数算实习第一次大作业实习报告

小组成员：
1200012741 史杨勋惟
1200012768 陈庆英
1200012826 姜双
1200012863 刘敏行

主要设计思想：

以Linux下的Ncurses函数库为主要工具，以Vi的基本操作为主要目标，以带通配符的KMP为主要算法支持，进行了整个程序的设计。主要实现的文本编辑功能有，基本插入，删除，带通配符的字符串匹配，全文替换等。

数据结构：

一共有如下三个主要数据结构

class Text：

基于文本的数据结构，主要实现的功能是，文本内部的数据存储，数据变更，以及字符串的匹配和替换。内部成员为两个字符串和一个字符串数组，两个字符串分别是content（原文本内容），buffer（变更文本内容），以及一个字符串数组rows将每行字符串分别保存起来。这样的好处是在对于一些行操作上能更方便的处理。Text类确保了文本内部的操作的正确性，是正确屏幕输出的重要保证。

class Screen:

面向用户的与屏幕相关的数据结构，主要功能是实现良好的人机交互，类中的主要成员为一个Cursor对象和Text对象。在文本编辑的过程中，Screen类负责读取用户的指令并通过调用cursor对象以及text对象的相关成员函数，再通过一系列的辅助操作使得屏幕能够正常输出。Ncurses的函数库是这个类的主要支持。

Class Cursor：

控制光标以及一些其他位置的类，类中有大量与位置相关的成员，其主要目的是保证光标所在窗口的位置，光标所在文本中的位置，屏幕输出起始位置在文本中的位置的正确性，是Screen类正确输出的另一方面的保证。与此同时Cursor类也起到了外部的输出与内部的文本中的衔接作用，在整个运行中起着关键的作用。

程序执行流程：

执行程序后，Text类根据指定文本率先初始化，随后Screen类初始化并生成输出窗口，与此同时初始化Screen类中的Cursor类，当用户有正确按键的操作时，Screen类会识别该操作的类型，并调用相关的cursor对象或者text对象的成员函数，其中cursor类保证了移动位置的正确性，screen类保证了输出文本的正确性，借助这两个类的辅助，Screen类通过show_text函数将整个显示内容正确地输出在屏幕上。并继续下一次用户的按键。

相关操作的主要实现如下：

移动：调用cursor的move函数

翻页：调用cursor的c_scroll函数

搜索与替换：调用text的search和replace函数

删除与插入：调用text相关的insert与delete函数，与此同时cursor控制光标的移动
具体细节可参考代码以及相关组员的实习报告。

算法框架：

带通配符的KMP，与朴素的KMP字符串模式匹配方式很类似，在get_next和kmp判断时，将？算作匹配，模式串和目标串同时后移一位；对于'*'，将字符串按照'*'切分，只有每一段都与模式串匹配且各自匹配的位置按顺序，才被认为匹配。对于带通配符的匹配，如果有重叠的匹配，则匹配最长的。

软件特色：

本软件的操作参考了vi，但和真正的vi有着很大的差距，然而相对于此次作业的一般要求，本小组的软件还是做了一定的改善。
移动方面，支持多种移动操作，如移动单词以及各按键的跳转。各种指定位置的跳转
插入方面，不仅限于简单的行插入，有类似于vi的进入插入模式的操作方案。
删除方面，支持多个字符以及多行删除。
搜索方面，支持/,?两种方式，以及nN两种移动方式，在搜索和替换过程中屏幕会有高亮显示的配合。
鼠标支持，支持鼠标点击定位，若无文本定位到改行最后一个字符。
交互优化，末行会显示当前光标位置，以及某些时候的相关提示。
关于操作具体手册尽请参考README。

主要分工：

1200012741 史杨勋惟（队长）：
设计了整个代码的框架以及各结构，并自行完成了screen.cpp，主要负责人机交互的接口完善以及整体的调试。
代码长度：5， 难度 5， 繁琐度 4.
1200012768 陈庆英：
实现了cursor类，完成了一些细节上的底层的移动以及翻页的处理，并协助调试了程序。
代码长度：5， 难度 3， 繁琐度 5
1200012826 姜双：
实现了text类中与内部文本相关的操作，如插入删除保存等。保证了文本内部运转的正确性。
代码长度：4， 难度 4， 繁琐度 4
1200012863 刘敏行：
主要负责搜索与替换层面上的算法设计，并协助完成了text类中与搜索替换相关的函数。
代码长度：3， 难度 5， 繁琐度 4

