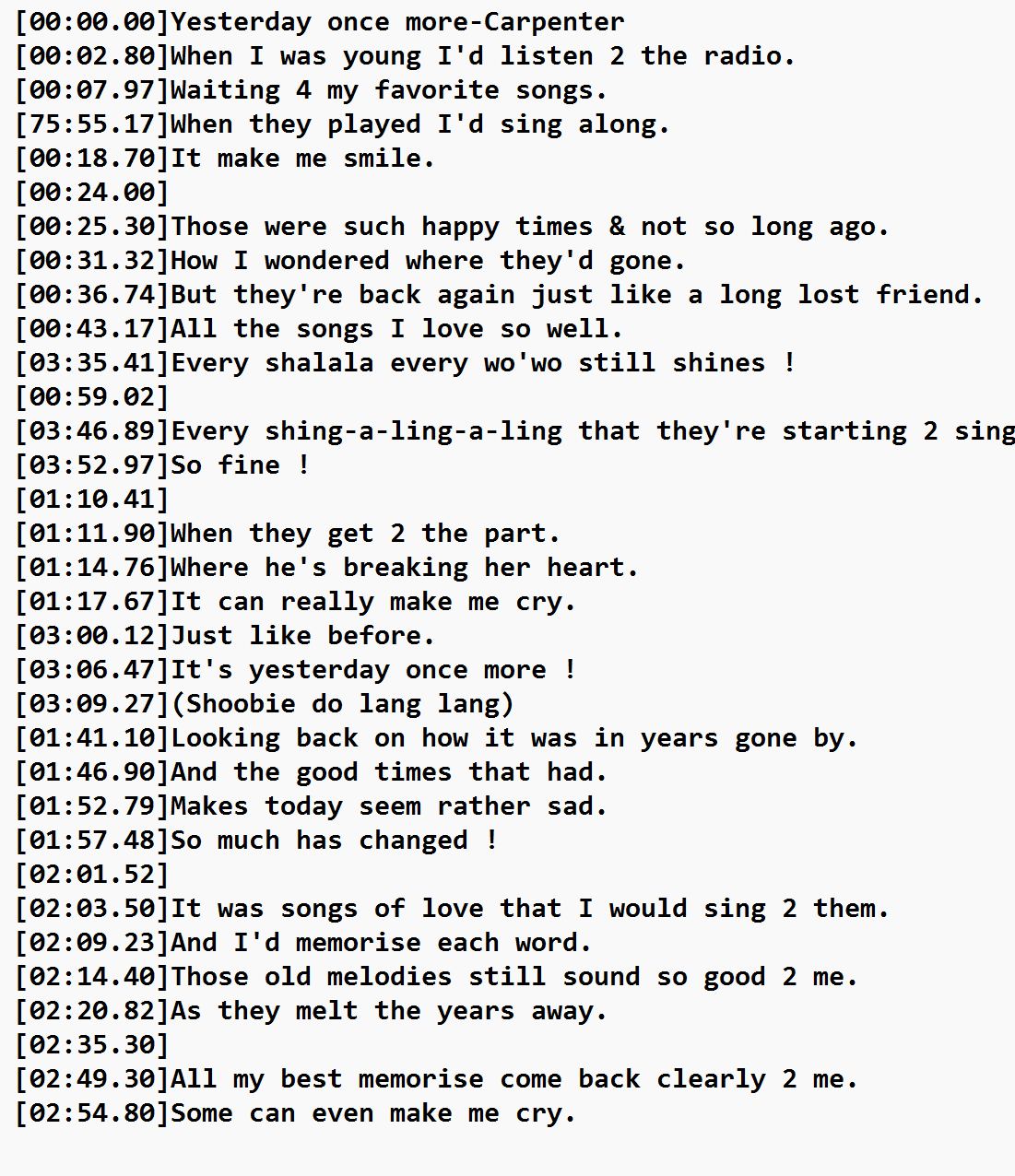
**这是 文件song.txt：**

**按以下要求编写程序**

**请各位考生从课程信息发布网站下载数据文件song.txt，然后将数据文件手动保存在D盘根目录下。并按要求编写一个类Clyric，存储文件song.txt中的数据。该文件是MP3音乐的歌词文件， 每一行存储一句歌词,格式如下：**

**[分钟(占两位):秒(占两位).百分之一秒数(占两位)]歌词文本**

**下面是两句歌词的例子：**

**[00:00.00]Yesterday once more-Carpenter**

**[03:46.89]Every shing-a-ling-a-ling that they're starting 2 sing.**

**Clyric类的要求如下：**

1. **该类的数据成员如下：**

class **Clyric**

{

private:

