**1)What is VI/VIM?**

**VI** is a screen-oriented [text editor](https://en.wikipedia.org/wiki/Text_editor) originally created for the [Unix](https://en.wikipedia.org/wiki/Unix) operating system. The portable subset of the behavior of vi and programs based on it, and the [ex](https://en.wikipedia.org/wiki/Ex_(text_editor)" \o "Ex (text editor)) editor language supported within these programs, is described by (and thus standardized by) the [Single Unix Specification](https://en.wikipedia.org/wiki/Single_Unix_Specification) and [POSIX](https://en.wikipedia.org/wiki/POSIX).

The original code for vi was written by [Bill Joy](https://en.wikipedia.org/wiki/Bill_Joy) in 1976, as the visual mode for a [line editor](https://en.wikipedia.org/wiki/Line_editor) called ex that Joy had written with Chuck Haley. Bill Joy's ex 1.1 was released as part of the first [BSD](https://en.wikipedia.org/wiki/Berkeley_Software_Distribution) [Unix](https://en.wikipedia.org/wiki/Unix) release in March, 1978. It was not until version 2.0 of ex, released as part of Second Berkeley Software Distribution in May, 1979 that the editor was installed under the name vi (which took users straight into ex's visual mode), and the name by which it is known today. Some current implementations of vi can trace their source code ancestry to Bill Joy; others are completely new, largely compatible reimplementations.

VIM is a [text editor](https://en.wikipedia.org/wiki/Text_editor) written by [Bram Moolenaar](https://en.wikipedia.org/wiki/Bram_Moolenaar) and first released publicly in 1991. Based on the ideas of the [vi](https://en.wikipedia.org/wiki/Vi) editor common to [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems, Vim is designed for use both from a [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface) and as a standalone application in a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface). Vim is [free and open source software](https://en.wikipedia.org/wiki/Free_and_open_source_software)

**2)Difference between VI and VIM**

Some improvements in vim:

* [completion](https://en.wikipedia.org/wiki/Autocomplete)
* [comparison](https://en.wikipedia.org/wiki/Data_comparison) and [merging](https://en.wikipedia.org/wiki/Merge_(revision_control)) of files (known as vimdiff)
* a comprehensive integrated help system
* extended [regular expressions](https://en.wikipedia.org/wiki/Regular_expression)
* [scripting languages](https://en.wikipedia.org/wiki/Scripting_language) (both native and through alternative scripting interpreters such as Perl, Python, Ruby, Tcl, etc.) including support for [plugins](https://en.wikipedia.org/wiki/Plug-in_(computing))
* a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (known as gvim), limited [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment)-like features, [mouse](https://en.wikipedia.org/wiki/Mouse_(computing)) interaction (both with and without the GUI)
* [folding](https://en.wikipedia.org/wiki/Folding_editor)
* editing of compressed or archived files in[gzip](https://en.wikipedia.org/wiki/Gzip), [bzip2](https://en.wikipedia.org/wiki/Bzip2), [zip](https://en.wikipedia.org/wiki/ZIP_(file_format)), and [tar](https://en.wikipedia.org/wiki/Tar_(file_format)) format and files over network protocols such as [SSH](https://en.wikipedia.org/wiki/Secure_Shell), [FTP](https://en.wikipedia.org/wiki/File_Transfer_Protocol), and [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol" \o "Hypertext Transfer Protocol)
* session state preservation
* [spell checking](https://en.wikipedia.org/wiki/Spell_checker)
* split (horizontal and vertical) and tabbed windows
* [unicode](https://en.wikipedia.org/wiki/Unicode) and other multi-language support
* [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting)
* trans-session command
* search and cursor position [histories](https://en.wikipedia.org/wiki/Command_history)
* multiple level and branching [undo/redo](https://en.wikipedia.org/wiki/Undo) history which can persist across editing sessions
* visual mode

**3)VI/VIM modes**

*Normal or Escape mode* - main mode in VI/VIM. You can move through the file and execute some commands, like copy, paste, delete etc. Press ESC-key to return to it from any other mode.

*Insert mode* - main mode to edit text. Press *Insert* once to enter it. Press *Insert* twice to enter to the Replace mode.

Search mode - is used for searching something in file. / - forward search. ? - backward search. If vim reaches end of file, it begin search from the first line of file(in case of forward search). For backward search - vice versa.

Command - is used for executing different commands. Press : - to enter commands mode, and then enter command to execute and press Enter.

Visual - this mode allows you to select multiple characters, lines, blocks in text and execute some commands.

**4)VI/VIM commands**

*Main “rules”*

<number> <command> - repeat <command> <number> times. F.e. 5dd - delete 5 lines.

<ESC> - return to normal mode

you can combine commands to get what you need).

