VI (Visual Editor) Cheat Sheet

Operators

Operators	Description
`d` ***operand***	delete the *operand* into the (delete) buffer
`p`	paste the contents of the (delete) buffer after the cursor
`y` ***operand***	yank the *operand* into the (delete) buffer
`i` ***operand***	inserts the *operand* (before current character)
`a` ***operand***	appends the *operand* (insert after current character)
`r` ***operand***	replaces current character with *operand*
`s` ***operand***	substitute the *operand* with typed-in text
`c` ***operand***	change the *operand* to typed-in text
`!` ***operand***	pass the *operand* to a (Unix) shell as standard input; standard output replaces the *operand*.

Common Macros	Description
`I`	insert at beginning of line (same as **^i**)
`A`	append at end of line (same as **$a**)
`D`	delete to end of line (same as **d$**)
`C`	change to end of line (same as **c$**)
`x`	delete one character (same as **dl**)
`ZZ`	save and exit
`:w` ***filename***	save as *filename* without exiting
`:q!`	quit immediately (without save)

Miscellaneous

`R`	enter replace (overstrike) mode
`o`	open line below current line
`O`	open line above current line
`"` ***n***	*n* is 0-9: delete buffers
`"` ***x***	*x* is lowercase a-z: replace user buffer
`"` ***x***	*x* is uppercase A-Z: append to user buffer
`.`	perform last change again
`u`	undo last change
`U`	undo all changes to current line

Operands	Description
`h j k l`	left, down, up, right; one character/line at a time
`w b e`	next **w**ord, **b**ack word, **e**nd of word

W B E	(same as above, but ignores punctuation)
/string	search for *string* (use **?** for reverse search)
n	search for *string* again (see **/**, above)
%	find matching **( )**, **{ }**, or **[ ]**
( )	beginning of current/previous sentence and beginning of next sentence
{ }	beginning of current/previous paragraph (two adjacent newlines) and beginning of next paragraph (see also **set paragraphs**)
[[ ]]	beginning of current/previous section and beginning of next section (mostly user-defined; see also **set sections**)
***line* G**	**g**oto particular line number (defaults to end-of-file)
0 ^ $	move to column 0, move to first non-whitespace, move to end of line
f *x*	**f**orward to character *x* on same line (inclusive)
t *x*	**t**o character *x* on same line (not inclusive)
;	last **f** or **t** again in the same direction
,	last **f** or **t** again in the opposite direction
m *x*	set mark *x* at current position
' *x*	move to line containing mark *x*
` *x*	move to exact position of mark *x*
''	move to line of last jump point
``	move to exact position of last jump point

Ex (colon-mode) commands

Range	**Description**
1,$	From line 1 to the end of the file.
10,20	From line 10 to line 20, inclusive.
.,.+10	From the current line to current line + 10 (11 lines total).
'a,'d	From the line containing mark *a* to the line containing mark *d*.
/from/,/to/	From the line containing "from" to the line containing "to", inclusive.

Commands which change the file being edited.

:e *filename*	Change from the current file being edited to *filename*. "**%**" means current file, and "**#**" means alternate file. Use **:e #** to edit the file most recently edited during the same session.
:n [*filename(s)*]	Edits the next file from the command line. With optional list of filenames, changes command parameters and edits the first file in the list. Filenames are passed to the shell for wildcard substitution. Also consider command substitution: **:n `grep -l pattern *.c`**
:args	Lists the files from the command line (possibly as modified by **:n**, above).

`:rew`	Restarts editing at the first filename from the command line.

Commands which modify the text buffer or disk file being edited.

`:g/`***RE***`/cmd`	Globally search for regular expression and execute **`cmd`** for each line containing the pattern.
`:s/`***RE***`/`***string***`/opt`	Search-and-replace; *string* is the replacement. Use **`opt`** to specify options **c** (confirm), **g** (globally on each line), and **p** (print after making change).
`:w` ***file***	Write the contents of the buffer to *file*. If *file* starts with an exclamation mark, the filename is interpreted as a shell command instead, and the buffer is piped into the command as stdin.
`:r` ***file***	Reads the contents of the file into the current buffer. If *file* starts with an exclamation mark, the filename is interpreted as a shell command instead, and the stdout of the command is read into the buffer.

These commands control the environment of the vi session.

`:set` ***opt***	Turns on boolean option *opt*.
`:set no`***opt***	Turns off boolean option *opt*.
`:set` ***opt=val***	Sets option *opt* to *val*.
`:set` ***opt?***	Queries the setting of option *opt*.

Miscellaneous commands.

`:abbr` ***string phrase***	Creates abbreviation *string* for the phrase *phrase*. Abbreviations are replaced immediately as soon as recognized during text or command input. Use **`:unab string`** to remove an abbreviation.
`:map` ***key string***	Creates a mapping from *key* to *string*. This is different from an abbreviation in two ways: abbreviations are recognized as complete units only (for example, a word with surrounding whitespace) while mappings are based strictly on keystrokes, and mappings can apply to function keys by using a pound-sign followed by the function key number, i.e. *#8* would map function key 8. If the terminal doesn't have an key, the mapping can be invoked by typing "#8" directly (doesn't work in the AIX 5L version of **vi**).

Resources:

1. https://bit.ly/3bVnQbF
2. https://bit.ly/3H7WT2X
3. https://bit.ly/3klV48G