

# Coding Standard

Language: C++

Author: Dan Turk

Date: 1997 Mar 24

Adapted from:

Humphrey, Watts S. (1995). *A Discipline for Software Engineering*.

New York: Addison-Wesley. Pages 670-672.

Purpose	To guide the development of C++ programs
Program Headers	<ul style="list-style-type: none"><li>• Begin all programs with a descriptive header.</li></ul>
Program Header Format	<pre>/****** * * Program Name: <i>the program name</i> * Version: <i>version number</i> * * Author: <i>the author's name</i> * * Date written: <i>date</i> * * Description: <i>sentence / paragraph description of what the</i> * <i>program does</i> * * Platform(s) tested on: <i>hardware / OS</i> * * Modification history: * <i>list of dates, authors, and changes made</i> * *****/</pre>

Program Header Example	<pre> /***** * * Program Name: <i>date.cpp</i> * Version: <i>1.0</i> * * Author: <i>Dan Turk</i> * * Date written: <i>1997 Mar 10</i> * * Description: <i>date.cpp performs data calculations such * as determining the how many days old a person is, * given the current date and their birthday, determining * the day a person was born given the current date, * day of week, and their birthday, determining if a * given year is a leap year or not, determining how * many days there are in a given month, etc.</i> * * Platform(s) tested on: <i>Pentium / Windows 95</i> * * Modification history: *   <i>1997 Mar 21   Dan Turk   Added "days old" calculations</i> *   <i>1997 Mar 15   Dan Turk   Corrected leap year determination</i> * *****/ </pre>
Function Headers	<ul style="list-style-type: none"> <li>• Begin each function with a descriptive header.</li> </ul>
Function Header Format	<pre> /***** * * Function Name: <i>the function name</i> * Version: <i>version number</i> * * Author: <i>the author's name</i> * * Date written: <i>date</i> * * Description: <i>sentence / paragraph description of what the * function does</i> * * Parameter descriptions: *   <i>name and description of each parameter</i> * * Modification history: *   <i>list of dates, authors, and changes made</i> * *****/ </pre>

Function Header Example	<pre> /***** * * Function Name: <i>days_in_month()</i> * Version: <i>1.0</i> * * Author: <i>Dan Turk</i> * * Date written: <i>1990 Feb 13</i> * * Description: <i>days_in_month()</i> determines the number of days in month <i>m</i>. *   It returns the days in the month if <i>m</i> is a valid month number (1-12), *   or -1 if <i>m</i> is invalid. * * Parameter descriptions: *   INPUT: *       <i>m</i>   month (1-12) for which to determine number of days *   OUTPUT: *       none *   RETURN: *       28, 29, 30, or 31 depending on the valid month number *       -1 if <i>m</i> is not valid * * Modification history: * *****/ </pre>
White Space	<ul style="list-style-type: none"> <li>• Write programs with sufficient spacing so that they do not appear crowded.</li> <li>• Separate every program construct with at least one space.</li> </ul>
Blank Lines	<ul style="list-style-type: none"> <li>• Use blank lines to separate logical blocks of code and to improve readability.</li> <li>• Put at least one blank line between the end of one function and the beginning of the next.</li> </ul>
Indentation	<ul style="list-style-type: none"> <li>• Indent every level of logic from the previous one.</li> <li>• Indent a minimum of 2 and a maximum of 8 spaces for each additional level.</li> <li>• Start all lines at the same logical level at the same indentation level.</li> </ul>
Line Spacing	<ul style="list-style-type: none"> <li>• Single-space all lines, except when double-spacing (inserting blank lines) will clarify sections of code, such as setting off logical blocks of code from one another.</li> </ul>
Begin-End block delimiters	<ul style="list-style-type: none"> <li>• Put begin-block braces on the same line as the beginning statement.</li> <li>• Put end-block braces on a separate line, indented to the same level as all code within the block.</li> </ul>

Examples of effective use of White Space, Blank Lines, Indentation, and Line Spacing	<ul style="list-style-type: none"> <li>Good Example: <pre> void main (void) {      int i, n;      for (i=0; i&lt;n; i++) {         cout &lt;&lt; i;         cout &lt;&lt; " Hello, world!\n";     } // for      } // main() </pre> </li> <li>Bad Example: <pre> void main(void){ int i,n; for(i=0;i&lt;n;i++) { cout&lt;&lt;i; cout&lt;&lt;"Hello, world!\n"; } } </pre> </li> </ul>
Grouping	<ul style="list-style-type: none"> <li>Group logical types of code together (Ex: #includes for header files, #defines, prototypes)</li> </ul>
Prototypes	<ul style="list-style-type: none"> <li>Declare all prototypes at the beginning of the program before main()</li> </ul>
Includes	<ul style="list-style-type: none"> <li>Include all header files at the beginning of the program before main()</li> </ul>
Defines	<ul style="list-style-type: none"> <li>Define all constants &amp; macros at the beginning of the program before main()</li> </ul>
Naming Conventions	<ul style="list-style-type: none"> <li>Use meaningful names for all variables, constants, and functions.</li> <li>Use lower-case names for variables, and upper-case for constants.</li> <li>Separate portions of long names with underscores.</li> <li>Examples of good naming conventions: <pre> int total_cost, color; #define TAX_RATE 0.06 void calculate_taxes (float gross_income, int exemptions); </pre> </li> <li>Examples of bad naming conventions: <pre> int tc, c; #define R 0.06 void calc (float gi, int ex); </pre> </li> </ul>

Comments	<ul style="list-style-type: none"> <li>• Document the code as necessary so the reader can easily understand it.</li> <li>• Make sure comments say more than what the code already says.</li> <li>• Do not comment every line of code.</li> <li>• Comment the beginning of logical blocks of code.</li> <li>• Clarify end-blocks by commenting them.</li> </ul> <ul style="list-style-type: none"> <li>• Good Examples: <pre> // read until EOF and count number of input items total = 0; while (cin &lt;&lt; i) {     n++;     total += i; } // while avg = total / n;  // print results cout &lt;&lt; "total=" &lt;&lt; total &lt;&lt; "\n"; cout &lt;&lt; "n=" &lt;&lt; n &lt;&lt; "\n"; cout &lt;&lt; "avg=" &lt;&lt; avg &lt;&lt; "\n"; </pre> </li> <li>• Bad Examples: <pre> total = 0;           // set total to zero while (cin &lt;&lt; i)    // read i {     n++;           // add 1 to n     total += i;    // add i to total }  avg = total / n;    // calculate average cout &lt;&lt; "total=" &lt;&lt; total &lt;&lt; "\n";    // print total cout &lt;&lt; "n=" &lt;&lt; n &lt;&lt; "\n";          // print n cout &lt;&lt; "avg=" &lt;&lt; avg &lt;&lt; "\n";      // print average </pre> </li> </ul>
----------	--