Lab2: MiniEd

Ed是一个非常古老的行文本编辑器，曾经被用于文本文件的创建、显示、更改和其他一些操作。

Ed有两种模式，刚运行时Ed处于命令模式（command mode）。此时输入到Ed的文本会被当做命令进行处理。通过一些特殊命令，Ed可以被切换到输入模式（input mode），在输入模式下输入的文本会被当做文件内容进行保存。

本次Lab的任务是使用C++通过面向对象的方法完成一个简化版的Ed编辑器MiniEd。在MiniEd中，仅需实现Ed命令的一个子集。

命令与符号解释

在以下列出的Ed命令中，每个`?`为一个数字n，表示第n行。其值应该在第一行到最后一行之间（包含第一行和最后一行）。

每个`?,?`表示一个区间，其中`?`的含义同上。如`1,2`表示第一行到第二行；`5,11`表示第五行到第十一行；`6,6`表示第六行。应保证区间起点不大于终点。

每个命令为一行输入，以下为需要实现的命令说明。

> - `a`
>
>   Appends text to the buffer after the current address line. If the current address is zero, the entered text is placed at the beginning of the buffer. Text is entered in input mode. The current address is set to the address of the last line entered or, if there were none, the current address is not changed.
>
> - `?,?d`
>
>   Deletes the addressed lines from the buffer. The current address is set to the new address of the line after the last line deleted; if the lines deleted were originally at the end of the buffer, the current address is set to the address of the new last line; if no lines remain in the buffer, the current address is set to zero.
>
> - `i`
>
>   Inserts text in the buffer before the current addressed line. If the current address is zero, the entered text is placed at the beginning of the buffer. Text is entered in input mode. The current address is set to the address of the last line entered or, if there were none, the current address is not changed.
>
> - `?,?n`
>
>   Number command. Prints the addressed lines, preceding each line by its line number and a <tab> (`'\t'`). The current address is set to the address of the last line printed. Specially, `1,$n` prints all lines in the buffer.
>
> - `Q`

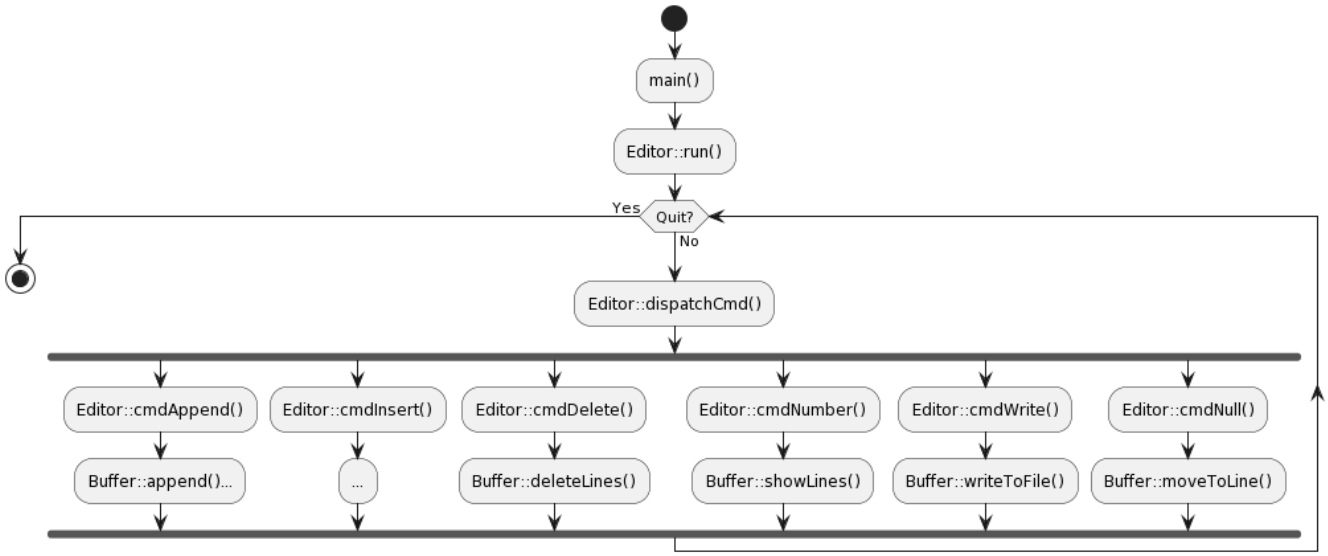
> Quits ed unconditionally. Unwritten changes are discarded.
>
> - `w file`
>
>   Writes all lines to file and prints the number of bytes written to the file. Print an error if no filename is specified. The current address is unchanged.
>
> - `?`
>
>   Null command. An address alone prints the addressed line. The current address is set to the address of the printed line.

对于无效指令或错误指令，请输出一个以问号开头的错误提示信息：

```
? Bad/Unknown command
```

程序大体逻辑

MiniEd的部分代码已经给出，共五个文件：`Buffer.cc`、`Buffer.h`、`Editor.cc`、`Editor.h`和`ed.cc`。其中`Buffer.cc/h`、`Editor.cc/h`分别声明和实现了两个类。`ed.cc`中包含了`main`函数，并在其中建造并运行了一个`Editor`实例。在`Editor::run`中，程序会不断读入命令，对命令进行解析并调用相应的函数（`Editor::cmdXXX`）。`Buffer`类用于保存所有输入的文本，其中每行文本保存在一个`string`类型的字符串中，所有的行保存在一个链表中。`Buffer`类有各种用于操作和展示文本的函数，供`Editor`调用。



