19-5-2-学生管理类（练习使用文件）

对于之前的学生管理类（学生姓名使用 MyString 类的对象来存储），设计保存学生信息和恢复学生信息的菜单，前者将输入的学生信息保存到一个文件中，后者从给定的文件中读取数据恢复之前学生管理类的状态。
请使用文本模式和二进制模式分别完成上述功能。

对这个程序不做更多解释。对于文件相关部分请看程序中的注释。另外要说明：
（1）MyString 类完成了之前作业要求的全部功能，也提供了文本模式和二进制模式的读写功能，请大家参考学习；
（2）CStudent 类中删除了之前作业中实现过但不必要实现的函数，如复制构造函数和赋值运算符函数，如果你还不太理解上述内容，请学习之前的作业；
（3）CStudent 类和 CStudentMng 类只实现了文本模式的读写功能。请大家自行完成二进制读写的功能。
（4）MyString 类没有写在一个名字空间中，而 CStudent 和 CStudentMng 写在一个名字空间中。请根据你的作业的情况完成作业，比如每一个类都有自己的头文件和实现文件、这些类位于一个名字空间中或位于全局名字空间中。

(1) MyString 类的头文件

```
#ifndef __MYSTRING_H__
#define __MYSTRING_H__
#include<iostream>
using namespace std;

class MyString
{
    static int total;    // 此处声明为私有数据成员
public:
    MyString();
    MyString(const char * p);
    MyString(const MyString & s);
    ~MyString();
    MyString & operator=(const MyString & s);

    // 取得字符串的首地址
    const char * get_string() const  { return m_pbuf; }
    // 将 p 指向的字符串保存在MyString类中
    const char * set_string(const char * p = NULL);
    // 将 p 指向的字符串追加到原有字符串之后
    const char * append(const char * p = NULL);
    // 将 s 对象中的字符串追加到当前对象的字符串之后并返回对象
    MyString & append(MyString & s);
    // 取得保存的字符串的长度
    int get_length() const { return strlen(m_pbuf); }
```

```
    static int get_total() { return total; }

    MyString & read_text(istream & in);       //文本模式读取函数
    MyString & read_binary(istream & in);//二进制模式读取函数
    void write_text(ostream & o);             //文本模式输出函数
    void write_binary(ostream & o);           //二进制模式输出函数
    //插入符，文本模式
    friend ostream & operator<< (ostream & o, const MyString & s);
private:
    char * m_pbuf;
    static int count;
};

#endif
```

(2) MyString 类的实现文件

```
#include"MyString.h"

int MyString::total = 0;

ostream & operator<< (ostream & o, const MyString & s)
{
    o<<s.get_length()<<" "<<s.m_pbuf; return o; //在长度后面加一个空格作为分隔符
}

MyString::MyString()
{
    total++;
    m_pbuf = new char('\0');
}

MyString::MyString(const char * p)
{
    total++;
    if (NULL == p)
        m_pbuf = new char('\0');
    else
    {
        int len = strlen(p) + 1;
        m_pbuf = new char[len];
        strcpy_s(m_pbuf, len, p);
    }
}
```

```
MyString::MyString(const MyString & s)
{
     total++;
     int len = strlen(s.m_pbuf) + 1;
     m_pbuf = new char[len];
     strcpy_s(m_pbuf, len, s.m_pbuf);
}

MyString::~MyString()
{
     total--;
     delete[] m_pbuf;
}

const char * MyString::set_string(const char * p)
{
     delete[] m_pbuf;

     if (NULL == p)
          m_pbuf = new char('\0');
     else
     {
          int len = strlen(p) + 1;
          m_pbuf = new char[len];
          strcpy_s(m_pbuf, len, p);
     }

     return m_pbuf;
}

const char * MyString::append(const char * p)
{
     if (NULL != p)
     {
          int len = strlen(m_pbuf) + strlen(p) + 1;
          char * tmp = new char[len];
          sprintf_s(tmp, len, "%s%s", m_pbuf, p);
          delete[] m_pbuf;
          m_pbuf = tmp;
     }

     return m_pbuf;
}
```

```
MyString & MyString::append(MyString & s)
{
    int len = strlen(m_pbuf) + strlen(s.m_pbuf) + 1;
    char * tmp = new char[len];
    sprintf_s(tmp, len, "%s%s", m_pbuf, s.m_pbuf);
    delete[] m_pbuf;
    m_pbuf = tmp;

    return *this;
}

MyString & MyString::operator=(const MyString & s)
{
    if (this != &s)  // 防止自赋值
    {
        delete[] m_pbuf;
        int len = strlen(s.m_pbuf) + 1;
        m_pbuf = new char[len];
        strcpy_s(m_pbuf, len, s.m_pbuf);
    }
    return *this;
}

MyString & MyString::read_text(istream & in)
{
    delete[] m_pbuf;  //首先回收之前的内存
    int len;
    in >> len;                    //读取字符串的长度
    m_pbuf = new char[len + 1];  //准备接受数据的空间
    in.get();                     //抛掉后面的空格
    in.read(m_pbuf, len);         //读取数据
    m_pbuf[len] = '\0';           //在最后加上结尾符号

    return *this;
}

MyString & MyString::read_binary(istream & in)
{
    delete[] m_pbuf;  //首先回收之前的内存
    int len;
    in.read((char *)&len, sizeof(int));  //先读取将要读取的字符串的长度
    m_pbuf = new char[len + 1]; //准备接受数据的空间：要多一个字节用于保存零字符
    in.read(m_pbuf, len);       //读取len个字节到m_pbuf指向的空间
    m_pbuf[len] = '\0';             //在最后加上结尾符号
```

```
    return *this;
}

void MyString::write_text(ostream & o)          // 文本模式输出函数
{
    o << *this;         //通过调用插入符函数实现文本模式的输出
}

void MyString::write_binary(ostream & o)  // 二进制模式输出函数
{
    int len = strlen(m_pbuf);   //取得字符串占用的空间的大小，即字符串的长度
    o.write((char *)&len, sizeof(int));  //在输出流中写入字符串所占空间的大小
    o.write(m_pbuf, len);//从m_pbuf开始写入len个字节；注意最后一个零字符并未写入
}
```