string m\_text; //歌词文本

int m\_min; //分钟

int m\_sec; //秒

int m\_cenSec;//百分之一秒数

static int m\_errCount;//记录对象构造失败的次数

public:

**};**

1. **该类public部分包含的函数如下：**

* **构造函数，歌词文本默认值为“”,其余参数默认值为0；**
  + **歌词文本长度应该小于等于100；**
  + **分钟应该大于等于0并且小于60；**
  + **秒应该大于等于0并且小于60；**
  + **百分之一秒数应该大于等于0，小于100；**
  + **若4个参数任意一个不合法，则歌词文本为“”,其余参数为0，并且**m\_errCount加1。
* **4个常成员函数，分别返回歌词文本、分钟、秒、百分之一秒数；**
* **一个静态成员函数，返回对象构造失败的次数；**
* **基于成员函数重载前自增操作符++，用于将当前对象的百分之一秒数加1。在实现的过程中需要注意进位问题。例如：01:59:99执行该操作的结果应该是02:00:00；59:59:59执行该操作的结果应该是00:00:00。**

1. **重载流输出运算符，显示一个对象的信息。其中时间部分用英文的方括号包含，分钟、秒数和百分之一秒数都占两位，不足两位前面补0，下面是输出的样例：**

**[00:00.00]Yesterday once more-Carpenter**

1. **通过友元函数实现重载+运算符，得到当前对象的百分之一秒数加上一个小于等于100的数字后的对象并返回，需要和++一样处理进位问题。**
2. **编写函数readData，完成数据的读入，结果存放在向量vec\_lyric中。**
3. **编写函数showData，用于把向量vec\_lyric的前n条数据显示在屏幕上。注意：显示格式在重载流输出运算符中已有描述；如果向量中的数据不足n条时，则显示向量中的全部数据。**
4. **编写函数deleteLyric，删除向量vec\_lyric中时间全为0且歌词是空“”的对象。**
5. **编写函数writeLyric，将上一步操作后的vec\_lyric中歌词不是空“”的对象顺序写入到文本文件中，每个对象占一行，写入时只需要写入歌词部分，不需要写时间信息。**
6. **main函数的具体内容如下：**

**程序的main函数（注意：不得修改main函数！对main函数每修改一处扣2分，最多扣10分。）**

**int main()**

**{**

**Clyric ly1("Soochow University", 7, 58, 99);**

**cout << "歌词为【" << ly1.getText()<<"】"<<endl;**

**cout << "时间为【" <<ly1.getMin()<<":"<<ly1.getSec()<<":"<<ly1.getCenSec()<<"】"<<endl;**

**Clyric ly2("Test", 17, 75, 54);**

**cout << "对象构造失败次数:" << ly1.getErrCount()<< endl;**

**cout << "调用前置++后:";**

**cout << ++ly1 << endl; //调用前置++**

**ly2 = ly1 + 100;**

**cout << "调用+运算符后:";**

**cout << ly2 << endl;**

**vector <Clyric> vec\_lyric;**

**if (readData("d:\\song.txt", vec\_lyric) == false)**

**{**

**cout << "打开文件失败";**

**return -1;**

**}**

**cout << "合计" << vec\_lyric.size() << "句歌词" << endl;**

**cout << "前5句歌词为" << endl;**

**showData(vec\_lyric, 5);**

**int count = deleteLyric(vec\_lyric);**

**cout << "删除了" << count << "句歌词" << endl;**

**count = writeLyric("d:\\text.txt", vec\_lyric);**

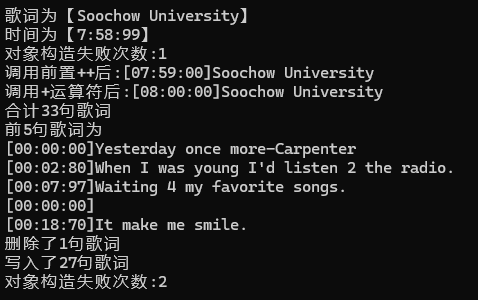
**cout << "写入了" << count << "句歌词" << endl;**

