




Emacs support for C++

Description	Keystroke	Function	Note
Support for the C++ Programming Language	 This table is in early development stage. More information will be written and content may change.		
Display current Mode settings	<ul style="list-style-type: none"><li>&lt;f12&gt; M-?</li><li>&lt;f11&gt; SPC C M-?</li></ul>	(pel-cc-mode-info)	Display information about current CC mode derivative for the current c++-mode buffer. <ul style="list-style-type: none"><li>Example of the information displayed (which reflects PEL's defaults):<pre>-UU-:----F1  hello.cpp      All (1,0)      (C++//1a WK) ----- - Tab width      : 3 - Indenting with : spaces only - Bracket style   : stroustrup - Comment style   : Line comments: // - Electric chars  : active: #*/&lt;&gt;(){};,, - Auto newline    : on - Syntactic indent : on - Hungry delete   : off, but the F11-⌘ and F11-⌘ keys are available.</pre></li></ul>
Toggle Electric state	<ul style="list-style-type: none"><li>C-c C-1</li><li>&lt;f12&gt; M-e</li><li>&lt;f11&gt; SPC C M-e</li></ul>	(c-toggle-electric-state &optional ARG)	Toggle the electric indentation feature done with the electric character keys. <ul style="list-style-type: none"><li>Optional numeric ARG, if supplied, turns on electric indentation when positive, turns it off when negative, and just toggles it when zero or left out.</li></ul>
Toggle auto-newline insertion mode	<ul style="list-style-type: none"><li>C-c C-a</li><li>&lt;f12&gt; M-RET</li><li>&lt;f11&gt; SPC C M-RET</li></ul>	(c-toggle-auto-newline &optional ARG)	Toggle <b>auto-newline</b> feature. <ul style="list-style-type: none"><li>Optional numeric ARG, if supplied, turns on auto-newline when positive, turns it off when negative, and just toggles it when zero or left out.</li><li>Turning on auto-newline automatically enables <b>electric indentation</b>.</li><li>When the auto-newline feature is enabled (indicated by "/la" on the mode line after the mode name) newlines are automatically inserted after special characters such as brace, comma, semi-colon, and colon.</li></ul>
Set indentation style	<ul style="list-style-type: none"><li>C-c .</li><li>&lt;f12&gt; M-s</li><li>&lt;f11&gt; SPC C M-s</li></ul>	(c-set-style STYLENAME &optional DONT-OVERRIDE)	Set the <u>bracket/indentation style</u> for the current buffer. <ul style="list-style-type: none"><li>Prompts for the name.</li><li>Supports tab completion (so use tab to see the list). Can be one of the <u>values supported by Emacs</u> but you can also add your customized mode with some Emacs Lisp code.</li></ul>
Toggle syntactic indentation	<ul style="list-style-type: none"><li>&lt;f12&gt; M-i</li><li>&lt;f11&gt; SPC C M-i</li></ul>	(c-toggle-syntactic-indentation &optional ARG)	Toggle syntactic indentation. <ul style="list-style-type: none"><li>Optional numeric ARG, if supplied, turns on syntactic indentation when positive, turns it off when negative, and just toggles it when zero or left out.</li><li>When syntactic indentation is turned on (the default), the indentation functions and the electric keys indent according to the syntactic context keys, when applicable.</li><li>When it's turned off, the electric keys don't reindent, the indentation functions indents every new line to the same level as the previous nonempty line, and M-x c-indent-command adjusts the indentation in steps specified by 'c-basic-offset'. The indentation style has no effect in this mode, nor any of the indentation associated variables, e.g. 'c-special-indent-hook'.</li></ul>
Toggle Comment Style	<ul style="list-style-type: none"><li>C-c C-k</li><li>&lt;f12&gt; M-;</li><li>&lt;f11&gt; SPC C M-;</li></ul>	(c-toggle-comment-style &optional ARG)	Toggle the comment style between block and line comments. <ul style="list-style-type: none"><li>Optional numeric ARG, if supplied, switches to block comment style when positive, to line comment style when negative, and just toggles it when zero or left out.</li></ul>
Toggle Hungry Delete mode	<ul style="list-style-type: none"><li>&lt;f12&gt; M-DEL</li><li>&lt;f11&gt; SPC C M-DEL</li></ul>	(c-toggle-hungry-state &optional ARG)	Toggle hungry-delete-key feature. Affect <b>&lt;DEL&gt;</b> and <b>C-d</b> keys. <ul style="list-style-type: none"><li>Optional numeric ARG, if supplied, turns on hungry-delete when positive, turns it off when negative, and just toggles it when zero or left out.</li><li>When the hungry-delete-key feature is enabled (indicated by "/h" on the mode line after the mode name) the delete key gobbles all preceding whitespace in one fell swoop.</li></ul>  More commands can be used to perform hungry delete. See the section on hungry delete below.