(3) 测试 MyString 类的文件读写功能的函数

为测试 MyString 类的文件读写功能，我们实现一个函数来测试文件读写的各种方式。你可以在 main 函数中调用该函数来完成测试。在测试函数之后给出了测试示例。

```
//通过测试返回true，否则返回false
bool testMyStringIO()
{
    MyString str("I love C++ programming!"), str2("haha");

    ofstream out("MyString.txt");
    //1. 文本模式：使用插入符写数据，使用read_text读数据
    out << str;
    out.close();

    ifstream in("MyString.txt");
    str2.read_text(in);
    in.close();
    //读取的字符串与原字符串不同则测试不通过
    if (strcmp(str2.get_string(), str.get_string()) != 0)
        return false;

    //2. 文本模式：使用write_text写数据，使用read_text读数据
    out.open("MyString.txt");   //打开文件并清空原数据
    str.write_text(out);
    out.close();

    in.open("MyString.txt");
    str2.read_text(in);
    in.close();
```

```
    if (strcmp(str2.get_string(), str.get_string()) != 0)
        return false;

    //3. 二进制模式：使用write_binary写数据，使用read_binary读数据
    out.open("MyString.dat");
    str.write_binary(out);
    out.close();

    in.open("MyString.dat");
    str2.read_binary(in);
    in.close();
    if (strcmp(str2.get_string(), str.get_string()) != 0)
        return false;

    return true;
}

int main()
{
    if (testMyStringIO())       //测试后MyString类的文件读写功能是否正确
        cout << "MyString IO OK" << endl;
    else
        cout << "MyString IO failed" << endl;
    return 0;
}
```

