ITE315 Module 1 Part B - Text Editors: vi(m)

Athens State University

Contents

1 The Tools of the Trade

The Basic Tools

Every craftsman starts his journey with a basic set of good quality tools

You Start With Raw Materials

- We work with knowledge
  - Gather requirements as knowledge about a problem
  - Express that knowledge as
    - Design
    - Implementation
    - Tests
    - Documents

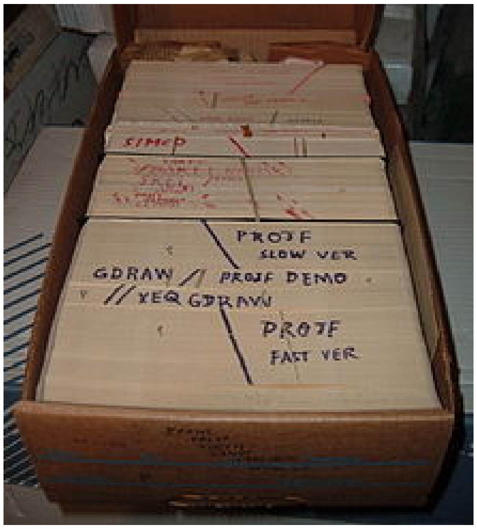
Plain Text

Plain text is made up of printable characters in a form that can be read and understood directly by people

- Note: *understood by people*
- Plain text ! = unstructured
- Binary formats don't carry context - metadata

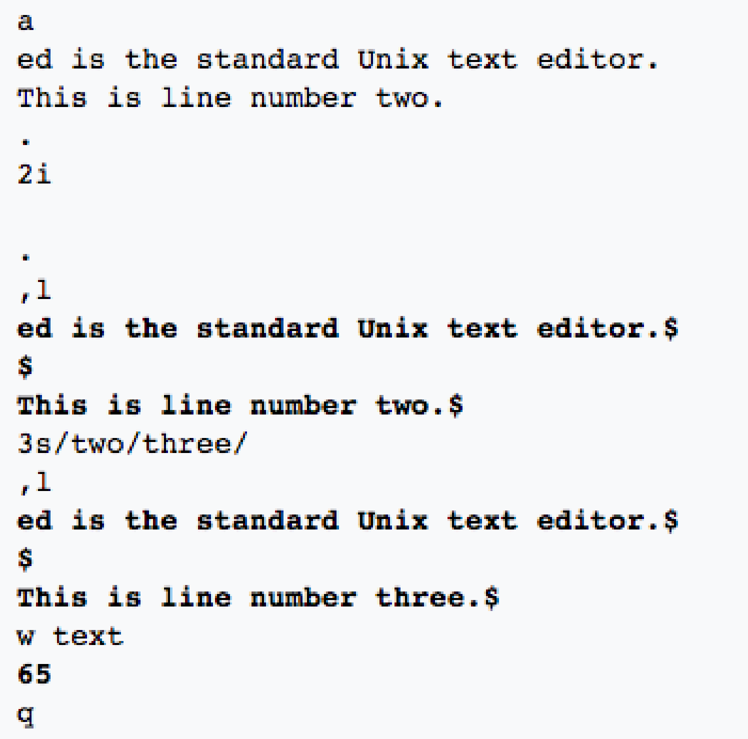
- Benefits
  - Insures against obsolescence
  - Leverage (just about everybody can read text)
  - Lowest common denominator

A Little History



- **Line editors**: text editor where each editing command applies to one more complete lines of text specified by the user
- Think of having to use a modern editor with the document being out of view

The `ed` Line Editor

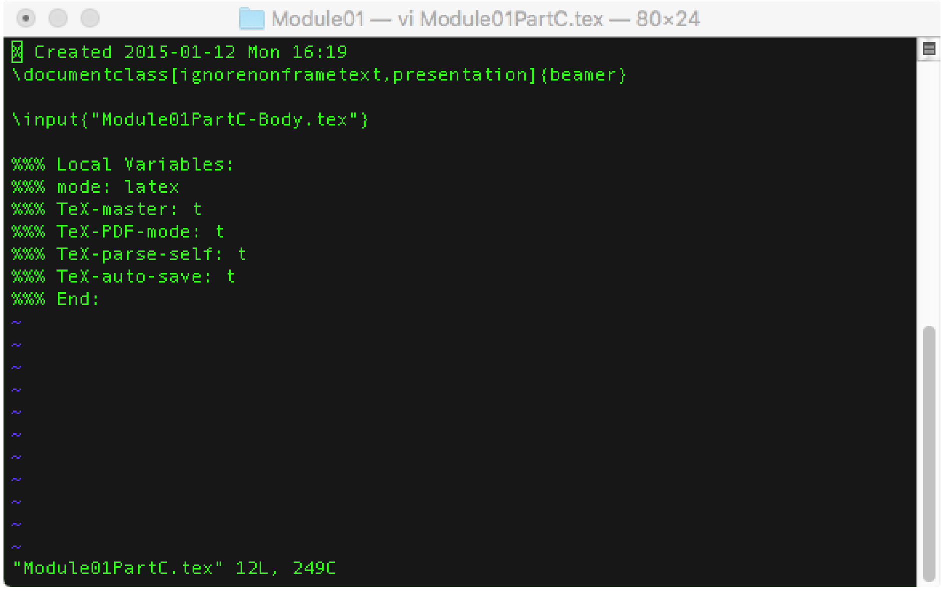


```
a
ed is the standard Unix text editor.
This is line number two.
.
2i

.
,l
ed is the standard Unix text editor.$
$
This is line number two.$
3s/two/three/
,l
ed is the standard Unix text editor.$
$
This is line number three.$
w text
65
q
```

