The final project is a text editor that operates on LINES of text (no text wrapping from line to line), so it is like a PROGRAM-CODE editor. IT can be either an MFC or a WPF project, although I recommend using MFC.

Each line has a proper linend (CR+LF which is \r\n) as required on a Windows system, which does NOT display, nor is it accessible by the user.

See below for the requirements for commandline commands, suffix commands and other aspects of the program.

Here’s an EXAMPLE of what the final project COULD look like, depending on YOUR implementation and the actual requirements.

This EXAMPLE is based on a 1980’s PC program, called Kedit, which was based on the IBM program called Xedit (still in use and which you can visit on our TJW systems once I give you access).

It has the following rows at the top:

A command(menu) bar,

a toolbar with icons,

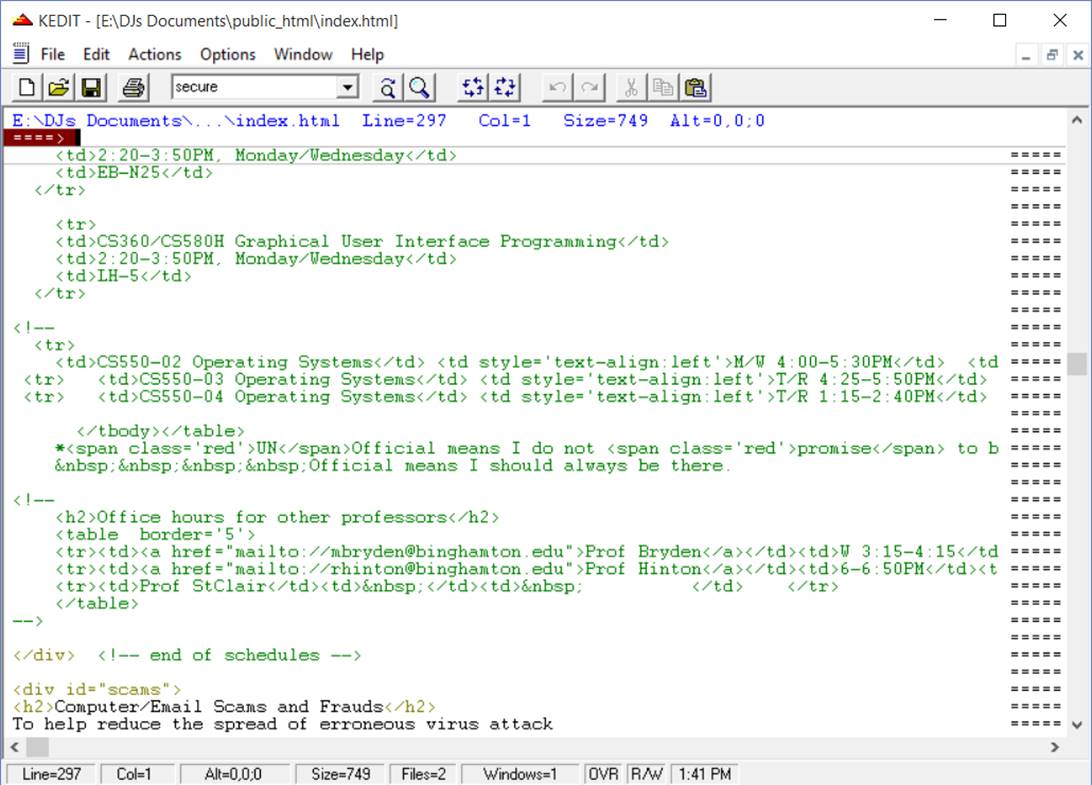
the name of the file being edited, the “current” line # (the line at the top of the view), # lines of the file and 3 #’s indicating what was last changed

a commandline (denoted by the ++++> on a red background)

file editing area

suffix column (denoted by the = = = = = rows)

There are more indicators at the bottom below the view area (I might not require these)



|  |  |
| --- | --- |
| Commandline/menu commands | explanation |
| Save | No parameters. Overwrites the original file. If no file exists, prompt for a path and name. |
| Save as | Saves the file to the name specified on the cmdline |
| Open | Opens the specified file reads it into memory, closes it unchanged, but ready for editing |
| Find | Allows the user to search for a string. See rules below these tables for how a user can specify the search. |
| n | Skip to line# of the file |
| Up n | Scroll up n lines |
| Down | Scroll down n lines |
| Left | Scroll left (the view shows more data on the left if any exists) |
| Right | Scrolls to the right (shows more data on the right if any exists) |
| Change | Modifies a searched-for string, starting at the current line  e.g.; c/abc/ABC/n1 n2  changes abc to ABC on n1 lines n2 times per line |
| Setcl n | Sets the number of the line on the screen which is to act as the current line until further notice |
| Help | Opens a window with a list of commands and their usage |

|  |  |
| --- | --- |
| Suffix commands | explanation |
| i | Insert |
| xn | Exclude n lines (default n=1) both from view and from actions – show a “hidden line” marker across the screen with its own suffix area for suffix commands. |
|  |  |
| sn | Show n lines (default n=1) of those excluded  (must be entered on the “hidden lines” indicator line) |
| a | Insert AFTER this line |
| b | Insert BEFORE this line |
| cn | Copy n lines (default n=1) starting at this line |
| mn | Move n lines (default n=1) starting at this line |
| “ | Duplicate this 1 line |
|  |  |

## Specifying a search

This must be implemented as both a command-line command and with a dialog box, so the user can do it either way.

Command-line example: find /mystring/#lines\_from\_current\_line starting\_column (if >1)

Dialog example:

find what: …..

replace with what:

#lines: (may be an \* meaning form the current line to the end of file

Starting column #:

Do not worry about finding a string with a / in it in the command-line version. This would require a user-defined escape character instead of the /.

## Replacing strings

Similar to a search, but with an extra string in it as follows:

Replace /mystring/newstring/lines column

In both the above if \* is used for column #, then the editor must check for the string EVERYWHERE on the lines to be searched. \* may also be used as the number of lines on which the replacement is to be done.