(4) CStudent 和 CStudentMng 类的头文件

```
#ifndef _123456_2_H_
#define _123456_2_H_

#include <iostream>
#include "MyString.h"
using namespace std;

namespace N123456
{
    class CStudent
    {
    public:
        CStudent(){ }
        CStudent(const int number, const MyString & name,
            const MyString & major, const double score)
            : number(number), name(name), major(major), score(score)
        { }
```

```
    int get_number() const { return number; }
    int set_number(const int number) { this->number = number; return number; }

    MyString get_name() const { return name; }     //返回值，常对象使用
    MyString & get_name() { return name; }         //返回引用，非常对象使用
    MyString & set_name(const char * s) { name.set_string(s); return name; }
    MyString & set_name(const MyString & name)
    { this->name = name; return this->name; }

    MyString get_major() const { return major; }
    MyString & get_major() { return major; }
    MyString & set_major(const char * s) { major.set_string(s); return major; }
    MyString & set_major(const MyString & major)
    { this->major = major; return this->major; }

    double get_score() const { return score; }
    double set_score(const double score) { this->score = score; return score; }
    double modifyScore(const double score)
    { this->score = score; return score; }

    friend ostream & operator<<(ostream & o, const CStudent & s);
    CStudent & read_text(istream & in)
    {
        in >> number;           //读取学号
        in.get();               //抛掉随后的一个空格
        name.read_text(in);     //读取学生的名字
        major.read_text(in);    //读取学生的专业
        in >> score;            //读取学生的成绩
        in.get();               //抛掉随后的空格

        return *this;
    }

    void display(ostream & o) const {
        o << number << "\t" << name.get_string() << "\t"
          << major.get_string() << "\t" << score << endl;
    }
private:
    int  number;
    MyString name;
    MyString major;
    double score;
};
```

```
    class CStudentMng
    {
    #define BLOCK 10
    public:
        CStudentMng();
        CStudentMng(const CStudent & stu);
        CStudentMng(const CStudentMng & stuMng);
        CStudentMng & operator=(const CStudentMng & stuMng);
        ~CStudentMng();

        void add_student(CStudent & stu);
        CStudent & getByName(char * name);
        const CStudent & getByName(char * name) const;
        bool existByName(char * name) const;
        CStudent & modifyScoreByName(char * name, double score);
        void display() const;
        void display(char * name) const;

        //清空当前对象中的数据，在read函数中会调用此函数，为读取数据准备条件
        void clear()
        {
            delete[] gStu;
            gStu = NULL;
            count = 0;
            available = 0;
        }

        friend ostream & operator<<(ostream & o, const CStudentMng & mng);
        CStudentMng & read_text(istream & in);
    private:
        CStudent * gStu;
        int count;//已经加入的学生个数
        int available;//可用的空间
    };

}

#endif
```

(5) CStudentMng 类的实现文件

```
#include "123456_2.h"

namespace N123456
{
```

```
ostream & operator<<(ostream & o, const CStudent & s)
{
    o << s.number << " " << s.name << s.major << s.score;
    o << " ";  //再输出一个空格，以防止两个学生的信息连在一起无法区分开
    return o;
}

//CStudentMng类的实现------------------------------------------------------
ostream & operator<<(ostream & o, const CStudentMng & mng)
{
    for (int i = 0; i < mng.count; i++)
        o << mng.gStu[i];

    return o;
}

CStudentMng & CStudentMng::read_text(istream & in)
{
    clear();              //清空数据，为读取数据做好准备
    CStudent s;
    while (true)
    {
        in.get();         //尝试读取一个字符，以检查是否到了文件尾
        if (in.eof())
            break;
        else  //没有到文件尾，所以回退一个字符，将之前多读取的字符送回输入流
            in.seekg(-1, ios_base::cur);

        s.read_text(in);
        add_student(s);
    }

    return *this;
}

CStudentMng::CStudentMng() : gStu(NULL), count(0), available(0)
{
}

CStudentMng::CStudentMng(const CStudent & stu) : count(1), available(BLOCK - 1)
{
    gStu = new CStudent[BLOCK];
    gStu[0] = stu;
}
```

```
CStudentMng::CStudentMng(const CStudentMng & stuMng)
{
    int num = stuMng.count + stuMng.available;
    if (0 == num)
        gStu = NULL;
    else
        gStu = new CStudent[num];
    count = stuMng.count;
    available = stuMng.available;

    for (int i = 0; i < stuMng.count; i++)
        gStu[i] = stuMng.gStu[i];
}

CStudentMng & CStudentMng::operator=(const CStudentMng & stuMng)
{
    if (this != &stuMng)
    {
        int num = stuMng.count + stuMng.available;
        if (0 == num)
            gStu = NULL;
        else
            gStu = new CStudent[num];
        count = stuMng.count;
        available = stuMng.available;

        for (int i = 0; i < stuMng.count; i++)
            gStu[i] = stuMng.gStu[i];
    }

    return *this;
}

CStudentMng::~CStudentMng()
{
    delete[] gStu;
}

void CStudentMng::add_student(CStudent & stu)
{
    if (0 == available)
    {//如果没有空间，需先扩展空间
        CStudent * tmp = new CStudent[count + BLOCK];
```

```
            for (int i = 0; i < count; i++)
                tmp[i] = gStu[i];

            delete[] gStu;
            gStu = tmp;
            available = BLOCK;
        }
        //增加学生信息
        gStu[count] = stu;
        count++;
        available--;
}

bool CStudentMng::existByName(char * name) const
{
        for (int i = 0; i < count; i++)
        {
            if (strcmp(name, gStu[i].get_name().get_string()) == 0)
                return true;
        }

        return false;
}

CStudent & CStudentMng::getByName(char * name)
{
        for (int i = 0; i < count; i++)
        {
            if (strcmp(name, gStu[i].get_name().get_string()) == 0)
                return gStu[i];
        }

        //此时找不到要获取的学生，所以意味着出错。由于还没有讲异常处理，
        //所以先返回一个非法的CStudent对象，以满足返回值类型的要求
        return gStu[-1];
}

const CStudent & CStudentMng::getByName(char * name) const
{
        for (int i = 0; i < count; i++)
        {
            if (strcmp(name, gStu[i].get_name().get_string()) == 0)
                return gStu[i];
        }
```

```
        //此时找不到要获取的学生，所以意味着出错。由于还没有讲异常处理，
        //所以先返回一个非法的CStudent对象，以满足返回值类型的要求
        return gStu[-1];

    }

    CStudent & CStudentMng::modifyScoreByName(char * name, double score)
    {
        CStudent & stu = getByName(name);
        stu.modifyScore(score);
        return stu;
    }

    void CStudentMng::display() const
    {
        cout << "学号\t姓名\t专业\t成绩" << endl;
        for (int i = 0; i < count; i++)
        {
            gStu[i].display(cout);
        }
        cout << endl;
    }

    void CStudentMng::display(char * name) const
    {
        cout << "学号\t姓名\t专业\t成绩" << endl;
        for (int i = 0; i < count; i++)
        {
            if (strcmp(gStu[i].get_name().get_string(), name) == 0)
                gStu[i].display(cout);
        }
        cout << endl;
    }
}
```

