目标：

**构件库部分：**

构建记录调试信息的类。

**应用验证部分：**

应用记录调试信息的类验证第13讲双向链表程序的new和delete是否匹配。

原理：

编写构件库程序，定义一个构建记录调试信息的类，具体如下所示：

class FH\_Debug\_Log

{

public:

FH\_Debug\_Log();

FH\_Debug\_Log(string Path);

~FH\_Debug\_Log();

template <typename T>

FH\_Debug\_Log& operator <<(const T& data)

{

if (recordData)

{

outStream.open(filePath, ios::out | ios::app);

outStream << data;

outStream.close();

}

return(\*this);

}

map<int, Class\_State> classStates;

void clear();

private:

string filePath;

ofstream outStream;

};

为了判断是否为debug模式（是否需要输出），定义：

extern bool recordData;

在cpp文件中有：

#ifdef \_DEBUG

bool recordData = true;

#else

bool recordData = false;

#endif // \_DEBUG

为了方便记录每个指针的现有情况，定义：

enum Class\_State { NotNewed, HaveNewed, HaveDeleted, DeleteWithoutNew, DoubleNewed, ClassStateNum };

编写手动测试与自动测试函数，分别提供不同的运行接口。在自动测试部分，使用输出重定向，将输出到显示器的部分重定向到文本文件中，方便用文件输入流进行比较判断程序运行错误与否。

在13讲程序的new和delete处，添加输出代码。

**如何编程和运行：**

* 软件构件库部分：

编写原理中所述函数的具体实现。

* 构件库测试部分：

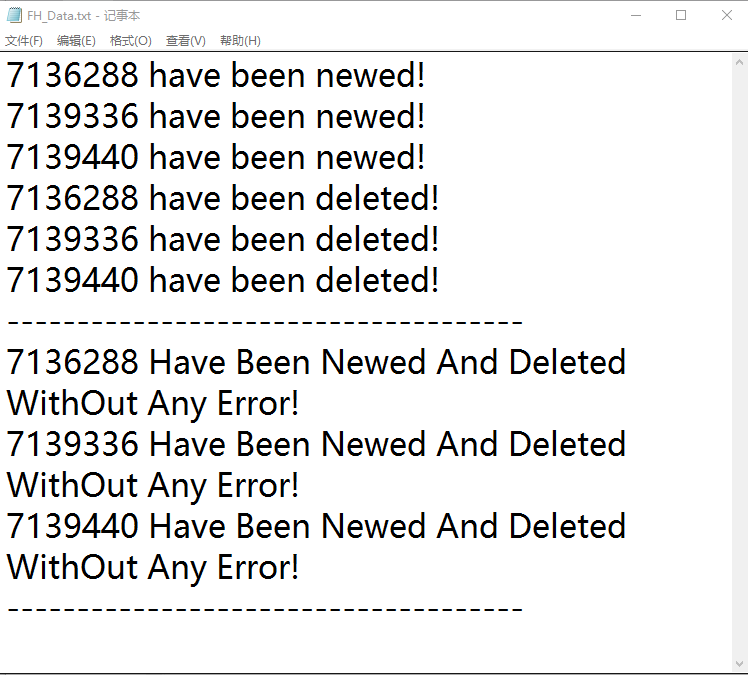
编写原理中所述自动测试与手动测试部分的代码实现。

* 构件库应用部分：

建立工程，编写 main() 函数并且选择手动测试与自动测试的功能，之后调用类中的函数进行操作。

结果：

以下测试结果均在64位操作系统下用VS 2017 测试得到。



程序验证结果报告：

* 对于64位的计算机

在上述的程序运行过程中未发现错误，在程序运行过程中可以方便地发现程序的错误报告，同时自动测试的部分显示重点突出，方便使用。

结论：

程序能较好完成设计目的要求，在使用的过程中未发现错误。