The Vim Alphabetic Reference Card of Commonly Used Commands

Based on a list compiled by Charles E Campbell, Jr.,12/99. Updated by Alan E Klietz 5/2005.
Important/frequently used commands are boxed.

Command line options:

-b binary mode
-n no swap file; use memory only
-r recover aborted session via swap file
-R readonly mode
-V[N] verbose mode (to troubleshoot .vim scripts)

: enter Command Mode
Esc return to Normal Mode
. repeat the last command

% jump to matching parenthesis, brace, or bracket
'a jump to line with mark a. See m.
(jump to previous sentence
) jump to next sentence

---

0 goto beginning of line
$ goto end of line
+ goto first character of next line
, reverse direction of last f, F, t, or T
- goto first character of previous line
/<text> search forward for text, / to repeat
?<text> search backward text, ? to repeat
***** search for word under cursor N times
same as *, but search backwards
& synonym for ":s//~/" (repeat last substitution)

; repeat last f, F, t, or T
={motion} filter {motion} lines through 'equalprg' external program

---

@ execute command in register
[(goto previous unmatched '('
[[previous section
[] go backwards count sections or to previous '}' in 1st column
[{ goto previous unmatched '{'

[# goto previous unmatched "#if" or "#else"
[* goto previous start of a C comment (/*)
[i display 1st line that contains keyword under cursor
[I display all lines that contain keyword under cursor
[^I jump to 1st line that contains keyword under cursor

[d display 1st macro definition that contains macro under cursor
[D display all macro definition that contain macro under cursor
[^D jump to 1st macro definition that contains keyword under cursor
][go forwards count sections to to next '}' in 1st column
]] next section

]} goto next unmatched '}'
]# goto next unmatched "#if" or "#else"
]* goto next end of a C comment (*/)
]i like '[i', but start at current cursor position
]I like '[I', but start at current cursor position

---

^] jump to file having tag under cursor. See ^t

]^I like '[^I', but start at current cursor position
]d like '[d', but start at current cursor position
]D like '[D', but start at current cursor position
^ first non-whitespace character
_ move (count-1) lines downward on first non-blank character

` goto mark
{ previous paragraph
| to screen column [count] in current line
} next paragraph
~ switch case of current character

" go to start of line of previous mark or location before search
`` return to previous mark or location before a search
<return> next line
<spacebar> next character
!!{filter} filter [count] lines through the external program {filter}
!{cmd}{filter} send {motion} text through external program {filter}
!!scramble toggle encryption of C/C++ string (AEK)

<'a shift left up to mark
<% shift left until matching (, [, or {
<< shift line one shiftwidth to the left
>'a shift right up to mark
>% shift right until matching (, [, or {
>> shift line one shiftwidth to the right

---

a append after the current location
A append at the end of the line
^a add [count] to the number at or after the cursor
b beginning of previous word
B beginning of previous word, ignore punctuation
^b scroll back one screen; see ^f

---

c delete {motion} text (into register) and begin insert
C change to end of line
^c ends insert mode, unused in command mode
d{motion} delete text covered by a {motion}
d'a delete up to mark
Ndd delete N lines
D delete to end of line
^d scroll down half a window (N^d sets new default); see ^u

---

e end of word
E end of word, ignore punctuation
^e scroll screen down one line

---

fx find given character forwards
F find given character backwards
^f scroll forward one screen; see ^b

---

g ^g show information about current cursor position
g ^h start Select block mode
g ^] :tjump to tag under the cursor
g# like "#", but without using "\<" and "\>"
g$ wrap off: goto rightmost onscreen character of current line
g* like "*", but without using "\<" and "\>"
g0 wrap off: goto leftmost onscreen character of current line
g? rot13 encoding operator

---

g?? rot13 encode current line
g?g? rot13 encode current line
gD goto definition of word under cursor in current file
gE go backwards to end of previous WORD
gH start Select line mode

---

gI like "I", but always start in column 1
gJ join lines without inserting space (like :j!)
["x]gP put text (from register x) N times
gR enter virtual replace mode
gU{motion} make Nmove text uppercase

---

gV don't reselect previous Visual area (maps, menus) in Select mode
g] :tselect on tag under cursor
g^ wrap off: goto leftmost non-white onscreen char on current line
ga print ascii value of character under cursor
gd goto definition of word under cursor in current function

---

ge go backwards to end of previous word
gf start editing file whose name is under cursor
gg cursor to line N (default: 1) ; start Select mode
gh start Select mode
gj wrap on: like "j", but go N screen lines down

---

gk wrap on: like "k", but go N screen lines up
gm goto character at middle of screenline
go cursor to byte N in buffer
["x]gp put text (from register x) after cursor N times
gq{motion} format text
gr{char} virtual replace N chars with given char
gs goto sleep for N (1) seconds
gu{motion} make Nmove text lowercase
gv reselect previous Visual area
g~{motion} swap case for Nmove text

---

g<Down> same as gj
g<End> same as g$
g<Home> same as g0
g<LeftMouse> same as <ctrl-LeftMouse>
g<RightMouse> same as <ctrl-RightMouse>
g<Up> same as gk

G goto line (100G); defaults to EOF
^g show status line
h left
H goto first line on screen
^h backspace in insert mode, left in command mode

---

i insert before current location (until Esc)
I insert before first non-whitespace character on line
^I goto [count] cursor position in jump list