Open Line in Context (See also: ⌘ Whitespace)	<ul style="list-style-type: none"><li>&lt;f12&gt; RET</li><li>&lt;f11&gt; SPC C RET</li></ul>	(c-context-open-line)	Insert a line break suitable to the context and leave point before it. <ul style="list-style-type: none"><li>This is the '<b>c-context-line-break</b>' equivalent to '<b>open-line</b>', which is normally bound to <b>C-o</b>. See 'c-context-line-break' for the details.</li></ul>
Electric Keys and Keywords	The following characters have special meaning when the electrical state is active in a buffer using c-mode.		
	#	(c-electric-pound ARG)	Insert a "#". <ul style="list-style-type: none"><li>If 'c-electric-flag' is set, handle it specially according to the variable 'c-electric-pound-behavior', which can only be nil or 'alignleft'. If a numeric ARG is supplied, or if point is inside a literal or a macro, nothing special happens.</li></ul>
	( )	(c-electric-paren ARG)	Insert a parenthesis. <ul style="list-style-type: none"><li>If 'c-syntactic-indentation' and 'c-electric-flag' are both non-nil, the line is reindented unless a numeric ARG is supplied, or the parenthesis is inserted inside a literal.</li><li>Whitespace between a function name and the parenthesis may get added or removed; see the variable 'c-cleanup-list'.</li><li>Also, if 'c-electric-flag' and 'c-auto-newline' are both non-nil, some newline cleanups are done if appropriate; see the variable 'c-cleanup-list'.</li></ul>
	{ }	(c-electric-brace ARG)	Insert a brace. <ul style="list-style-type: none"><li>If 'c-electric-flag' is non-nil, the brace is not inside a literal and a numeric ARG hasn't been supplied, the command performs several electric actions:<ul style="list-style-type: none"><li>If the auto-newline feature is turned on (indicated by "/la" on the mode line) newlines are inserted before and after the brace as directed by the settings in 'c-hanging-braces-alist'.</li><li>Any auto-newlines are indented. The original line is also reindented unless 'c-syntactic-indentation' is nil.</li><li>If auto-newline is turned on, various newline cleanups based on the settings of 'c-cleanup-list' are done.</li></ul></li></ul>
	:	(c-electric-colon ARG)	Insert a colon. <ul style="list-style-type: none"><li>If 'c-electric-flag' is non-nil, the colon is not inside a literal and a numeric ARG hasn't been supplied, the command performs several electric actions:<ul style="list-style-type: none"><li>If the auto-newline feature is turned on (indicated by "/la" on the mode line) newlines are inserted before and after the colon based on the settings in 'c-hanging-colons-alist'.</li><li>Any auto-newlines are indented. The original line is also reindented unless 'c-syntactic-indentation' is nil.</li><li>If auto-newline is turned on, whitespace between two colons will be "cleaned up" leaving a scope operator, if this action is set in 'c-cleanup-list'.</li></ul></li></ul>
	; ,	(c-electric-semi&comma ARG)	Insert a comma or semicolon. <ul style="list-style-type: none"><li>If 'c-electric-flag' is non-nil, point isn't inside a literal and a numeric ARG hasn't been supplied, the command performs several electric actions:<ul style="list-style-type: none"><li>When the auto-newline feature is turned on (indicated by "/la" on the mode line) a newline might be inserted. See the variable 'c-hanging-semi&amp;comma-criteria' for how newline insertion is determined.</li><li>Any auto-newlines are indented. The original line is also reindented unless 'c-syntactic-indentation' is nil.</li><li>If auto-newline is turned on, a comma following a brace list or a semicolon following a defun might be cleaned up, depending on the settings of 'c-cleanup-list'.</li></ul></li></ul>