任务

代码：

1. 实现`Buffer::writeToFile`，将当前所有的文本保存到指定文件中，并打印所保存的字节数。
2. 实现`Buffer::showLines`，将相应的文本行以规定格式打印出来。
3. 实现`Buffer::insertLine`，将指定文本插入到当前行之前。
4. 实现`Editor::cmdAppend`，完成对`a`命令的处理。
5. 在`Editor::dispatchCmd`中，完成对`cmdWrite`的调用。

6. 在`Editor::dispatchCmd`中，完成对`1,$n`情况的判断，以及后续的处理。
7. 完成`Editor`的析构函数。
8. 添加对各种异常情况的处理。异常情况如指定了错误的行号区间、指定了超出范围的行号等。
9. 在熟悉代码逻辑的情况下，增加一个新命令的支持，`?,?s`命令：将指定的两行文本进行替换，注意是交换两个链表节点，不是里面的内容，提供的`Buffer.cc`代码中，`loadAddr2`和`testSwap`以及`printAddr`是用来测试swap前后，节点地址是否发生了转换

链表还需实现其构造函数，不可使用任何第三方库的实现进行替换。

除`ed.cc`的代码和`Buffer.cc`代码中的`loadAddr2`，`testSwap`和`printAddr`外，可以随意更改代码，实现了要求的功能即可。（提供文件中的代码可能会有问题，需要大家发现并改正）

回答问题：

1. 请解释在`Editor::dispatchCmd`中下述代码的逻辑，尤其注意对stringstream的使用。

```
    int start, end;
    char comma, type = ' ';
    stringstream ss(cmd);
    ss >> start;
    if (ss.eof()) {
        cmdNull(start);
        return;
    }
    ss >> comma >> end >> type;
    if (ss.good()) {
        if (type == 'n') {
            cmdNumber(start, end);
            return;
        } else if (type == 'd') {
            cmdDelete(start, end);
            return;
        }
    }
```

使用样例

```
cmd> a
It's input mode now. Quit with a line with a single dot(.)
#include <iostream>

int main()
{
}
.
cmd> 1,$n
1       #include <iostream>
2
3       int main()
```

```
4       {
5       }
cmd> 3
int main()
cmd> i
It's input mode now. Quit with a line with a single dot(.)
using namespace std;

#includ
.
cmd> 1,$n
1       #include <iostream>
2
3       using namespace std;
4
5       #includ
6       int main()
7       {
8       }
cmd> 4,5d
cmd> 1,$n
1       #include <iostream>
2
3       using namespace std;
4       int main()
5       {
6       }
cmd> 1,3c
? Bad/Unknown command
cmd> 1,5n
1       #include <iostream>
2
3       using namespace std;
4       int main()
5       {
cmd> a
It's input mode now. Quit with a line with a single dot(.)
    cout << "Hello my editor" << endl;

    return 0;
.
cmd> 1,$n
1       #include <iostream>
2
3       using namespace std;
4       int main()
5       {
6           cout << "Hello my editor" << endl;
7
8           return 0;
9       }
cmd> 1,4l
cmd> 1,4s
cmd> 1,4t
```

```
Swap 2 Nodes successfully!
cmd> 1,$n
1       int main()
2
3       using namespace std;
4       #include <iostream>
5       {
6           cout << "Hello my editor" << endl;
7
8           return 0;
9       }
cmd> write
? Bad/Unknown command
cmd> 1,999n
? Line number out of range
cmd> 9,1n
? Number range error
cmd> 4,2d
? Delete range error
cmd> 2019
? Line number out of range
cmd> w hello.cc
115 byte(s) written
cmd> Q
```

输出格式规范

- 你的程序应该从标准输入`stdin`中获取输入，输出到标准输出`stdout`
- 输出行时，行号和内容之间用一个制表符（\t）隔开
- Write指令成功后输出写入的字节数：`%d byte(s) written`
    - 统计输出字节数需要统计可见的字符和所有不可见的字符（包括所有空格和每行末尾的换行符），按照ASCII编码每个字符占一字节。一般来说，使用流输出或者C语言的输出函数时，可以通过相关函数或者输出函数返回值得知成功输出的字节数。自己计算字符串长度统计也可以。
- 错误处理：
    - 指令错误：`? Bad/Unknown command`
    - 行号越界：`? Line number out of range`
    - Number指令中范围错误：`? Number range error`
    - Delete指令中范围错误：`? Delete range error`
    - Write指令中未指定文件名：`? Filename not specified`

  补充说明：

  非数字的范围判定为指令错误。例：`1,xn`判定为指令错误，而非范围错误

  范围错误优先于行号越界。例：`999,1n`判定为范围错误，而非行号越界

- 提示性的输出不作规定。例如样例中的`cmd>`以及`It's input mode now. Quit with a line with a single dot(.)`

提交

提交时，请将

- 你完成的MiniEd源代码和你对上述问题的回答`lab2-XXX.pdf`放在同一个文件夹下，该文件夹名应为`lab2`
- 将`lab2`文件夹压缩成7z压缩包，并重命名为`lab2-XXX.7z`

上传到 canvas 中。

文件结构如下：

```
lab2-XXX.7z
    ├── Buffer.cc
    ├── Buffer.h
    ├── CMakeLists.txt
    ├── ed.cc
    ├── Editor.cc
    ├── Editor.h
    ├── lab2-XXX.pdf
    └── spec
        ├── lab2.md
        └── lab2.pdf
```

（其中XXX为学号，如`lab2-518037910001.7z`和`lab2-518037910001.pdf`）

如果需要更改，请在文件名后加版本号，最终以最高版本号为准。如第二次提交可用`lab2-518037910001-2.7z`。

其余要求同 lab1。

注1：未按照格式提交可能会引起评测错误，导致零分

注2：建议将代码写入提供的框架代码文件中。将代码写入新建的源代码文件可能会导致评测时编译失败

注3：本次实验可以多次提交，每30分钟将会进行一次自动评测