- The very first version of UNIX (1969) included the `ed` line editor
- Still available on modern UNIX-like OSes such as Linux and macOS
- Inspiration for the `ex`, and `sed`, and `vi` editors
- Still used (in the `sed` variant) in shell scripts as part of instruction pipelines

The Original `vi` Editor



```
% Created 2015-01-12 Mon 16:19
\documentclass[ignorenonframetext,presentation]{beamer}

\input{"Module01PartC-Body.tex"}

%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% TeX-PDF-mode: t
%%% TeX-parse-self: t
%%% TeX-auto-save: t
%%% End:
~
~
~
~
~
~
~
~
~
~
~
"Module01PartC.tex" 12L, 249C
```

- `vi` was the first screen editor for UNIX, designed for a time when the primary user interface revolved around green screen terminals
- Is an evolution of the `ex` line editor, which was an evolution of the `ed` text editor

But Why Learn To Use It?

Dr. Lewis's Law Of Text Editors: Pick one, become a ninja on how use that editor, but learn enough `vi` to where you can navigate, save a file, and exit

- One can find *vi* or it's more recent clone, `vim` on just about any Unix system in the world
- Often the default editor on many systems (although `pico`) is getting popular)

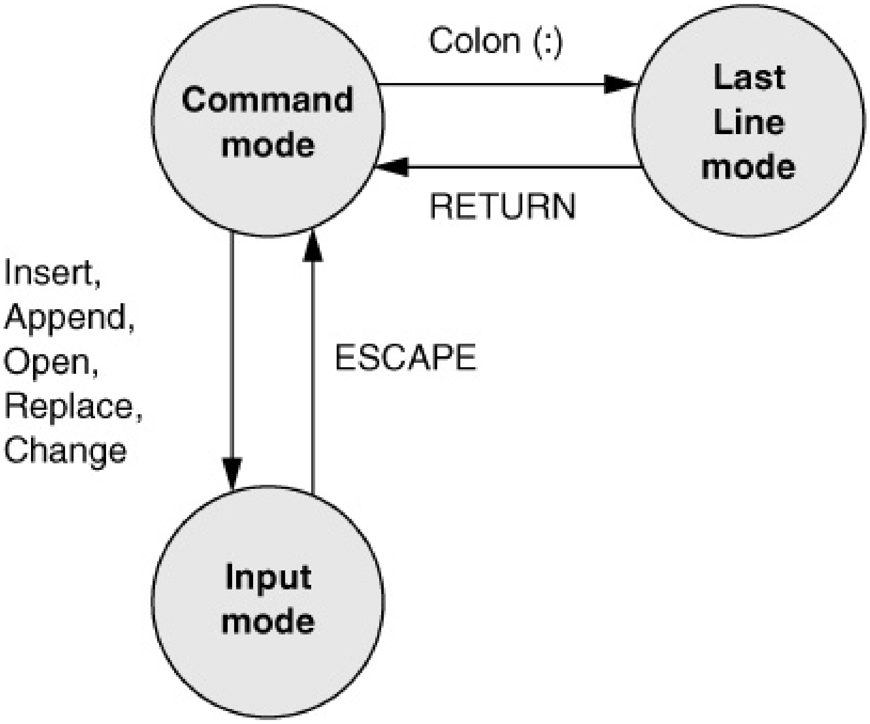
As we said at the start of the lecture, we spend our entire life with computers manipulating text. So, it's real important that you intimately understand the tools we use to manipulate text. This is why I tell novice programmers to pick one programming editor that will be your standard tool. This maybe a classic like `vi` or `Emacs`, one of the desktop computer oriented tools such as `Notepad++`, `Sublime Text`, or `BBedit`, or the editor in an IDE such as Visual Studio, Xcode, or Eclipse. But the ubiquity of `vi` in UNIX-like environments means that you need to at least know the basic command-set in that editor.