Description	Keystroke	Function	Note
	<ul style="list-style-type: none"> <li>&lt;</li> <li>&gt;</li> </ul>	(c-electric-lt-gt ARG)	<p>If the current language uses angle bracket parens (e.g. template arguments in C++), try to find out if the inserted character is a paren and give it paren syntax if appropriate.</p> <p>If ‘c-electric-flag’ and ‘c-syntactic-indentation’ are both non-nil, the line will be reindented if the inserted character is a paren or if it finishes a C++ style stream operator in C++ mode. Exceptions are when a numeric argument is supplied, or the point is inside a literal.</p>
<b>C++ Comments</b>	2 more characters have electric behaviour: / and * to help support comments in C++. C++ supports 2 types of comments: <ul style="list-style-type: none"> <li>Block Comments: <code>/* comment */</code></li> <li>Line Comments: <code>// comment to end of line</code></li> </ul>		
	/	(c-electric-slash ARG)	<p>Insert a slash character.</p> <ul style="list-style-type: none"> <li>If the slash is inserted immediately after the comment prefix in a c-style comment, the comment might get closed by removing whitespace and possibly inserting a <code>""</code>. See the variable ‘c-cleanup-list’.</li> <li>Indent the line as a comment, if:               <ol style="list-style-type: none"> <li>The slash is second of a <code>"/"</code> line oriented comment introducing token and we are on a comment-only-line, or</li> <li>The slash is part of a <code>"/"</code> token that closes a block oriented comment.</li> </ol> </li> <li>If a numeric ARG is supplied, point is inside a literal, or ‘c-syntactic-indentation’ is nil or ‘c-electric-flag’ is nil, indentation is inhibited.</li> </ul>
	*	(c-electric-star ARG)	<p>Insert a star character.</p> <ul style="list-style-type: none"> <li>If ‘c-electric-flag’ and ‘c-syntactic-indentation’ are both non-nil, and the star is the second character of a C style comment starter on a comment-only-line, indent the line as a comment.</li> <li>If a numeric ARG is supplied, point is inside a literal, or ‘c-syntactic-indentation’ is nil, this indentation is inhibited.</li> </ul> <p>With this key it becomes easy to type the following two styles of multi-line block comment:</p> <pre> /* Two star ** continuation ** prefix for ** multi-line ** C comment. */  /* Single star  * prefix for  * multi-line  * C comment.  */ </pre> <p>When typing the <code>""</code> at the beginning of the line, it indents automatically. If another <code>""</code> is typed, indentation is set to allow a two-star continuation, otherwise it is placed for a single star continuation.</p>
Comment/un-comment	M-;	(comment-dwim ARG)	<p>Comment line or region with <code>//</code> or <code>/* */</code> style comments depending on the comment style currently used in the buffer.</p> <ul style="list-style-type: none"> <li>When no marked region and no comment:               <ul style="list-style-type: none"> <li>On empty line: insert comment starter at the proper indentation level. Typed again: move it toward end of line.</li> <li>On line with code: insert comment starter after the code for an end-of-line comment</li> </ul> </li> <li>With marked un-commented region:               <ul style="list-style-type: none"> <li>Comment region (each line is commented)</li> </ul> </li> <li>With marked commented region:               <ul style="list-style-type: none"> <li>removes the comment.</li> </ul> </li> </ul> <ul style="list-style-type: none"> <li>Call the comment command you want (Do What I Mean).               <ul style="list-style-type: none"> <li>If the region is active and ‘transient-mark-mode’ is on, call ‘comment-region’ (unless it only consists of comments, in which case it calls ‘uncomment-region’). Else, if the current line is empty, call ‘comment-insert-comment-function’ if it is defined, otherwise insert a comment and indent it. Else if a prefix ARG is specified, call ‘comment-kill’. Else, call ‘comment-indent’.</li> </ul> </li> <li>You can configure ‘comment-style’ to change the way regions are commented: see <b>&lt;F12&gt; M-;</b> to toggle the comment style.</li> </ul>
