month(m)
21 {
22     day = 1;
23     year = 2005;
24 }
25
26 Date::Date() {
27     month = 1;
28     day = 1;
29     year = 2005;
30 }
31
32 void Date::display(){
33     cout << month << "/" << day << "/" << year << endl;
34 }
35
36 int main() {
37     Date d1(10, 13, 2005);
38     Date d2(12, 30);
39     Date d3(10);
40     Date d4;
41     d1.display();
42     d2.display();
43     d3.display();
44     d4.display();
45 }
```

分析：

d1会调用含三个参数的构造函数，所以d1的month是10，day是13，year是2005。

d2会调用含两个参数的构造函数，所以d2的month是12，day是30，year是默认值2005。

d3会调用含一个参数的构造函数，所以d3的month是10，day是默认值1，year是默认值2005。

d4会调用不含参的构造函数，所以d3的年月日是默认值1,1,2005。

2. 将1中的构造函数改为：

```
1 | Date::Date(int=1,int=1,int=2005);
```

分析程序所处的问题，并修改使得得到跟1一样的结果。

分析：

改为这种模式之后，就不需要那么多重载，可以将其他的重载函数删去，并且也要将无参数构造函数删去，只留下题目所给的构造函数。如下：

```
1 | class Date {
2 | public:
3 |     Date(int=1, int=1, int=2005);
4 |     void display();
5 | private:
6 |     int month;
7 |     int day;
8 |     int year;
9 | };
10 | #include "date.h"
11 | #include <iostream>
12 | using namespace std;
13 |
14 | Date::Date(int m, int d, int year)
15 | {
16 |     month = m; //对私有成员进行初始化
17 |     day = d;
18 |     this->year = year;
19 | }
20 | void Date::display(){
21 |     cout << month << "/" << day << "/" << year << endl;
22 | }
23 |
24 | int main() {
25 |     Date d1(10, 13, 2005);
26 |     Date d2(12, 30);
27 |     Date d3(10);
28 |     Date d4;
29 |     d1.display();
30 |     d2.display();
31 |     d3.display();
32 |     d4.display();
33 | }
```

得出结果与1一样。

10/13/2005
12/30/2005
10/1/2005
1/1/2005

3. 建立一个对象数组,内放 5 个学生的数据(学号、成绩),设立一个函数 max,用指向对象的指针作函数参数,在max 函数中找出5个学生中成绩最高者,并输出其学号。

```
1  //student.h
2  class student {
3  private:
4      char *num;
5      int score;
6  public:
7      void set_value(char*, int);
8      void display();
9      char* id();
10     int sc();
11 };
12 //student.cpp
13 #include"student.h"
14 #include<iostream>
15 using namespace std;
16
17 void student::set_value(char* num_n, int s) {
18     num = num_n;
19     score = s;
20 }
21
22 void student::display() {
23     cout << "ID: " << num << endl;
24     cout << "Score: " << score << endl;
25 }
26 char* student::id() {
27     return num;
28 }
29
30 int student::sc() {
31     return score;
32 }
33
34 void max(student* ptr) {
35     student* m = new student;
36     int mm = ptr->sc();
37     for (int i = 0; i < 5; i++) {
38         // (ptr+i)->display();
39         if ((ptr + i)->sc() >= mm) {
40             m = (ptr + i);
41             mm = (ptr + i)->sc();
42         }
43     }
44     m->display();
45 }
46
47 int main() {
```

```

48     student s[5];
49     for (int i = 0; i < 5; i++) {
50         char* id = new char;
51         int sc;
52         cout << "enter the student id: " << endl;
53         cin >> id;
54         cout << "enter the student score: " << endl;
55         cin >> sc;
56         s[i].set_value(id, sc);
57     }
58     max(s);
59 }

```

最开始，发现设置的每一个id都是最后的id。最后debug发现是因为每次循环虽然创建了新的变量char，但是地址未改变，每次输入都会导致前面的内容发生改变。最后用new动态构造解决了这个问题。输入输出结果如下：

```

enter the student id:
1001
enter the student score:
90
enter the student id:
1002
enter the student score:
96
enter the student id:
1003
enter the student score:
93
enter the student id:
1004
enter the student score:
91
enter the student id:
1005
enter the student score:
94
ID: 1002
Score: 96

```