**cout << "对象构造失败次数:" << ly1.getErrCount() << endl;**

**return 1;**

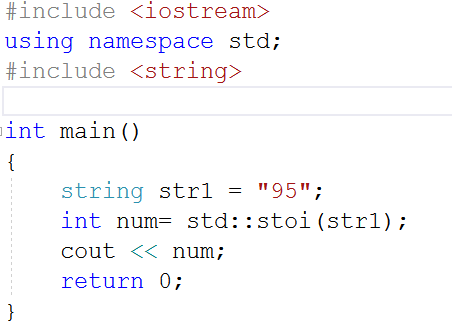
**}**

**程序运行结果：（不一定是标准答案）**

****

提示：如何将数字字符串转换为int数值？

有多种方法可以进行，下面通过一个样例给出一种方法。



#include<iostream>

#include<sstream>

#include<iomanip>

#include<vector>

#include<fstream>

using namespace std;

#include<string>

class Clyric

{

private:

string m\_text; //歌词文本

int m\_min; //分钟

int m\_sec; //秒

int m\_cenSec;//百分之一秒数

static int m\_errCount;//记录对象构造失败的次数

public:

//1

Clyric(string text = "", int min = 0, int sec = 0, int cenSec = 0)

{

if (text.size() <= 100 && (min >= 0 && min < 60) &&

(sec >= 0 && sec < 60) && (cenSec >= 0 && cenSec < 100))

{

m\_text = text;

m\_min = min;

m\_sec = sec;

m\_cenSec = cenSec;

}

else

{

m\_text = "";

m\_min = 0;

m\_sec = 0;

m\_cenSec = 0;

m\_errCount++;

}

}

string getText() const { return m\_text; }

int getMin() const { return m\_min; }

int getSec() const { return m\_sec; }

int getCenSec() const { return m\_cenSec; }

static int getErrCount() { return m\_errCount; }

Clyric& operator++()

{

m\_cenSec++;

if (m\_cenSec >= 100)

{

m\_cenSec = 0;

m\_sec++;

if (m\_sec >= 60)

{

m\_sec = 0;

m\_min++;

if (m\_min >= 60)

{

m\_min = 0;

}

}

}

return \*this;

}

//3

friend ostream& operator<<(ostream& out, const Clyric& r)

{

out << '[' << setfill('0') << setw(2) << r.m\_min << ':'

<< setw(2) << r.m\_sec << '.'

<< setw(2) << r.m\_cenSec << ']'

<< r.m\_text;

return out;

}

//4

friend Clyric operator+(const Clyric& r, int num)

{

Clyric temp;

temp.m\_text = r.m\_text;

temp.m\_min = r.m\_min;

temp.m\_sec = r.m\_sec;

temp.m\_cenSec = r.m\_cenSec + num;

if (temp.m\_cenSec >= 100)

{

temp.m\_cenSec -= 100;

temp.m\_sec++;

if (temp.m\_sec >= 60)

{

temp.m\_sec = 0;

temp.m\_min++;

if (temp.m\_min >= 60)

{

temp.m\_min = 0;

}

}

}

return temp;

}

};

int Clyric::m\_errCount = 0;

//5

// way 1 error

//bool readData(string path, vector <Clyric>& vec\_lyric)

//{

// ifstream file(path);

// if (!file.is\_open()) return false;

// string line;

// while (getline(file, line))

// {

// istringstream iss1;

// char d1;

// int min;

// char d2;

// int sec;

// char d3;

// int censec;

// char d4;

// string t;

// iss1 >> d1 >> min >> d2 >> sec >> d3 >> censec >> d4 >> t;