2 Using VIM

The VI iMproved Editor

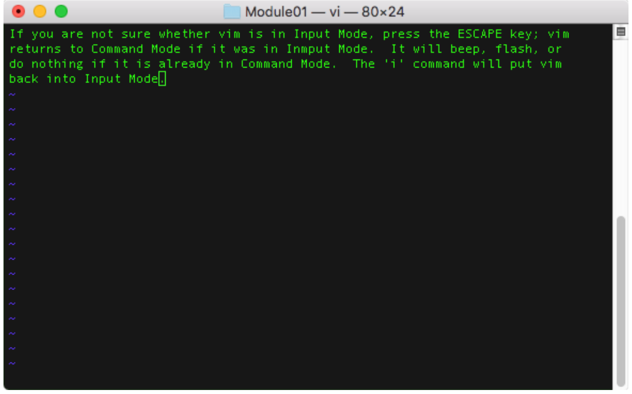
- Modern Linux distributions use the VI iMproved (`vim`) clone of `vi`
- `vim` was developed as fork for the Amiga of the Atari ST's `STEVIE` vi clone
- Was ported to Linux early in the history of Linux
- Includes classic `vi` features plus a GUI version, macro languages, and a plug-in model

`vim` Is A Modal Editor



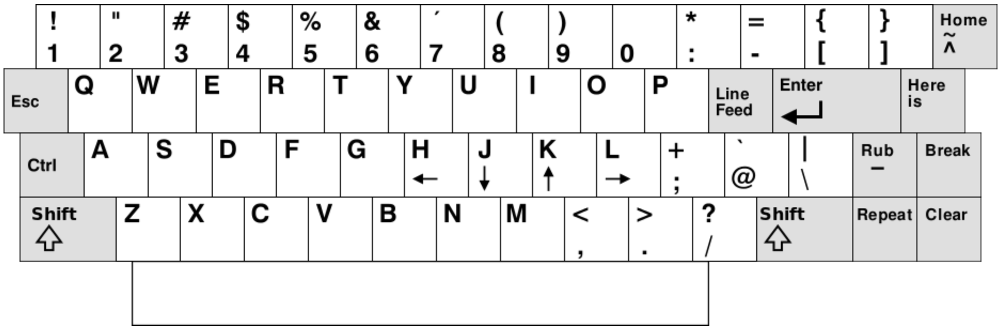
- **Command mode**: Issue commands to the editor
- **Input Mode**: Insert, append, open, replace, and change text
- **Last Line Mode**: Enter `ex` commands

