

VIM-PLUG-IN
c-support.vim
 VERSION 6.2
HOT KEYS

Key mappings for Vim and gVim.

<http://www.vim.org> — Wolfgang Mehner, wolfgang-mehner@web.de

(i) insert mode, (n) normal mode, (v) visual mode

<i>Help</i>		
<code>\he</code>	English dictionary	(n,i)
<code>\hd</code>	Doxygen command	(n,i)
<code>\hm</code>	manual for word under cursor	(n,i)
<code>\hp</code>	help (c-support)	(n,i)
<i>Comments</i>		
<code>[n]\cl</code>	end-of-line comment	(n,v,i)
<code>[n]\cj</code>	adjust end-of-line comment	(n,v,i)
<code>\cs</code>	set end-of-line comment column	(n)
<code>[n]\c*</code>	code ⇒ comment /* */	(n,v,i)
<code>[n]\cc</code>	code ⇒ comment //	(n,v,i)
<code>[n]\co</code>	comment ⇒ code	(n,v,i)
<code>[n]\cn</code>	toggle non-C comment	(n,v,i)
<code>\cfr</code>	frame comment	(n,i)
<code>\cfu</code>	function comment	(n,i)
<code>\cme</code>	method description	(n,i)
<code>\ccl</code>	class description	(n,i)
<code>\cfdi</code>	file description (implementation)	(n,i)
<code>\cfdh</code>	file description (header)	(n,i)
<code>\ccs</code>	C/C++-file sections (tab compl.)	(n,i)
<code>\chs</code>	H-file sections (tab compl.)	(n,i)
<code>\ckc</code>	keyword comment (tab compl.)	(n,i)
<code>\csc</code>	special comment (tab compl.)	(n,i)
<code>\cma</code>	template macros (tab compl.)	(n,i)
<code>\cd</code>	date	(n,v,i)
<code>\ct</code>	date & time	(n,v,i)
<code>[n]\cx</code>	exch. comment style: C ↔ C++	(n,v,i)

<i>Statements</i>		
<code>\sd</code>	do { } while	(n,v,i)
<code>\sf</code>	for	(n,i)
<code>\sfo</code>	for { }	(n,v,i)
<code>\si</code>	if	(n,i)
<code>\sif</code>	if { }	(n,v,i)
<code>\sie</code>	if else	(n,v,i)
<code>\sife</code>	if { } else { }	(n,v,i)
<code>\se</code>	else { }	(n,v,i)
<code>\sw</code>	while	(n,i)
<code>\swh</code>	while { }	(n,v,i)
<code>\ss</code>	switch	(n,v,i)
<code>\sc</code>	case	(n,i)
<code>\sb</code>	{ }	(n,v,i)

<i>Idioms</i>		
<code>\if</code>	function	(n,v,i)
<code>\isf</code>	static function	(n,v,i)
<code>\im</code>	main()	(n,v,i)
<code>\ie</code>	enum + typedef	(n,v,i)
<code>\is</code>	struct + typedef	(n,v,i)
<code>\iu</code>	union + typedef	(n,v,i)
<code>\ipr</code>	printf()	(n,i)
<code>\isc</code>	scanf()	(n,i)
<code>\ica</code>	p=calloc()	(n,i)
<code>\ima</code>	p=malloc()	(n,i)
<code>\ire</code>	p=realloc()	(n,i)
<code>\isi</code>	sizeof()	(n,v,i)
<code>\ias</code>	assert()	(n,v,i)
<code>\ii</code>	open input file	(n,v,i)
<code>\io</code>	open output file	(n,v,i)
<code>\ifsc</code>	fscanf	(n,i)
<code>\ifpr</code>	fprintf	(n,i)
<code>[n]\i0</code>	for(x=0; x<n; x+=1)	(n,v,i)
<code>[n]\in</code>	for(x=n-1; x>=0; x-=1)	(n,v,i)

<i>Preprocessor</i>		
<code>\pih</code>	include Std. Lib. header	(n,i)
<code>\piph</code>	include POSIX header	(n,i)
<code>\pg</code>	#include<...> (global)	(n,i)
<code>\pl</code>	#include"... " (local)	(n,i)
<code>\pd</code>	#define	(n,i)
<code>\pu</code>	#undef	(n,i)
<code>\pif</code>	#if #endif	(n,v,i)
<code>\pie</code>	#if #else #endif	(n,v,i)
<code>\pid</code>	#ifdef #else #endif	(n,v,i)
<code>\pin</code>	#ifndef #else #endif	(n,v,i)
<code>\pind</code>	#ifndef #def #endif	(n,v,i)
<code>\pe</code>	#error	(n,i)
<code>\pli</code>	#line	(n,i)
<code>\pp</code>	#pragma	(n,i)
<code>\pw</code>	#warning	(n,i)
<code>\pi0</code>	#if 0 #endif	(n,v,i)
<code>\pr0</code>	remove #if 0 #endif	(n,i)