Fill current paragraph (See also: ⌘ Filling/Justification)	<ul style="list-style-type: none"> <li>M-q</li> <li>&lt;f12&gt; f</li> <li>&lt;f11&gt; SPC C f</li> </ul>	(c-fill-paragraph &optional ARG)	<p>Like <b>&lt;f11&gt; t f p</b> but handles <code>//</code> and <code>/* */</code> style comments.</p> <ul style="list-style-type: none"> <li>If any of the current line is a comment or within a comment, fill the comment or the paragraph of it that point is in, preserving the comment indentation or line-starting decorations (see the ‘c-comment-prefix-regexp’ and ‘c-block-comment-prefix’ variables for details).</li> <li>If point is inside multiline string literal, fill it. This currently does not respect escaped newlines, except for the special case when it is the very first thing in the string. The intended use for this rule is in situations like the following:               <pre> char description[] = "\ A very long description of something that you want to fill to make nicely formatted output." </pre> </li> </ul> <ul style="list-style-type: none"> <li>If point is in any other situation, i.e. in normal code, do nothing.</li> <li>Optional prefix ARG means justify paragraph as well.</li> </ul>
Toggle subword-mode (See also: ⌘ Text Modes)	<ul style="list-style-type: none"> <li>&lt;f11&gt; t m b</li> <li>&lt;f12&gt; M-b</li> <li>&lt;f11&gt; SPC C M-b</li> </ul>	(subword-mode &optional ARG)	<p>Toggle subword-mode: a minor mode that treats sections of <u>camelCase</u> and <u>PascalCase</u> as distinct words.</p> <ul style="list-style-type: none"> <li>With a prefix argument ARG, enable Subword mode if ARG is positive, and disable it otherwise.</li> </ul>
<b>Hungry Deletion of Whitespace</b>	<p>The CC mode provides two commands that can perform “hungry whitespace deletion” that can also be used in every mode.</p> <ul style="list-style-type: none"> <li>👉 PEL provides the convenient keys with the <b>&lt;f11&gt;</b> prefix keys for those 2 commands, available in <b>all</b> modes.</li> <li>In modes compatible with the CC Mode (e.g. for C, C++, D, Java, Pike, etc..) it is also possible to activate the Hungry Delete Mode to modify the behaviour of the simple <b>&lt;DEL&gt;</b> and <b>C-d</b>, to perform hungry deletions. That’s not currently supported in other modes.               <ul style="list-style-type: none"> <li>When the Hungry Delete Mode is on, the mode-line displays a ‘h’ to the right of the ‘/!’ indication of electric mode.</li> <li>The Hungry Mode also activates the key prefixes below that start with <b>C-c</b>. They are listed but remember they are only available once the Hungry state mode is activated (and that can only be done in modes that are CC Mode compatible).</li> <li>In modes derived from CC Mode you can also activate the hungry state to make standard delete commands delete hungrily, but that does not work for other modes. PEL provides the <b>&lt;f12&gt; M-DEL</b> key for those modes, like the D Mode. See above.</li> </ul> </li> </ul>		