`vim` Is A Modal Editor

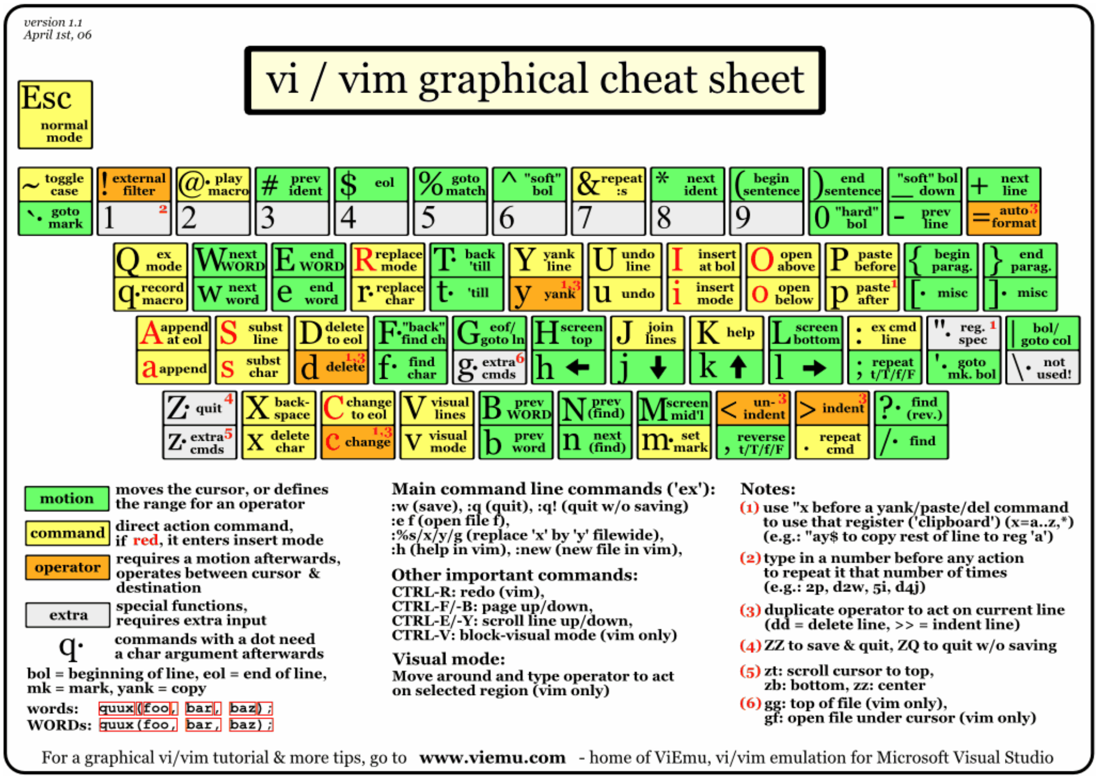


```
If you are not sure whether vim is in Input Mode, press the ESCAPE key; vim
returns to Command Mode if it was in Inmput Mode.  It will beep, flash, or
do nothing if it is already in Command Mode.  The 'i' command will put vim
back into Input Mode.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

vim **and the Keyboard**



The vim Command Set



motion: moves the cursor, or defines the range for an operator

command: direct action command, if red, it enters insert mode

operator: requires a motion afterwards, operates between cursor & destination

extra: special functions, requires extra input

q· commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: quux(foo, bar, baz);
WORDs: quux(foo, bar, baz);

Main command line commands ('ex'):
:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:
CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:
Move around and type operator to act on selected region (vim only)

Notes:

(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')

(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)

(3) duplicate operator to act on current line (dd = delete line, >> = indent line)

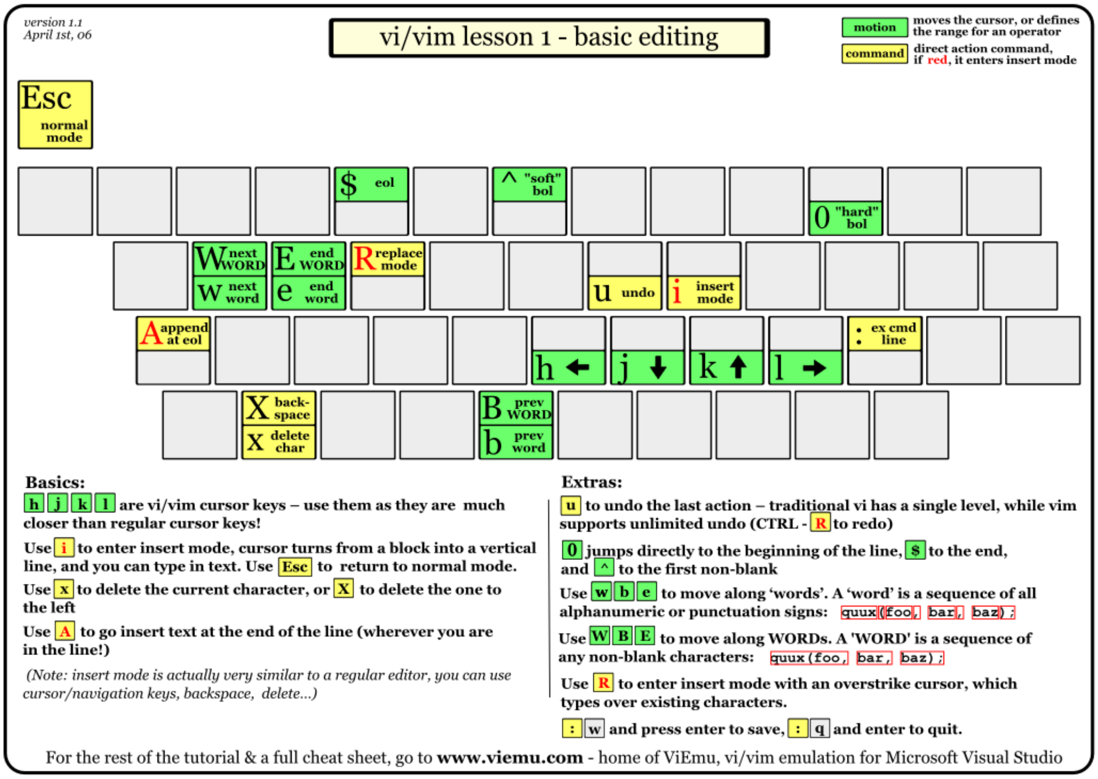
(4) ZZ to save & quit, ZQ to quit w/o saving

(5) zt: scroll cursor to top, zb: bottom, zz: center

(6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

The vim Command Set



Basics:

h j k l are vi/vim cursor keys – use them as they are much closer than regular cursor keys!