//

// Clyric temp(t, min, sec, censec);

// vec\_lyric.push\_back(temp);

// }

// file.close();

// return true;

//}

// way 2 stoi belongs to <string>

bool readData(string path, vector<Clyric>& vec\_lyric)

{

ifstream file(path);

if (!file.is\_open())

return false;

string line;

while (getline(file, line))

{

if (line.size() < 10 || line[0] != '[' || line[9] != ']') {

continue; // 如果行格式不正确，跳过

}

string timePart = line.substr(1, 8); // 提取时间部分 [00:00.00]

string textPart = line.substr(10); // 提取歌词部分

int min = stoi(timePart.substr(0, 2)); // 提取并转换分钟

int sec = stoi(timePart.substr(3, 2)); // 提取并转换秒

int cenSec = stoi(timePart.substr(6, 2)); // 提取并转换百分之一秒

Clyric temp(textPart, min, sec, cenSec);

vec\_lyric.push\_back(temp);

}

file.close();

return true;

}

//way 3

//bool readData(string path, vector<Clyric>& vec\_lyric)

//{

// ifstream file(path);

// if (!file.is\_open()) return false;

//

// string line;

// while (getline(file, line))

// {

// istringstream iss(line);

// char d1, d2, d3, d4;

// int min, sec, censec;

// string text;

//

// // 解析时间部分

// if (iss >> d1 >> min >> d2 >> sec >> d3 >> censec >> d4)

// {

// // 获取歌词文本部分

// getline(iss, text);

//

// // 去掉歌词文本前面的空格

// text.erase(0, text.find\_first\_not\_of(" "));

//

// Clyric temp(text, min, sec, censec);

// vec\_lyric.push\_back(temp);

// }

// }

//

// file.close();

// return true;

//}

//6

void showData(vector <Clyric>& vec\_lyric, int n)

{

for (int i = 0; i < vec\_lyric.size() && i < n; i++)

{

cout << vec\_lyric[i] << endl;

}

}

//7

int deleteLyric(vector <Clyric>& vec\_lyric)

{

int count = 0;

for (int i = 0; i < vec\_lyric.size();)

{

if (vec\_lyric[i].getCenSec() == 0 &&

vec\_lyric[i].getSec() == 0 &&

vec\_lyric[i].getMin() == 0 &&

vec\_lyric[i].getText() == "")

{

vec\_lyric.erase(vec\_lyric.begin() + i);

count++;

}

else

{

i++;

}

}

return count;

}

//8

int writeLyric(string newf, vector <Clyric>& vec\_lyric)

{

int count1 = 0;

ofstream outfile(newf);

for (int i = 0; i < vec\_lyric.size(); i++)

{

if (vec\_lyric[i].getText() != "")

{

count1++;

cout << vec\_lyric[i].getText();

}

}

outfile.close();

return count1;

}

int main()

{

Clyric ly1("Soochow University", 7, 58, 99);

cout << "歌词为【" << ly1.getText() << "】" << endl;

cout << "时间为【" << ly1.getMin() << ":" << ly1.getSec() << ":" << ly1.getCenSec() << "】" << endl;

Clyric ly2("Test", 17, 75, 54);

cout << "对象构造失败次数:" << ly1.getErrCount() << endl;

cout << "调用前置++后:";

cout << ++ly1 << endl; //调用前置++

ly2 = ly1 + 100;

cout << "调用+运算符后:";

cout << ly2 << endl;

vector <Clyric> vec\_lyric;

if (readData("d:\\song.txt", vec\_lyric) == false)

{

cout << "打开文件失败";

return -1;

}

cout << "合计" << vec\_lyric.size() << "句歌词" << endl;

cout << "前5句歌词为" << endl;

showData(vec\_lyric, 5);

int count = deleteLyric(vec\_lyric);

cout << "删除了" << count << "句歌词" << endl;

count = writeLyric("d:\\text.txt", vec\_lyric);

cout << "写入了" << count << "句歌词" << endl;

cout << "对象构造失败次数:" << ly1.getErrCount() << endl;

return 1;

}