Delete preceding char or all preceding whitespace.  (See also: ⌘ Cut & Paste)	<ul style="list-style-type: none"> <li>C-c DEL</li> <li>C-c ☒</li> <li>C-c C-☒</li> <li>C-c &lt;C-backspace&gt;</li> <li>C-c C-DEL</li> <li>&lt;f11&gt; ☒</li> </ul>	(c-hungry-delete-backwards)	<p>Delete the preceding character or all preceding whitespace back to the previous non-whitespace character.</p> <p>👉 In terminal mode, even though <b>C-☒</b>, <b>&lt;C-backspace&gt;</b> and <b>C-DEL</b> are not available, they are mapped to the non-control key so attempting to type them end up invoking the command anyway because the first key bindings are recognized.</p> <p>👉 With PEL, the <b>&lt;f11&gt; ☒</b> binding is always available, in all modes. The other keys are only available in modes derived from the CC Mode. This prevents conflicts with other modes that may use the popular C-c bindings.</p>

Description	Keystroke	Function	Note
Delete next char or all following whitespace.  (See also: ⌘ Cut & Paste)	<ul style="list-style-type: none"> <li>• <b>C-c C-d</b></li> <li>• <b>C-c</b> </li> <li>• <b>C-c C-</b></li> <li>• <b>C-c &lt;C-delete&gt;</b></li> <li>• <b>&lt;f11&gt;</b> </li> </ul>	(c-hungry-delete-forward)	Delete the following character or all following whitespace up to the next non-whitespace character. In terminal mode, even though <b>C-</b> and <b>&lt;C-delete&gt;</b> are not available, they are mapped to the non-control key so attempting to type them end up invoking the command anyway because the first key bindings are recognized. 👉 With PEL, the <b>&lt;f11&gt;</b> binding is always available, in all modes. The other keys are only available in modes derived from the CC Mode. This prevents conflicts with other modes that may use the popular C-c bindings.
<b>Indentation</b>	All syntactic indentation control for D is controlled by the CC-Mode logic and provided commands listed below. • Rigid indentation commands are also available and listed at the end of this list. They are also listed in the ⌘ Indentation table.		
Indent current line or region  (See also: ⌘ Indentation)	<b>&lt;tab&gt;</b>	(c-indent-line-or-region &optional ARG REGION)	Indent active region, current line, or block starting on this line.  Behaviour depends on syntactic-indentation mode (enabled by default but can be toggled on/off with the <b>&lt;f12&gt; M-i</b> key): <ul style="list-style-type: none"> <li>• With syntactic-indentation on (the default): <ul style="list-style-type: none"> <li>• In Transient Mark mode, when the region is active, reindent the region.</li> <li>• Otherwise, with a prefix argument, rigidly reindent the expression starting on the current line.</li> <li>• Otherwise reindent just the current line.</li> </ul> 👉 This might seem strange for new Emacs users, but it ends up being very useful. You can type <b>&lt;tab&gt;</b> anywhere in the line to adjust the indentation of the current line or everything in the marked area if a block is marked.</li> <li>• With syntactic-indentation off: <ul style="list-style-type: none"> <li>• <b>&lt;tab&gt;</b> always indent current line by one level</li> <li>• <b>C-u - &lt;tab&gt;</b> or <b>M- &lt;tab&gt;</b> always un-indent current line by one level</li> <li>• Indenting marked region is done without syntax knowledge and at the same level as previous line.</li> </ul> </li> <li>• 👉 If you want to indent rigidly you can use: <ul style="list-style-type: none"> <li>• (<b>pel-indent-rigidly &amp;optional N</b>) (bound to <b>C-x &lt;tab&gt;</b> and to <b>&lt;f11&gt; &lt;tab&gt;&lt;tab&gt;</b>) to indent the line or region rigidly.</li> <li>• (<b>tab-to-tab-stop</b>), bound to <b>M-i</b> to insert spaces to the next tab stop column.</li> </ul> </li> </ul>
Indent lines of list after point (See also: ⌘ Indentation)	<b>C-M-q</b>	(indent-pp-sexp &optional ARG)	Indent each line of the list starting just after point, or pretty-print it. • A prefix argument ( <b>C-u</b> ) specifies pretty-printing. Pretty-printing essentially uses more lines as it places the beginning of each list on a new line.
