**程序逻辑**

lab3中共有6个文件：ListNode.h、LIstBuffer.cc、ListBuffer.h、Editor.cc、Editor.h和ed.cc。其中ListNode.h、ListBuffer.cc/h、Editor.cc/h分别声明和实现了三个类，ed.cc中包含了main函数。

**ListNode**

ListNode.h定义了ListNode类，作为节点储存输入的函数（LineNumber）、文本（statement）和下个节点的位置指针（\*next）,构造函数通过初始化列表赋值。

**ListBuffer**

ListBuffer.h声明了ListNode类，其作用为作为链表存放输入的文本。其私有成员为已记录总行数（currentLength）、已记录节点中的最大行数（maxLine）、头指针（\*head，默认为0行）以及move函数（用来移到特定节点）。其公有成员函数实现了构造链表（ListBuffer）、写入文件（writeToFile）、清除链表（clear）、输出文件（showLines）、删除行（deleteLine）、插入行（insertLine）和析构链表（~ListBuffer）的操作。

ListBuffer实现了ListBuffer类，实现思想如下：

**ListBuffer()**

初始化已记录总行数(currentLength)和已记录节点中的最大行数(maxLine)为0，初始化头指针（head）。

**void writeToFile(const string &filename) const**

实现思想如lab2。

**void clear()**

从头指针开始逐个删除节点。

**void showLines() const**

从头指针开始逐个输出节点的行数和文本

**void insertLine(int line\_idx, const string &text)**

这个就有意思了，分类讨论：

1. 插入行比当前最大行大，移到当前最大行（pos），在当前最大行后面插入一个新的节点，同时最大行变成新插入的行的行数，总行数加1。
2. 插入行没有当前行大，先找到一个节点（p）满足其行数小于插入行同时下一个节点（p->next）的行数大于等于插入行，然后if判断插入行是否为已有行，如果是，就删除原有行；反之，则不用。接下来在p的后面插入一个新节点。

**void deleteLine(int line\_idx)**

找到要删除的节点的前一个节点（p），用delp保存要删除的节点，将p的指针指向delp->next，再删除delp。

**Editor**

Editor.h声明了Editor类，其私有成员为ListBuffer类的对象buffer和处理函数（dispatchCmd），其公有成员为构造函数（Editor），析构函数（~Editor），运行函数（run）

Editor.cc实现了Editor类，实现思想如下：

**Editor()**

构造一个ListBuffer类的对象buffer。

**~Editor()**

删除 buffer。

**void run()**

实现思想如lab2。

**void dispatchCmd(const string &cmd)**

分类讨论：

1. 输入为数字，说明是插入操作，用linenumber储存行数，text存储文本，调用buffer的insertLine函数。
2. 输入为‘d’，说明是删除操作，调用buffer的deleteLine函数。
3. 输入为“save”，说明是储存操作，调用buffer的writeToFile函数。
4. 利用string库里的transform函数，将输入全都大写化，判断是否为“LIST”，若为真，则调用buffer的showLine函数。
5. 无效操作，return。

**ed**

ed.cc中包含了main函数。