Use i to enter insert mode, cursor turns from a block into a vertical line, and you can type in text. Use Esc to return to normal mode.

Use x to delete the current character, or X to delete the one to the left

Use A to go insert text at the end of the line (wherever you are in the line!)

(Note: insert mode is actually very similar to a regular editor, you can use cursor/navigation keys, backspace, delete...)

Extras:

u to undo the last action – traditional vi has a single level, while vim supports unlimited undo (CTRL - R to redo)

0 jumps directly to the beginning of the line, $ to the end, and ^ to the first non-blank

Use w b e to move along 'words'. A 'word' is a sequence of all alphanumeric or punctuation signs: `quux(foo, bar, baz);`

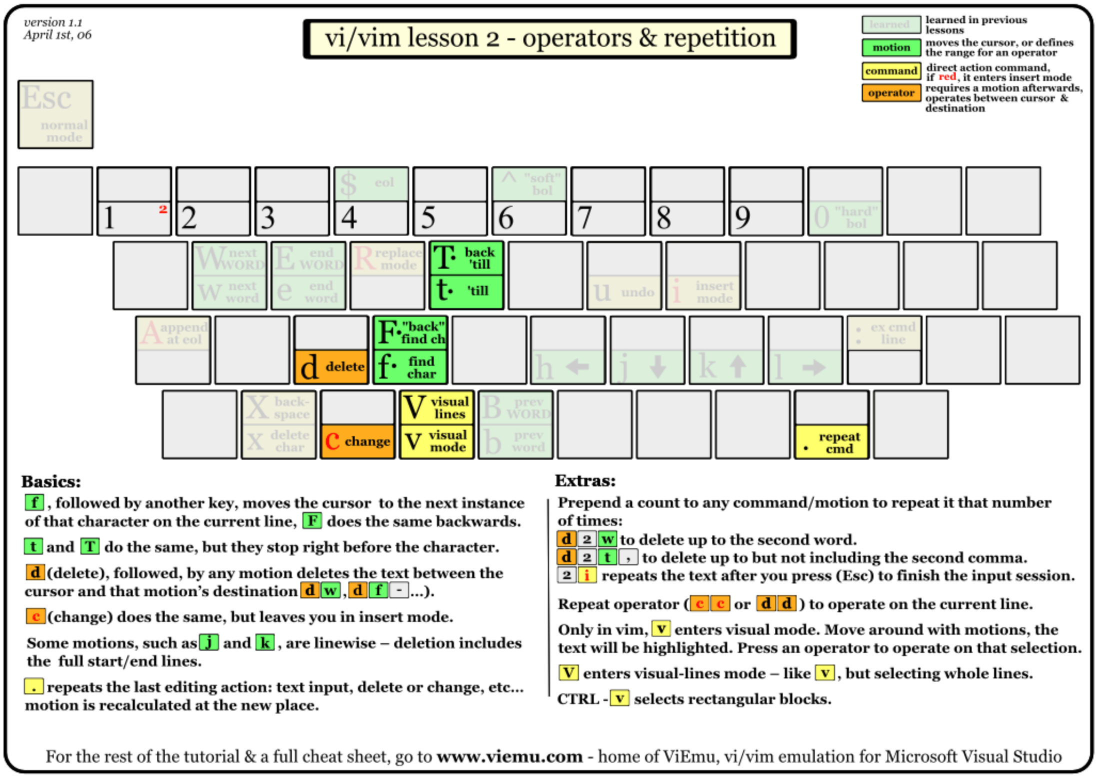
Use W B E to move along WORDs. A 'WORD' is a sequence of any non-blank characters: `quux(foo, bar, baz);`

Use R to enter insert mode with an overstrike cursor, which types over existing characters.

: w and press enter to save, : q and enter to quit.

For the rest of the tutorial & a full cheat sheet, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

The vim Command Set



Basics:

f, followed by another key, moves the cursor to the next instance of that character on the current line, F does the same backwards.

t and T do the same, but they stop right before the character.

d (delete), followed, by any motion deletes the text between the cursor and that motion's destination d w, d f - ...).

