C++ review checklists

Personal Software Process for Engineers

# Table C84: C++ PSP3 Design Review Checklist

| Program Name and Number: | | | | | |
| --- | --- | --- | --- | --- | --- |
| Purpose | To guide you in conducting an effective design review |  |  |  |  |
| General | * As you complete each review step, check off that item in the box to the right. * Complete the checklist for one program unit before you start to review the next. * **As you encounter issues whose resolution must be deferred, record them in the issue tracking log.** |  |  |  |  |
| Complete | Ensure that the requirements, specifications, and high-level design are completely covered by the design:   * All specified outputs are produced. * All needed inputs are furnished. * All required includes are stated. |  |  |  |  |
| State machine | Verify the state machine design:   * The structure has no hidden traps or loops. * It is complete; that is, all possible states have been identified. * It is orthogonal; that is, for every set of conditions, there is one and only one possible next state. * The transitions from each state are complete and orthogonal; that is, from every state, a unique next state is defined for every possible combination of state machine input values. |  |  |  |  |
| Logic | * Verify that program sequencing is proper:   - Stacks, lists, and so on are in the proper order.  - Recursion unwinds properly.   * Verify that all loops are properly initiated, incremented, and terminated. * **Use defined verification methods, such as execution tables, trace tables, or mathematical verification.** |  |  |  |  |
| Special cases | Check all special cases:   * Ensure proper operation with empty, full, minimum, maximum, negative, and zero value for all variables. * Protect against out-of-limits, overflow, and underflow conditions. * Ensure “impossible” conditions are absolutely impossible. * Handle all incorrect input conditions. |  |  |  |  |
| Functional use | * Verify that all functions, procedures, or objects are fully understood and properly used. * Verify that all externally referenced abstractions are precisely defined. |  |  |  |  |
| Names | * Verify that all special names and types are clear or specifically defined. * Verify that the scopes of all variables and parameters are self-evident or defined. * Verify that all named objects are used within their declared scopes. |  |  |  |  |
| Standards | Review the design for conformance to all applicable design standards. |  |  |  |  |

# Table C58: C++ Code Review Checklist

| Program Name and Number: | | | | | |
| --- | --- | --- | --- | --- | --- |
| Purpose | To guide you in conducting an effective code review |  |  |  |  |
| General | * As you complete each review step, check off that item in the box to the right. * Complete the checklist for one program unit before you start to review the next. |  |  |  |  |
| Complete | Verify that the code covers all the design. |  |  |  |  |
| Includes | Verify that includes are complete. |  |  |  |  |
| Initialization | Check variable and parameter initialization   * at program initiation * at start of every loop * at function/procedure entry |  |  |  |  |
| Calls | Check function call formats:   * pointers * parameters * use of ‘&’ |  |  |  |  |
| Names | Check name spelling and use:   * Is it consistent? * Is it within declared scope? * Do all structures and classes use ‘.’ reference? |  |  |  |  |
| Strings | Check that all strings are   * identified by pointers * terminated in NULL |  |  |  |  |
| Pointers | Check that pointers are   * initialized NULL * only deleted after new * new pointers are always deleted after use |  |  |  |  |
| Output format | Check the output format:   * Line stepping is proper. * Spacing is proper. |  |  |  |  |
| {} pairs | Insure that the {} are proper and matched. |  |  |  |  |
| Logic operators | Verify the proper use of ==, =, ||, etc.  Check every logic function for proper (). |  |  |  |  |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  |  |  |  |
| Standards | Ensure that the code conforms to the coding standards. |  |  |  |  |
| File open and close | Verify that all files are   * properly declared * opened * closed |  |  |  |  |

Document Markings

Copyright 2020 Carnegie Mellon University. All rights reserved.  
  
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.  
  
Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.  
  
NO WARRANTY. THIS MATERIAL IS FURNISHED ON AN “AS-IS” BASIS WITH NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, ANY WARRANTY WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT, OR THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.  
  
[Distribution Statement A] This material has been approved for public release and unlimited distribution. The United States Government has Unlimited Rights in this material as defined by DFARS 252.227-7013.

The text and illustrations in this material are licensed by Carnegie Mellon University under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

The Creative Commons license does not extend to logos, trade marks, or service marks of Carnegie Mellon University.