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：

A、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。

B、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：

A、 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。

B、 输入为‘d’，说明是删除操作，调用 buffer 的 deleteLine 函数。

C、 输入为“save”，说明是储存操作，调用 buffer 的 writeToFile 函数。

D、 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。

E、 无效操作，return。

ed

ed.cc 中包含了 main 函数。