调试报告：

在本次作业完成过程中，遇到的主要问题有以下这些：

移动操作各边界条件，需要针对各个细枝末节进行很细致的分析，非常容易出错。

由于变量较多，很容写错变量名而导致出现异常情况。

文件流相关操作引起的文件流错误。

字符串替换在一行中多次替换容易出现的bug。

Ncurses对于输出窗口的支持并不是特别好，所以有些地方还需要自己打补丁。

使用到的调试方法是：

首先通过各种调试找到错误发生的具体位置，以及是哪个类的问题。

对于Text类的错误，可以直接在text.cpp中添加main函数直接对其进行调试，对于cursor或者screen的错误，由于和输出屏幕相关，直接调试不是特别方便，我们采用的方法是对于每一个类新增一个输出文件成员，并将自己的调试信息输出到该输出文件中，这样可以很清晰的观察程序的执行过程，方便差错。对于输出文件的处理，在main函数的最后通过调用系统的shell语句将其删除即可。

收获与体会：

这是我们合作的第一次大作业，关于这次大作业，我们对于我们的完成状况还是比较满意的。

在合作的过程中，我们掌握了更多关于软件开发的相关流程，包括整体设计流程，数据结构设计流程，算法设计流程，以及调试排错等等。尤其是在调试排错方面，我们相比于以前经历了更多的bug，也学会了更多的调试技巧。

我们对基本的线性数据结构以及字符串的相关操作以及KMP算法有了更深层面的掌握和认识。

同时我们也了解了Linux下的NCurses函数库以及熟悉了vi的一些基本操作。

我们感受到了彼此合作的力量，艰辛，与快乐。也感受到了作为一个程序员的苦与乐（好吧，其实是苦中作乐）

欢迎继续阅读各成员的个人实习报告～

参考资料：

Beginning Linux Programming (4th Edition)

Programmer's Guide to Ncurses

Ncurses Programming HowTo

算法参考：

HDU3901

陈庆英

我写的Cursor类的主要作用是协调光标在屏幕上的位置和在文本中的位置。

实现的功能具体有：

1） 光标左移一个字符，可以移到上一行的行尾；

2） 光标右移一个字符，可以移到下一行的行头；

3） 光标上移一个显示行，当上一行长度太短时，移到上一行的行尾；

4） 光标下移一个显示行，当下一行长度太短时，移到下一行的行尾；

5） 光标移到逻辑行的行头、行尾；

6） 向上翻半页或一页；

7） 向下翻半页或一页；

8） 光标移到文件的第一个字符；

9） 光标移到文件的最后一个字符；

10）光标移到鼠标点在屏幕上的任意位置，当目的行（显示行）太短时，移到目的行的行尾；当目的行不在文本范围内，移到最后一行。

实现的方法是：用四个变量分别表示：光标所在的显示行、显示列、逻辑行、逻辑列。光标的每次移到，都要更新这四个量的值，确保光标在屏幕上的位置和在文本中的位置能对应得上。

另外，该类实现了更新第一个显示的字符在文本中的位置。我把一个假想的光标固定放置在屏幕的左上角，根据真正光标的移动，更新假想光标在文本中的位置。出现以下情况时，假想光标在文本中的位置就要改变：

1） 真正光标在屏幕第一行，还要向上移一行；

2） 真正光标在屏幕最后一行，还要向下移一行；

3） 翻页；

4） 到当前逻辑行的行头或行尾（不一定）；

5） 到文件的头或尾（不一定）。

说实话，这部分没什么技术含量，但也着实废脑子。因为光标的每次移动，要判断的条件较多，比如当前光标是不是在第一个逻辑行、是不是在最后一个逻辑行，当前逻辑行的长度有没有超过屏幕的宽度，等等。倘若把”>=”写成”>”，就会出大问题。好在最初写的时候，我很认真地考虑了各种临界条件，没有出太多问题。只是有感于有时手一抖，把加号写出减号，耗费半天的时间与精力找问题出处、解决问题。

另外，我几乎对每个情况都把如何改变成员变量的值的一系列操作写了出来，没有使用太多的整合，结果使得代码很长。主要是当时觉得蛮简单的，没多想就写，之后也就不想改了。其实，没有整合也有其好处，那就是运行的效率会较高些。例如，要实现向上移动一个显示行，可以把具体的操作写出来（我的代码就是按这样写的），也可以通过左移来实现，前者的效率会更高些。