c (change) does the same, but leaves you in insert mode.

Some motions, such as j and k, are linewise – deletion includes the full start/end lines.

. repeats the last editing action: text input, delete or change, etc... motion is recalculated at the new place.

Extras:

Prepend a count to any command/motion to repeat it that number of times:

d 2 w to delete up to the second word.
d 2 t , to delete up to but not including the second comma.
2 i repeats the text after you press (Esc) to finish the input session.

Repeat operator (c c or d d) to operate on the current line.

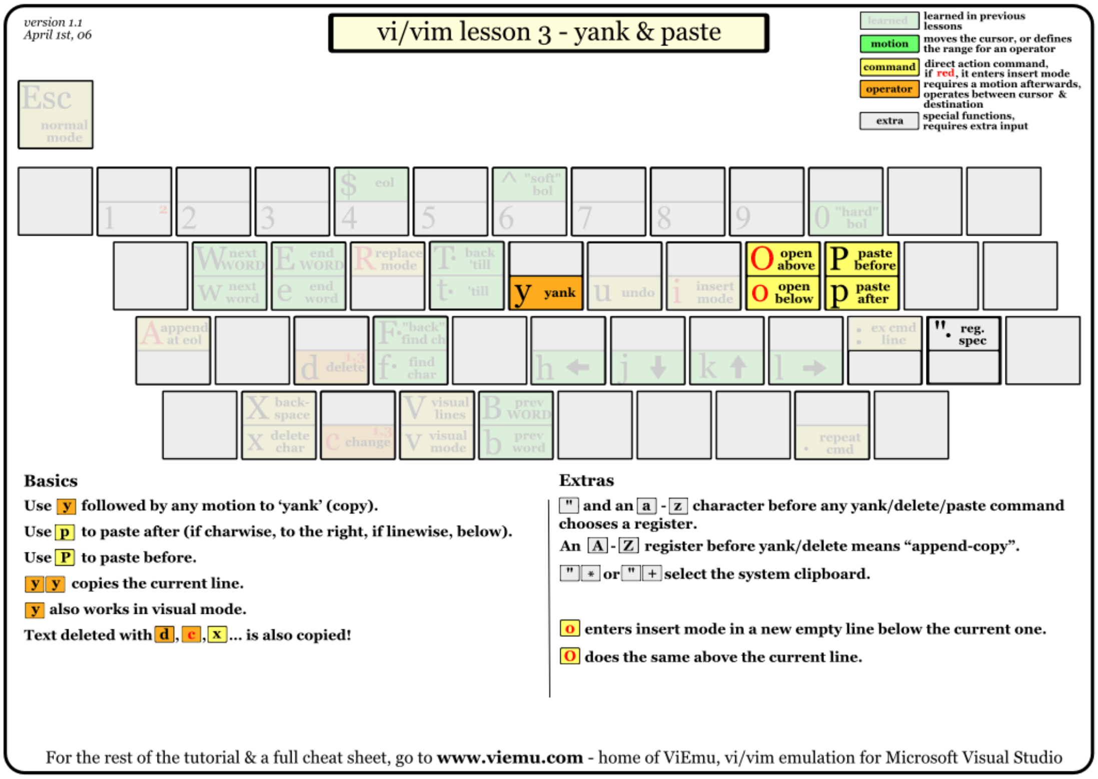
Only in vim, v enters visual mode. Move around with motions, the text will be highlighted. Press an operator to operate on that selection.

V enters visual-lines mode – like v, but selecting whole lines.

CTRL - v selects rectangular blocks.

For the rest of the tutorial & a full cheat sheet, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

The vim Command Set



version 1.1
April 1st, 06

vi/vim lesson 3 - yank & paste

learned	learned in previous lessons
motion	moves the cursor, or defines the range for an operator
command	direct action command, if red, it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input

Basics

Use y followed by any motion to 'yank' (copy).

Use p to paste after (if charwise, to the right, if linewise, below).

Use P to paste before.

y y copies the current line.

y also works in visual mode.