Indent current function or class	<b>C-c C-q</b>	(c-indent-defun)	Indent the content of the current top-level function or class. Leaves point unchanged.
Indent a region	<b>C-M-\</b>	(indent-region START END &optional COLUMN)	Indent each nonblank line in the region. <ul style="list-style-type: none"> <li>• A numeric prefix argument specifies a column: indent each line to that column.</li> <li>• With no prefix argument, the command chooses one of these methods and indents all the lines with it: <ol style="list-style-type: none"> <li>1. If ‘fill-prefix’ is non-nil, insert ‘fill-prefix’ at the beginning of each line in the region that does not already begin with it.</li> <li>2. If ‘indent-region-function’ is non-nil, call that function to indent the region.</li> <li>3. Indent each line via ‘indent-according-to-mode’.</li> </ol> </li> </ul> 👉 When a region is marked you can also use the simple <b>&lt;tab&gt;</b> to do the same when syntactic-indentation is active.
<b>Non Syntactic Indentation</b>	Emacs provides the following command to indent without regards to semantics. More information on indentation is available in the ⌘ Indentation table.		
Insert spaces or tabs to next defined tab-stop column (See also: ⌘ Indentation)	<b>M-i</b>	(tab-to-tab-stop)	Insert spaces or tabs to next defined tab-stop column. <ul style="list-style-type: none"> <li>• The exact location of the next tab stop is identified by the value of the <b>tab-stop-list</b> and <b>tab-width</b> for the current buffer.</li> </ul>
Indent/Unindent rigidly  (See also: ⌘ Indentation)	<ul style="list-style-type: none"> <li>• <b>C-x &lt;tab&gt;</b></li> <li>• <b>&lt;f11&gt; &lt;tab&gt; &lt;tab&gt;</b></li> </ul>	<b>(pel-indent-rigidly &amp;optional N)</b>   ----- ✂ PEL uses the above instead of the standard:  <b>(indent-rigidly START END ARG &amp;optional INTERACTIVE)</b>	Indent rigidly the marked region or current line N times. <ul style="list-style-type: none"> <li>• <b>If a region is marked</b>, it uses ‘indent-rigidly’ and provides the same prompts to control indentation changes.</li> <li>• <b>If no region is marked</b>, it operates on current line(s) identified by the numeric argument N (or if not specified N=1): <ul style="list-style-type: none"> <li>• N = [-1, 0, 1] : operate on current line</li> <li>• N &gt; 1 : operate on the current line and N-1 lines below.</li> <li>• N &lt; -1 : operate on the current line and (abs N) -1 lines above.</li> </ul> </li> </ul> ✂ PEL rebinds this key, but it extends the functionality: pel-indent-rigidly uses indent-rigidly, described below the dashed line. ----- Indent all lines starting in the region. <ul style="list-style-type: none"> <li>• If called interactively with no prefix argument, activate a transient mode in which the indentation can be adjusted interactively by typing <b>&lt;left&gt;</b>, <b>&lt;right&gt;</b>, <b>&lt;S-left&gt;</b>, or <b>&lt;S-right&gt;</b>.</li> </ul> ----- These commands activate a transient mode where Emacs prompts for extra keys to control how to indent. Indenting and un-indenting is possible. The capabilities are controlled by the variable <i>indent-rigidly-map</i> with by default provides: <ul style="list-style-type: none"> <li>• <b>S-&lt;right&gt;</b> indent-rigidly-right-to-tab-stop</li> <li>• <b>S-&lt;left&gt;</b> indent-rigidly-left-to-tab-stop</li> <li>• <b>&lt;right&gt;</b> indent-rigidly-right</li> <li>• <b>&lt;left&gt;</b> indent-rigidly-left</li> </ul> Typing any other key deactivates the transient mode. ⚠ Since cua-mode uses <b>C-x</b> , to invoke this command when cua-mode is active, type it really fast or type <b>C-x C-x &lt;tab&gt;</b> (or use the PEL binding <b>&lt;f11&gt; &lt;tab&gt; &lt;tab&gt;</b> ).