姜双：

本次数算实习的大作业文本编辑器已经编写完毕，我的收获很丰富。

在这次编写中，我和陈庆英、刘敏行、史杨勋惟合作完成。我主要负责编写和文本有关的Text类。我的任务是维护读入文本进行的基本操作。

Text类中主要成员变量有：

string类型变量content保存全部文本

string

类型变量buffer用于保存用户操作过程中即时变化的文本内容，如需保存修改则将buffer全部写入文件当中。

为了便于屏幕显示和插入、删除操作，我们将每个回车符视作一个逻辑行的结尾，并对每个逻辑行用一个string进行保存，存放在容器vector<string> rows中

特征变量还有文本的总逻辑行数lines， 总字数num_of_words，以及文件名FILENAME.

我编写的成员函数有：

构造函数Text()， 用于打开文本、复制文本到缓存， 将文本拆分成逻辑行保存等。

函数void save()， 用于将buffer中的所有内容写入文件，这个函数是最基本的文件流操作.

函数 void content_delete(int pos) 用于删除指定的一行pos中的所有内容。

函数void content_delete（int pos1, int pos2）删除pos1 和 pos2之间所有行的内容

函数 void content_insert (string s, int line, int pos)

用于在指定位置第line行第pos个字符处插入给定字符串s

函数 void content_insert_enter(int _line, int _pos)

插入一个回车，即插入新行，包括在给定行之前、给定行之后插入空行，以及在某行中间插入使之变成两行。

这是本次我编写的最复杂的函数了

函数 int one_line_length(int a)用于返回第a行的长度

除此之外的Text类其他函数由刘敏行编写

算法：

我负责的部分不涉及到复杂的算法，对文本的操作主要利用C++中String类的各种操作，就是负责维护整个文本记录的正常。同时为Cursor类、Screen类提供buffer内容、各个逻辑行的内容，

总代码量在210行左右。

这次实习中我负责的部分没有较难的算法挑战性，大多数是基本操作。这次实习在技能上主要的收获就是让我熟练掌握了文件流操作，文本的维护以及string类的多种使用。

最大的收获应该是锻炼了我的耐心。Text类中我的部分难度不高，但是需要很细心的完成，否则会有各种bug。比如每次更改文本后更新buffer，计算光标位置和文本的关系、逻辑行数目、插入回车等，稍有疏忽就会导致错误。通过这次实习我写代码的准确性提高了很多。

这对我以后完成各类作业、工程等保持严谨的作风和较高的准确度非常有帮助。

另外，这次团队合作也让我大致了解了最基本的合作项目完成的方式，我们几个人的默契提高了，看懂他人代码的能力也有了很大提升。

由于我的底子比较薄弱，外加今年的学生工作很多，所以这次我的任务总量比较少，这要感谢其他三位的较多贡献，我从这次的作业中学到了很多知识，弥补了之前的一些知识漏洞，对我在编程方面的影响很大。

最后感谢组长对整体框架的构建以及考虑到我学生工作较多给我很大的帮助，我会在今后的实习中努力学习更多的东西。

刘敏行：

在文本编辑器大作业中，我主要负责文本操作中的与字符串匹配相关的一些函数的编写，总的包括content_delete, update_highlights ,clear_highlights ,get_next , kmp, partition , content_search , replace 8个函数，其中最核心的是上课刚刚学的kmp函数，其他函数作为kmp函数的辅助。通过这些函数，我成功实现了如下功能：对于用户输入的带查找字串（可以带“*”、“？”两种通配符），对已有文本进行全文搜索，搜索结果高亮显示在屏幕上；之后，用户输入替换字符串，程序进行全文替换，替换结果高亮显示在屏幕上。接下来，我将阐述我是如何实现的。

首先通过replace函数接收用户输入的两个字符串，对于待替换字符串，将其作为参数调用content_search 函数，与全文匹配进行查找。这里与我们课上学习的字符串模式匹配方式很类似，只不过由于字符串可能含通配符，需要做改动——