---

j down
J join next line with current line
^j down in command, create newline in insert

---

k up
K use keywordprg (kp option) to lookup keyword under cursor
^k xx enter digraph specified by two characters

---

l right
L goto last line on screen
^l redraw screen

---

m mark position into register (ma)
M goto middle of screen
^m carriage return

---

n repeat last search
N repeat last search, reverse direction
^n down in command mode,
^n show the next completion in insert mode

---

o open line below current
O open line above current
^o goto older cursor position in jump list

---

p put below current line
P put above current line; see y. After Ndd, puts deleted text.
^p up in command mode
^p show the prev completion in insert mode

---

qx record typed characters into register x
q stop macro recording
Q quit and run ex
^q same as ctrl-v

---

r replace current character
R replace characters until Esc
^r redraw screen in command mode

---

s substitute char under cursor
S substitute entire line
^s split current window into two (AEK)

---

t to...
T backward to...
^t jump back to referer from tag def (see ^[ )
^t move to next shiftwidth in insert mode

---

u undo last change (toggle)
U undo changes to current line
^u scroll up half a window (N^u sets default)
v start Visual mode at given character
V start linewise Visual mode
^v start blockwise Visual mode (in insert mode begins insertion of a literal char)

---

w beginning of next word
W beginning of next word, ignore punctuation

---

^W ^] same as "^W]"
^W ^^ same as "^W ^"
^W ^_ same as "^W _"
^W + increase current window height N lines
^W - decrease current window height N lines
^W = make all windows the same height
^W] split window and jump to tag under cursor
^W ^ split current window and edit alternate file N
^W _ set current window height to N (default: very high)

---

^W ^B go to bottom window
^W ^C close current window (like |:close|)
^W ^D split window and jump to definition under the cursor
^W ^F split window and edit file name under the cursor
^W g ^] split window and do |:tjump| to tag under cursor
^W g] split window and do |:tselect| for tag under cursor
^W g } do a |:ptjump| to the tag under the cursor
^W ^I split window & jump to declaration of identifier under cursor
^W ^J go to N next window (stop at last window)
^W ^K go to N previous window (stop at first window)
^W ^N open new window, N lines high
^W ^O close all but current window (|:only|)
^W ^P go to previous window
^W ^Q quit current window (like |:quit|)
^W ^R rotate windows downwards N times
^W ^S split current window in two parts, new window N lines high (|:split|)
^W ^T go to top window
^W ^W go to N next window (wrap around)
^W ^X exchange current window with window N (default: next window)
^W ^Z close preview window
^W } show tag under cursor in preview window

---

^W <Down> same as "^W j"
^W <Up> same as "^W k"

---

x delete current character
X delete previous character
^x^n append next word to completion (see ^n)
^x^p ditto. Typical: longFunc^p^x^p^x^p^x^p

---

y yank (:1,'ay); see P
Y yank N lines (10Y); see P
^y scroll screen up one line

---

z redraw with line [count] at top-of-window
zc close fold
zd delete fold
zo open fold
zL scroll screen half a screenwidth to left
zH scroll screen half a screenwidth to right
zs scroll text&cursor horizontally so cursor on right of screen
ze scroll text&cursor horizontally so cursor on left of screen
ZZ write (only if changes have been made) and quit

---

z{height}<CR> redraw window {height} lines tall
z<Right> wrap off: scroll screen [count] characters to left
z<Left> scroll screen [count] characters to right

:s/old/new/ substitute text on current line
s/line1/line1^V^Mline2/ for newline
:s/old/new/<flags> apply special flags:
c confirm each subst with y/n
g subst all occurrences on same line
:1,Ns/old/new/ subst on lines 1-N
:.,'as/old/new/ subst from current line to mark a
:%s/old/new/ subst on all lines
:g/text/d delete all lines containing text
:v/text/d delete all lines not containing text

:cd <path> change dir (cd – to pop back)
:e <file> edit file. e! to force
:e ++enc=ucs-2le <file> Load as Unicode
:e ++ff=unix <file> Load EOLs as \n; also =dos
:.,.+10fold fold 10 lines at cursor. See zc, zd, zo
:f <name> rename file
:hardcopy send file to printer
:X encrypt file contents
:w <file> write file. :w! to force
:w ++enc=utf-8 <file> Write out as UTF-8
:w ++enc=ucs-2le <file> Write out as Unicode
:w ++ff=unix <file> Convert \r\n to \n; also =dos
:q quit. q! to force

:set [all] show modified options (or all)
:set [no]autoindent automatically indent each line
:set [no]expandtab convert inserted ^I to spaces
:set fileencodings=ucs-2le
show next file as Windows Unicode (use :e!)
:set [no]ignorecase ignore case in searches
:set [no]list show tabs as ^I and EOL with $
:set shiftwidth=n set shiftwidth for >>
:set tabstop=n set tabstop for ^I
:set textwidth=n set textwidth for autowrap

Clipboard:
^C copy highlighted text to clipboard
^X cut highlighted text to clipboard (deletes)
^Q paste text from clipboard (AEK mod)
(not ^V – toggles Visual Block Mode)

Notes:

1. ^ before a letter implies the Ctrl key
2. All commands apply to Normal Mode unless otherwise indicated. Many can be invoked in Insert Mode as well by typing ^O followed by the desired keystroke(s). Example: ^OFZ searches backward for Z.
3. See ":help motion" for a description of motion keystrokes for {motion}