Text deleted with d, c, x... is also copied!

Extras

" and an a - z character before any yank/delete/paste command chooses a register.

An A - Z register before yank/delete means "append-copy".

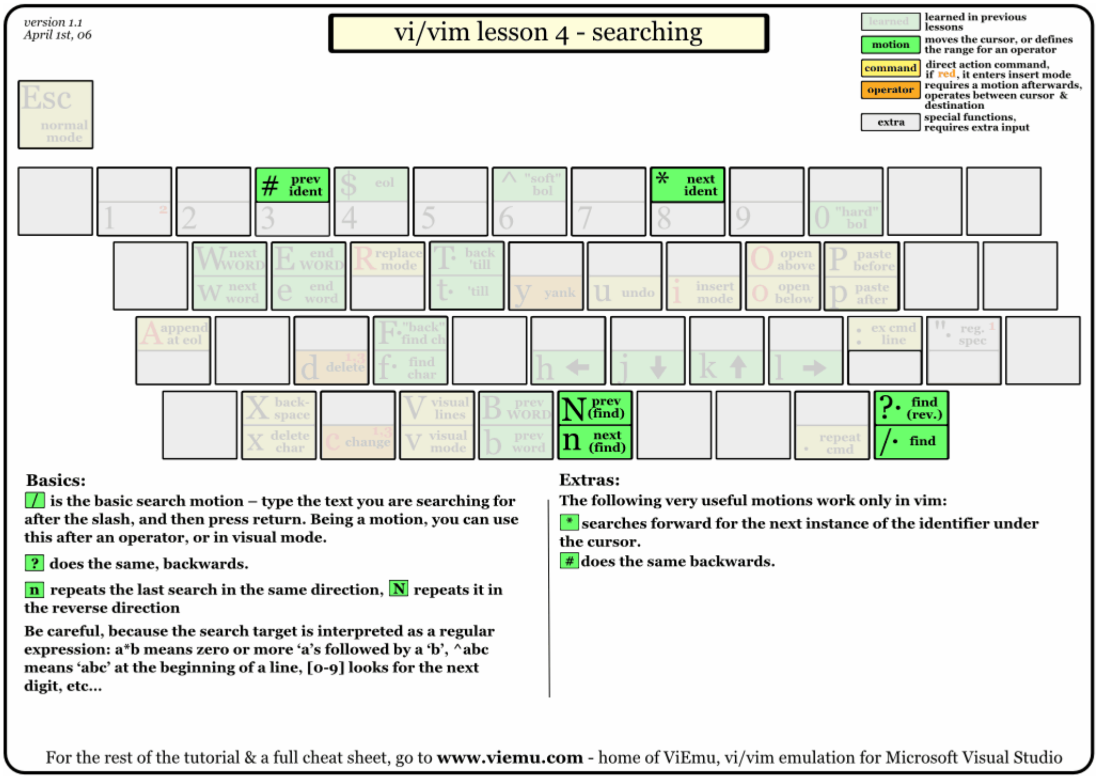
" * or " + select the system clipboard.

o enters insert mode in a new empty line below the current one.

O does the same above the current line.

For the rest of the tutorial & a full cheat sheet, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

The vim Command Set



Basics:

/ is the basic search motion – type the text you are searching for after the slash, and then press return. Being a motion, you can use this after an operator, or in visual mode.

? does the same, backwards.

n repeats the last search in the same direction, N repeats it in the reverse direction

Be careful, because the search target is interpreted as a regular expression: a*b means zero or more 'a's followed by a 'b', ^abc means 'abc' at the beginning of a line, [0-9] looks for the next digit, etc...

Extras:

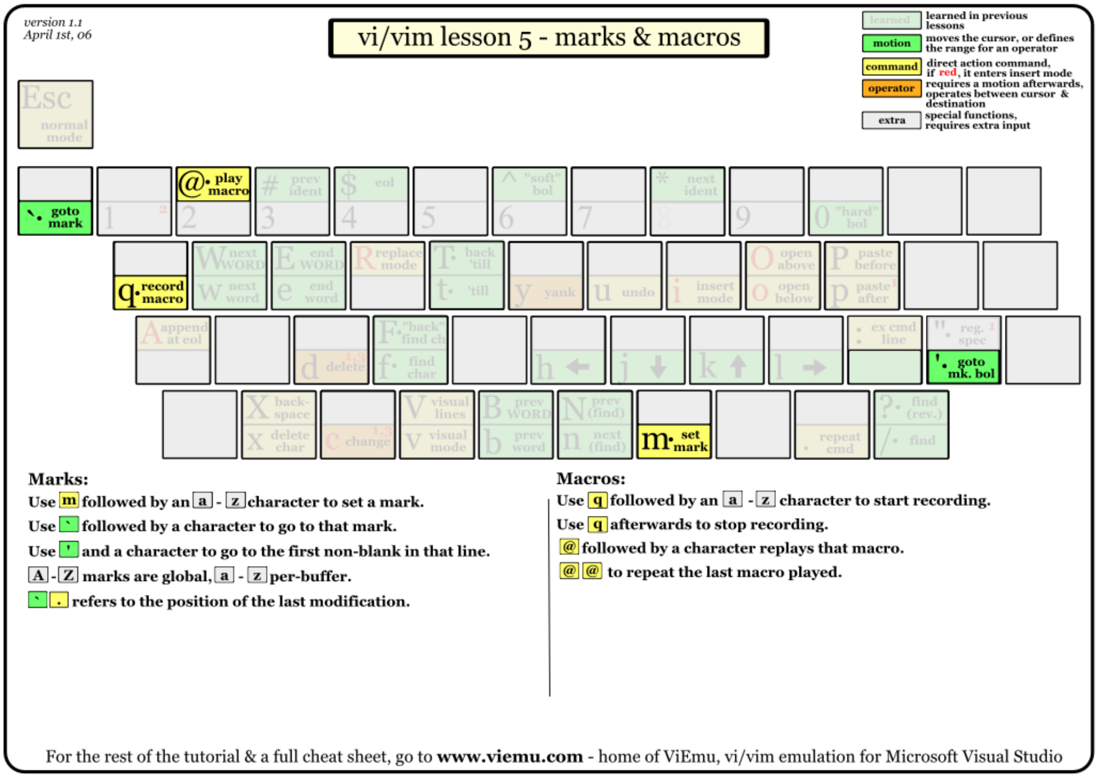
The following very useful motions work only in vim:

* searches forward for the next instance of the identifier under the cursor.

does the same backwards.

For the rest of the tutorial & a full cheat sheet, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

The vim Command Set



Marks:

Use m followed by an a - z character to set a mark.

Use ` followed by a character to go to that mark.

