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：

```
class Date {
public:
    Date(int, int, int);
    Date(int, int);
    Date(int);
    Date();
    void display();
private:
    int month;
    int day;
    int year;
};
#include"date.h"
#include<iostream>
using namespace std;
Date::Date(int m, int d, int y):month(m),day(d),year(y){}

Date::Date(int m, int d) :month(m), day(d) { year = 2005; }

Date::Date(int m) :month(m)
{
day = 1;
year = 2005;
}

Date::Date() {
    month = 1;
    day = 1;
    year = 2005;
}

void Date::display(){
    cout << month << "/" << day << "/" << year << endl;
}

int main() {
    Date d1(10, 13, 2005);
    Date d2(12, 30);
    Date d3(10);
    Date d4;
    d1.display();
    d2.display();
    d3.display();
    d4.display();
}
```

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：

```
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：

```
class Date {
public:
    Date(int=1, int=1, int=2005);
    void display();
private:
    int month;
    int day;
    int year;
};
#include"date.h"
#include<iostream>
using namespace std;

Date::Date(int m, int d, int year)
{
    month = m;  //对私有成员进行初始化
    day = d;
    this->year = year;
}
void Date::display(){
    cout << month << "/" << day << "/" << year << endl;
}

int main() {
    Date d1(10, 13, 2005);
    Date d2(12, 30);
    Date d3(10);
    Date d4;
    d1.display();
    d2.display();
    d3.display();
    d4.display();
}
```

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。

```
//student.h
class student {
private:
    char *num;
    int score;
public:
    void set_value(char*, int);
    void display();
    char* id();
    int sc();
};
//student.cpp
#include"student.h"
#include<iostream>
using namespace std;

void student::set_value(char* num_n, int s) {
    num = num_n;
    score = s;
}

void student::display() {
    cout << "ID: " << num << endl;
    cout << "Score: " << score << endl;
}
char* student::id() {
    return num;
}

int student::sc() {
    return score;
}

void max(student* ptr) {
    student* m = new student;
    int mm = ptr->sc();
    for (int i = 0; i < 5; i++) {
        // (ptr+i)->display();
        if ((ptr + i)->sc() >= mm) {
            m = (ptr + i);
            mm = (ptr + i)->sc();
        }
    }
    m->display();
}

int main() {
```

```
    student s[5];
    for (int i = 0; i < 5; i++) {
        char* id = new char;
        int sc;
        cout << "enter the student id: " << endl;
        cin >> id;
        cout << "enter the student score: " << endl;
        cin >> sc;
        s[i].set_value(id, sc);
    }
    max(s);
}
```

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