*Movement*

h - left

j - down

k - up

l - right

0 - begin of current line

^ - begin of current line and first non-space character

$ - end of current line

e - end of word

w - word right

b - word left

W - to the space character right

B - to the space character left

Ctrl + f - page down

Ctrl + b - page up

{ - paragraph up

} - paragrapg - down

gg - begin of file

G - end of file

‘. - jump to last modified line

`. - jump to the exact spot in last modified line

Ctrl + O - retrace your movements in file backward

Ctrl + I - retrace your movements in file forward

<number> gg - jump to exact line

gf - open file(if under cursor is path)

*Open, create, save, close files*

vim <filename> - open file

vim <filename> ++<number> - open file and place cursor at line <number>

vim <file1> <file2> … - open files, after closing the first file, edit the next one

vim -o <file1> <file2> … - open files immediately. they will be opened in spited windows

:e <filename> - open new file

:new <filename> - open new file

:w - save file

:w! - save file anyway. Even if file is read-only.

:w <filename> - save file as …

:wn - save current file and open next

:wa - save all opened files

:q - close editor

:q! - close editor without saving file

:qa! - close editor without saving files

:wq - save file and close editor

:x - save file and close editor

:sav <filename> - save as …

:sp <filename> - open file in split window

ZZ - save file and close editor

ZQ - close editor without saving

:args - shows list of vim arguments

:n - open next file in argument list

:prev - open previous file in argument list

:rew - open first file in argument list

*File editing*

i - insert mode from current position

I - insert mode from begin of current line

r - replace character under cursor

R - replace mode from current position

a - insert mode from position next to the cursor

A - insert mode at the end of line

o - new line under current one and insert

O - new line above current one and insert

x - delete character under cursor

X - delete character before cursor

d + <movement command> - delete

f.e.

dw - delete word from current position till the end of word

diw - delete word under cursor

dd - delete line

d<number>d - delete <number> lines from current one

c + <movement commands> - delete and after that insert

C or c$ - clear current line and insert

ce - delete current word and insert

yy - copy current line

y<number>y - copy <number> lines from current one

p - paste under cursor

P - paste above cursor

J - unite current line and next one

*Search*

/<pattern> - search forward

?<pattern> - search backward

n - next match

N - previous match

\* - next match of word under cursor

# - previous match of word under cursor

:%s/<pattern\_to\_search>/<pattern\_to\_replace>/gc - replace all matches of <pattern\_to\_search> with <pattern\_to\_replace>, highlight result and ask before replace

:<begin>,<end>s/<pattern\_to\_search>/<pattern\_to\_replace>/gc - replace all matches of <pattern\_to\_search> with <pattern\_to\_replace> between lines <begin> and <end>, highlight result and ask before replace

:g/one\|two/ - list of all lines that contains ‘one’ or ‘two’

*Visual*

v - select by character

Ctrl + v - select by line

Ctrl + V - select by rectangle

y - copy selected

p - paste selected

d - delete selected

gu - change selected to lowercase

gU - change selected to uppercase

*Other commands*

u - undo last change. not limited

U - undo changes

Ctrl + r - redo

:redo - redo

. - previous command

:set number - show line numbers

:set nonumber - hide line numbers

:set wrap - allow to wrap lines if their length is more than window width

:set nowrap - disallow to wrap lines if their length is more than window width

:sy[ntax] on/off - turn on/off syntax highlight

:h - help

*Tabs, buffers, windows*

:tabnew <name> - open new named tab

:tabs - tabs lis

:tabn or gt - next tab

:tabp or gT - previous tab

:tablast - last tab

:tabfirst - first tab

:tabdo <command> - execute command for all tabs

:Ex - open file explorer

:ls - open list of buffers

:cd .. - move to parent directory

:bn - next buffer

:bp - previous buffer

*Tips and tricks*

vimdiff

:vimgrep <pattern> <path> - search for pattern in path and put result in buffer

:vimgrep <pattern> <path> | :copen - search for pattern in path and put result in buffer and open that buffer

q: - command history

guu - change all characters in current line to lowercase

gUU - change all characters in current line to uppercase

g~~ - change case of characters in current line(upper -> lower; lower -> upper)

ggguG - change case of characters in all file to lowercase

gggUG - change case of characters in all file to uppercase

:ju - list of all movements

:history - list of all your commands

!!date - replace current line with current date

:w <some path>/%:r - save file in another directory but with save name

:!<linux command> - replace current line with output of linux command

:1,10 w <outfile> - write lines from 1 to 10 in <outfile >

:1,10 w >> <outfile> - append lines 1 to 10 in <output>

:r <infile> - insert <infile> in current position

:23r <infile> - insert <infile> in line 23

:g/^\s\*$/d - delete all blank lines

:g/<pattern>/d - delete lines that contain <pattern>

:v/<pattern>/d - delete lines that do NOT contain <pattern>

:%s/^/hello/g - add ‘hello ’ in the beginning of all lines

:$s/$/ bye bye/g - add ‘ bye bye’ in the end of all lines

:split <filename> - open file and put it in current window, split window horizontal

:vsplit <filename> - open file and put it in current window, split window vertical

markers in vim

ma - create marker ‘a’

`a - move to marker ‘a’

d’a - delete all from current position to marker ‘a’

:marks - list of all merkers