Use ' and a character to go to the first non-blank in that line.

A - Z marks are global, a - z per-buffer.

` . refers to the position of the last modification.

Macros:

Use q followed by an a - z character to start recording.

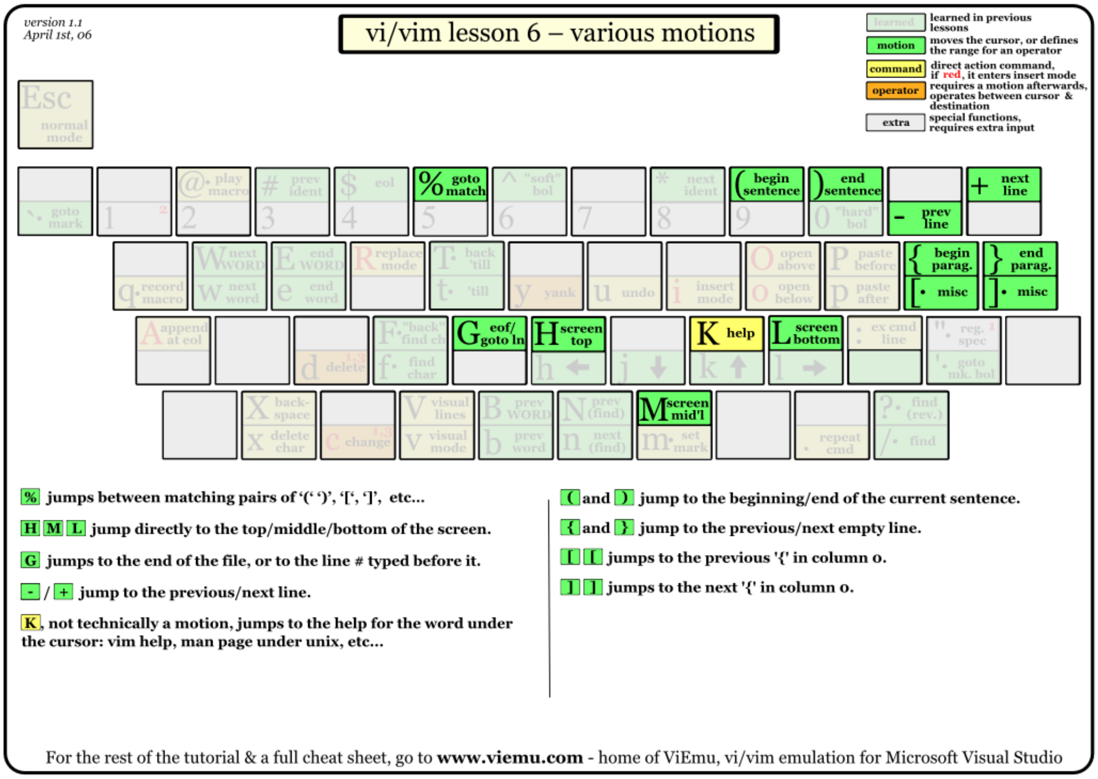
Use q afterwards to stop recording.

@ followed by a character replays that macro.

@ @ to repeat the last macro played.

For the rest of the tutorial & a full cheat sheet, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

The `vim` Command Set



vi/vim lesson 6 – various motions

learned	learned in previous lessons
motion	moves the cursor, or defines the range for an operator
command	direct action command, if red, it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input

% jumps between matching pairs of '(' ')', '[', ']', etc...

H **M** **L** jump directly to the top/middle/bottom of the screen.

G jumps to the end of the file, or to the line # typed before it.

- / **+** jump to the previous/next line.

K, not technically a motion, jumps to the help for the word under the cursor: vim help, man page under unix, etc...

(and **)** jump to the beginning/end of the current sentence.

{ and **}** jump to the previous/next empty line.

[**[** jumps to the previous '{' in column 0.

] **]** jumps to the next '{' in column 0.

For the rest of the tutorial & a full cheat sheet, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

3 Key Points

Key Points

- Text editors: why, what, and how
- The basic use of the vim editor