

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

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

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

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

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

C++	
<code>\+ih</code>	<code>#include C++ Std. Lib. header</code> (n,i)
<code>\+ich</code>	<code>#include C Std. Lib. header</code> (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>	<code>try ... catch</code> (n,v,i)
<code>\+ca</code>	<code>catch</code> (n,v,i)
<code>\+caa</code>	<code>catch(...)</code> (n,v,i)
<code>\+ex</code>	<code>extern "C" { }</code> (n,v,i)
<code>\+oif</code>	open input file (n,v,i)
<code>\+oof</code>	open output file (n,v,i)
<code>\+uns</code>	<code>using namespace std;</code> (n,v,i)
<code>\+un</code>	<code>using namespace xxx;</code> (n,v,i)
<code>\+unb</code>	<code>namespace xxx { }</code> (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> <code>* </code> <code>*/</code> (i)
<code>{<CR></code>	<code>{</code> <code> </code> <code>}</code> (i)
<code>{<CR></code>	<code>{</code> <code>(multiline) marked text</code> <code>}</code> (v)

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`