<i>Snippet</i>		
<code>\nr</code>	read code snippet	(n,i)
<code>\nv</code>	view code snippet	(n,v,i)
<code>\nw</code>	write code snippet	(n,v,i)
<code>\ne</code>	edit code snippet	(n,i)
<code>[n]\nf</code>	pick up function prototype	(n,v,i)
<code>[n]\np</code>		(n,v,i)
<code>[n]\nm</code>	pick up method prototype	(n,v,i)
<code>\ni</code>	insert prototype(s)	(n,i)
<code>\nc</code>	clear prototype(s)	(n,i)
<code>\ns</code>	show prototype(s)	(n,i)
<code>\ntl</code>	edit local templates	(n,i)
<code>\ntc</code>	edit custom templates	(n, i)
<code>\ntp</code>	edit personal templates	(n, i)
<code>\ntr</code>	reread the templates	(n,i)
<code>\ntw</code>	template setup wizard	(n, i)
<code>\nts</code>	choose template style	(n, i)
<code>\njt</code>	insert jump tag	(n,i)

C++	
<code>\+ih</code>	<code>#include</code> C++ Std. Lib. header (n,i)
<code>\+ich</code>	<code>#include</code> C Std. Lib. header (n,i)
<code>\+om</code>	output manipulators (n,i)
<code>\+fb</code>	ios flagbits (n,i)
<code>\+c</code>	class (n,i)
<code>\+cn</code>	class (using new) (n,i)
<code>\+tc</code>	template class (n,i)
<code>\+tcn</code>	template class (using new) (n,i)
<code>\+ec</code>	error class (n,i)
<code>\+tf</code>	template function (n,i)
<code>\+tr</code>	try ... catch (n,v,i)
<code>\+ca</code>	catch (n,v,i)
<code>\+caa</code>	catch(...) (n,v,i)
<code>\+ex</code>	extern "C" { } (n,v,i)
<code>\+oif</code>	open input file (n,v,i)
<code>\+oof</code>	open output file (n,v,i)
<code>\+uns</code>	using namespace std; (n,v,i)
<code>\+un</code>	using namespace xxx; (n,v,i)
<code>\+unb</code>	namespace xxx { } (n,v,i)
<code>\+na</code>	namespace alias (n,v,i)
<code>\+rt</code>	RTTI (n,v,i)
<code>\+ic</code>	class implementation (n,i)
<code>\+icn</code>	class (using new) implementation (n,i)
<code>\+im</code>	method implementation (n,i)
<code>\+ia</code>	accessor implementation (n,i)
<code>\+itc</code>	template class implementation (n,i)
<code>\+itcn</code>	template class (using new) impl. (n,i)
<code>\+itm</code>	template method implementation (n,i)
<code>\+ita</code>	template accessor implementation (n,i)
<code>\+ioi</code>	operator » (n,i)
<code>\+ioo</code>	operator « (n,i)

Run	
<code>\rc</code>	save and compile (n,i)
<code>\rl</code>	link (n,i)
<code>\rr</code>	run (n,i)
<code>\ra</code>	set comand line arguments (n,i)
<code>\rd</code>	start debugger (n,i)
<code>\re</code>	executable to run ¹ (n,i)
<code>\rp</code>	run splint ² (n,i)
<code>\rpa</code>	cmd. line arg. for splint (n,i)
<code>\rcc</code>	run cppcheck ³ (n,i)
<code>\rccs</code>	severity for cppcheck (n,i)
<code>\rk</code>	run CodeCheck ⁴ (n,i)
<code>\rka</code>	cmd. line arg. for CodeCheck (n,i)
<code>\ri</code>	run indent (n,i)
<code>[n]\rh</code>	hardcopy buffer (n,i,v)
<code>\rs</code>	show plugin settings (n,i)
<code>\rx</code>	set xterm size (n,i, only Unix & GUI)
<code>\ro</code>	change output destination (n,i)

Tool Box : Make	
<code>\rm</code>	run make ¹ (n,i)
<code>\rmc</code>	run make clean ¹ (n,i)
<code>\rmd</code>	run make doc ¹ (n,i)
<code>\rcm</code>	choose a makefile ¹ (n,i)
<code>\rma</code>	cmd. line arg. for make ¹ (n,i)
Additional Mappings⁵	
typing	expansion
<code>/*</code>	<code>/* */</code> (i)
<code>/*</code>	<code>/* (multiline) marked text */</code> (v)
<code>/*<CR></code>	<code>/*</code> (i)
	<code>* </code>
	<code>*/</code>
<code>{<CR></code>	<code>{</code> (i)
	<code> </code>
	<code>}</code>
<code>{<CR></code>	<code>{</code> (v)
	<code>(multiline) marked text</code>
	<code>}</code>

Ex Commands

Set command line arguments (same as `\ra`)

`:CCmdlineArgs`

Set severity for cppcheck (same as `\rccs`)

`:CppcheckSeverity`

¹ also working for filetype **make**

² www.splint.org

³ cppcheck.sourceforge.net

⁴ **CodeCheckTM** is a product of Abraxas Software, Inc.

⁵ defined in `~/ftplugin/c.vim`