(5) main 函数的实现

```
#include "123456_2.h"
#include<fstream>
using namespace N123456;

int main()
{
    CStudentMng stuMng;
```

```
int choice = -1;
CStudent s;
while (choice != 0)
{
    cout << "1 录入学生信息 \n";
    cout << "2 显示学生信息 \n";
    cout << "3 通过名字修改成绩（方法一）\n";
    cout << "4 通过名字修改成绩（方法二）\n";
    cout << "5 测试复制构造函数（复制当前学生管理对象并显示学生信息）\n";
    cout << "6 测试有参构造函数\n";
    cout << "7 保存学生信息 \n";
    cout << "8 从文件中加载学生信息 \n";
    cout << "9 退出\n";
    cout << "请选择所需要的操作：";
    cin >> choice;

    if (1 == choice)
    {
        cout << "请依次输入学号、姓名、专业和成绩\n";
        char buf[64];
        int  tmp;
        cin >> tmp;
        s.set_number(tmp);
        cin >> buf;
        s.set_name(buf);
        cin >> buf;
        s.set_major(buf);
        cin >> tmp;
        s.set_score(tmp);
        stuMng.add_student(s);
    }
    else if (2 == choice)
        stuMng.display();
    else if (3 == choice)
    {
        cout << "请输入要查询的学生姓名：";
        char name[20];
        cin >> name;
        if (stuMng.existByName(name))
        {
            cout << "请输入要修改的成绩：";
            double score;
            cin >> score;
```

```
            stuMng.getByName(name).modifyScore(score);
            stuMng.display(name);
        }
        else
            cout << "查询的学生 " << name << " 不存在！\n\n";
    }
    else if (4 == choice)
    {
        cout << "请输入学生的姓名和成绩：";
        char name[20];
        double score;
        cin >> name >> score;
        if (stuMng.existByName(name))
            stuMng.modifyScoreByName(name, score);
        else
            cout << "学生 " << name << " 不存在！\n\n";
    }
    else if (5 == choice)
    {
        CStudentMng stuMngBak(stuMng);
        stuMngBak.display();
    }
    else if (6 == choice)
    {
        CStudent stu(1, "zhangsan", "computer", 100);
        CStudentMng stuMng(stu);
        stuMng.display();
    }
    else if (7 == choice)
    {
        ofstream out("test.txt");
        out << stuMng;
        out.close();
    }
    else if (8 == choice)
    {
        ifstream in("test.txt");
        stuMng.read_text(in);
        in.close();
    }
    else if(9 == choice)
        break;
    else
        cout << "你的选择有误，请重新选择！\n" << endl;
```

```
        }

        return 0;
    }
```