VIM-PLUG-IN

c-support.vim

VERSION 6.2.1

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

		***H**elp*
\he	English dictionary	(n,i)
\hd	Doxygen command	(n,i)
\hm	manual for word under cursor	(n,i)
\hp	help (c-support)	(n,i)

		***C**omments*
[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)

		***S**tatements*
\sfo	for (; ;)	*standard* (n,v,i)
\sfr	for (:)	*range-based* (n,v,i)
\sd	do { } while	(n,v,i)
\sw	while { }	(n,v,i)
\sif	if { }	(n,v,i)
\sie	if { } else { }	(n,v,i)
\sei	else if { }	(n,v,i)
\sel	else { }	(n,v,i)
\ss	switch	(n,v,i)
\sc	case	(n,i)
\sb	{ }	(n,v,i)

		***I**dioms*
\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)

		***P**reprocessor*
\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)

		***Sn**ippet*
\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)

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

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

		*Tool Box : **M**ake*[4,5]
\rm	run make	(n,i)
\rmc	run make clean	(n,i)
\rmd	run make doc	(n,i)
\rcm	choose a makefile	(n,i)
\rma	cmd. line arg. for make	(n,i)

Additional Mappings[4]

typing	**expansion**	
/*	/* */	(i)
/*	/* (multiline) marked text */	(v)

/*<CR>	/*
	  * |
	  */
		(i)

{<CR>	{
	  |
	}
		(i)

{<CR>	{
	  (multiline) marked text
	}
		(v)

Ex Commands

Set command line arguments (same as \ra)

```
:CCmdlineArgs
```

Set severity for cppcheck (same as \rccs)

```
:CppcheckSeverity
```

[1] *www.splint.org*
[2] *cppcheck.sourceforge.net*
[3] **CodeCheck**^{TM} is a product of Abraxas Software, Inc.
[4] see c-support/rc/c.vim for how to define them
[5] see c-support/rc/make.vim for how to define them for filetype **make**