对于‘？’，因为它匹配任何一个字符，在get_next和kmp判断时，将？算作匹配，模式串和目标串同时后移一位；对于‘*’，因为它匹配任意长度的字符串，求不出它的next值，于是必须将字符串按照‘*’切分，只有每一段都与模式串匹配且各自匹配的位置按顺序，才被认为匹配。因此在content_search中，再调用partition对字符串切分的同时用get_next得到所有串的next值，回到partition函数后对每一行文本通过kmp函数进行匹配，对于每一个匹配位置的信息（行号，起始位置，终止位置）用一个数组记录，全部匹配完成后通过clear_highlights 和 update_highlights 对匹配字符串高亮处理。返回总匹配数到replace函数后，通过content_delete 和 content_insert函数一删一插完成替换，再通过clear_highlights 和 update_highlights 对替换过的字符串高亮处理，从而全部任务完成！

说实话，我在之前小组内部分工之时，我是主动挑选了“基于kmp算法实现带通配符的字符串匹配相关功能”这一功能，主要是我觉得我在之前就比较细致地研究过kmp算法，支持通配符匹配应该只用做些小改动。但是实际做才发现远没有想象的那么简单，很容易想当然写出一些错误代码。比如我在处理问号时，在get_next中的判断条件一开始写成了if(j==-1 || s[i]==s[j] || s[j]=='?' || s[i]==’?’)，因为我觉得只要是‘？’都匹配就随手写上了，后面出问题了才发现后面一个条件不应该出现。还有不少细节问题要一个个攻克，例如输入字符全是‘*’，字符串头/尾是‘*’的情况，必须自己慢慢调试才会发现问题，做出改正。最后，我本来想的我的任务只是的单纯的算法任务，后面，为了将我的函数与其他组员写的类接合，自己还花了不少功夫在温习C++类的知识，倒也算是一举两得了。

现在想来，我作为小组一员参与完成了一个简单的文本编辑器，感觉还是蛮自豪的，毕竟在之前我对编写这样一个大的实用软件想都不敢想，现在跟组员分工合作奋斗两周完成任务，感觉也还好。这个大作业着实具有综合性，不仅帮助我们温习了刚刚在数据结构与算法课上学过的数据结构与算法，还把C++类、文件流等之前的知识串在了一起，进一步提高了我的综合编程能力。我很感谢其他组员们的给力支持，也很感谢这个大作业！

史杨勋惟：

这也许是我第一次自行设计并开发一个比较实用的程序，深感到作为一个程序员的艰辛。

这次大作业我是我们组的组长，所依进行了整个代码体系的设计，我简单的写了一些数据结构的头文件，并分配好了组员们的工作。当然，有些数据结构的细节以及一些在算法上的设计都是组员处理的，我只是设计了数据结构的框架而已。

screen是我自己实现的数据结构，主要用到了很多NCurses的函数，其主要作用即为处理用户与屏幕的互相交互，在screen的成员中包括了cursor和text，可以说是三个类中最为外层的一个类，直接提供给用户接口。screen类负责接受用户的按键，并将移动的信息传递给cursor类，将修改文本的信息传递给text类，再将text类和cursor类操作后返回的信息通过屏幕输出的方式（如光标的位置，以及一些提示信息等等）传递给用户。其中大量用到了switch语句，用于各种用户入的响应，也用到了一些按键的映射，用于简化代码。

Screen主要函数说明

get_key,get_act是对于用户输入的响应操作，是数据结构的核心

Init，exit_scr是初始化和退出函数

show_text为整个输出的核心函数，从cursor获取位置信息，从text获取文本信息，进行正确的屏幕输出

move_curse,s_scroll,move_curse_to与cursor相关，移动操作

go_insert_mode,go_delete_mode.save,go_search_mode,go_replace_mode,_inserting为与文本相关的操作，需要text类的支持

show_yx,print_invalid主要为了提供给用户更好的操作提示

对于我来说，这次大作业主要的收获在于了解了vi的基本操作，了解了Linux下的NCurses库，并掌握了一些关于程序开发的基本流程和操作。不过最大的收获应该是自己的调试程序的能力有了增强，当然也也认识到了还有需要提高的地方，比如整个代码体系的设计还不是特别的清晰，有些代码重用没有很好地避免，还比如有些地方设计的有些不够实用。

总体来说这次大作业对于我们组的完成情况还是非常满意的，虽说程序不是特别的完美，也有瑕疵，也有些许bug，不过我们四个也都不是大神，能够通过共同的合作实现出这样一个还有模有样的程序，已经算是一个很好的结果了吧～

当然，还是要感谢我们的团队每个人的付出，从刚开始设计到这次第一个版本的发布，每个人在辛苦的码代码和枯燥的调试中花费了很多时间。

希望这次大作业可以是一个开始，希望自己以后可以写出更好的程序。