Insert an indented line below current line (See also: ⌘ Indentation)	<ul style="list-style-type: none"> <li>• <b>M-&lt;RET&gt;</b></li> <li>• <b>&lt;f11&gt; &lt;tab&gt; &lt;RET&gt;</b></li> </ul>	(pel-newline-and-indent-below)	Insert an indented line just below current line. <ul style="list-style-type: none"> <li>• This does the same as the normal <b>&lt;RET&gt;</b> in d-mode.</li> </ul>
Indent rigidly C-mode style	<ul style="list-style-type: none"> <li>• <b>&lt;f6&gt; &lt;tab&gt;</b></li> <li>• <b>&lt;f11&gt; &lt;tab&gt; c</b></li> </ul>	(pel-insert-c-indent &optional N)	Insert as many spaces as identified by <b>c-basic-offset</b> variable on the current line or all marked lines. <ul style="list-style-type: none"> <li>• If a region was marked before the command it remains marked, allow further use of the same or other command to control the region. Use <b>C-g</b> to de-activate the region.</li> </ul>
Un-indent rigidly C-mode style	<ul style="list-style-type: none"> <li>• <b>&lt;backtab&gt;</b></li> <li>• <b>&lt;f6&gt; &lt;backtab&gt;</b></li> <li>• <b>&lt;f11&gt; &lt;tab&gt; C</b></li> </ul>	(pel-unindent &optional N)	Un-indent current line or marked lines by N times <b>c-basic-offset</b> spaces. <ul style="list-style-type: none"> <li>• Works for point is anywhere on the line.</li> <li>• If a region was marked before the command it remains marked, allow further use of the same or other command to control the region. Use <b>C-g</b> to de-activate the region.</li> <li>• 🚧 Limitation: does not handle hard tabs properly.</li> </ul>
<b>Marking</b>	Emacs provides the following command to quickly mark the whole content of the current function. More mark commands exists, see the ⌘ Marking table.		

Description	Keystroke	Function	Note
<b>Mark the complete function body</b>  (See also: ⌘ Marking)	C-M-h	(c-mark-function)	Mark complete function. <ul style="list-style-type: none"> <li>Put mark at end of the current top-level declaration or macro, point at beginning.</li> <li>If point is not inside any then the closest following one is chosen. Each successive call of this command extends the marked region by one function.</li> <li>A mark is left where the command started, unless the region is already active (in Transient Mark mode).</li> <li>As opposed to C-M-a and C-M-e, this function does not require the declaration to contain a brace block.</li> </ul>
<b>Style Management</b>	 To be written		
Guess style	M-x c-guess		
	M-x c-guess-buffer		
	M-x c-guess-region		
View Guessed style	M-x c-guess-view		
Show/Modify syntactic context	C-c C-o		

### Emacs & C++ — References

Document	Notes
<a href="#">GNU emacs - CC Mode Manual</a>	
<a href="#">GNU Emacs Manual - Styles</a>	
<a href="#">Emacs BSD/Allman Style with 4 Space Tabs?</a>	
<a href="#">Emacs: Linux Kernel Style but with Allman/BSD Style Braces?</a>	
<a href="#">Emacs Wiki - Indenting C</a>	
<a href="#">Indent preprocessor directives as C code in emacs</a>	Does not fully address the way I want to have multi-indentations for pre-processor
<a href="#">elisp code - ppindent.el</a>	Implements pre-processor indentation with the # always in the first column. Not yet exactly what I want.
<a href="#">Demystify C++ Metaprograms using Emacs</a>	
<a href="#">Programming in C++, Rules and Recommendations</a>	ellemtel style
<a href="#">company-mode ; Modular in-buffer completion framework for Emacs</a>	