Checkliste zum Sourcecode

Basis für ein Review ist das projektspezifisch vorgegebene Inhaltsverzeichnis, das formal und inhaltlich erfüllt sein muß. Alle Inhaltspunkte, insbesondere die vorgegebenen Unterpunkte, müssen enthalten und verständlich sowie eindeutig formuliert sein.

Allgemeine Fragen zur Unterlage

1. Ist ein Deckblatt vorhanden und ist es vollständig ausgefüllt?
2. Gibt es ein Inhaltsverzeichnis?
3. Stimmt das Inhaltsverzeichnis mit der Gliederung des Dokuments überein?
4. Hat die Unterlage einen eindeutigen Versionsstand?
5. Sind alle Seiten numeriert und eindeutig der Dokumentversion zugehörig zu identifizieren?
6. Ist das Dokument vollständig (fehlen keine Textstellen, Seiten, Abbildungen)?
7. Sind Querverweise eindeutig? (Wird in dem Kapitel, auf das verwiesen wird, auch der zutreffende Sachverhalt beschrieben?)
8. Sind alle Referenzdokumente aufgelistet?
9. Sind allgemeine firmeninterne und/oder projektspezifische Richtlinien zur Unterlagenerstellung eingehalten worden?
10. Sind Vertraulichkeitsstufen und Urheberrechte ausgewiesen?

Spezielle Fragen zur Entwicklungsunterlage

Abgleich der Entry-Unterlage "Entwurfsunterlage für die elementare Komponente"

11. Stimmt der definierte Sourcecode genau mit der geforderten Funktionalität überein?
12. Sind die Kommentare aus dem Entwurf als inline-Dokumentation übernommen und stimmen diese überein?
13. Entsprechen die Datendefinitionen in der Zielsprache mit der Definition im Entwurf überein?
14. Wurden bei bestimmten Resourcen (Speicher, Verwaltungsobjekte, ...) die vorgegebenen Ressourcenlimits beachtet?

Formale Anforderungen an den Source-Code

15. Ist der Sourcecode übersichtlich strukturiert? (Trennung von Bereichen, Einrücken von Bedingungen und Schleifen usw, Kommentare einheitlich abgesetzt)
16. Wurden abteilungs- bzw. projektspezifisch vorgegebene Coding-Standards eingehalten?
17. Hat der Sourcecode einen Copyright-Vermerk?
18. Sind die verwendeten Namen (Variable, Funktionen usw.) geeignete Bezeichner?

Inhaltliche Anforderungen an den Source-Code

Allgemeines

19. Sind sprach- und ggf. betriebssystem-spezifische Richtlinien und Standards eingehalten?
20. Entspricht die Implementierung den Entwurfsanforderungen?
21. Sind alle geforderten Kommentare vorhanden, aussagekräftig, leicht verständlich und richtig; entsprechen sie der Implementierung?
22. Werden im Sourcecode nur die benötigten include-Files eingezogen?
23. Sind gleiche Programmteile nur einmal vorhanden (als Makros, Prozeduren definiert)?
24. Gibt es nicht angsprochene Variablen in der Cross Reference List?
25. Ist der Source-Code konistent, wenn verschiedene Übersetzungs-Optionen verwendet werden?
26. Wird I/O nur auf geöffneten Files ausführt, werden Files korrekt geschlossen?
27. Sind Zugriffe auf „shared memory“-Bereiche entsprechend abgesichert?
28. Ist die Abhängigkeit von Konstanten untereinander oder von sonstigen Programmteilen dokumentiert?

Datendeklaration

29. Sind alle Variablen explizit vereinbart und Konstanten definiert?
30. Sind Felder richtig initialisiert?
31. Paßt die Initialisierung zu der Speicherklasse?
32. Sind korrekte Längen, Typen und Speicherklassen (lokal, global) zugewiesen?
33. Wurde in der Strukturdeklaration das „alignment“ berücksichtigt (evtl. Füllbytes)?
34. Sind lokale Variable nur für den Bereich definiert in dem sie auch benutzt werden?

35. Wurden alle globalen Variablen und Funktionen, die nur innerhalb eines Moduls (Sourcefile) sichtbar sein sollen, entsprechend definiert (z.B. „static“ in C)?
36. Gibt es Variablen mit ähnlichen Namen?

Datenreferenz

37. Werden Variablen angesprochen, die nicht initialisiert wurden?
38. Liegen bei Feldaufrufen die Indizies außerhalb der definierten Grenzen?
39. Werden nur ganzzahlige Indizes verwendet?
40. Gibt es „off by one“-Fehler (Randwert +-1) beim Indizieren von Feldern?
41. Existiert bei jeder Zeigerreferenz der angesprochene Speicherplatz (dangling reference)?
42. Sind bei einer Redefinition eines Speicherplatzes die Attribute korrekt (z.B. gespeichert wird eine REAL-Zahl, gelesen eine INTEGER)?
43. Hat der über Zeiger angesprochene Speicherplatz die erwartete Datenstruktur?
44. Wird bei einem Zugriff aus verschiedenen Programmteilen auf denselben Datenbereichen die gleiche Datenstruktur verwendet?

Fehler bei dynamischer Speicherverwaltung

45. Wird dynamisch angeforderter Speicherplatz wieder freigegeben?
46. Werden nicht erfolgreiche Anforderungen richtig behandelt?

Berechnungen

47. Werden Rechnungen mit inkonsistenten Datentypen durchgeführt (z.B. nicht-numerische)?
48. Gibt es Rechnungen mit unterschiedlichen Datentypen?
49. Gibt es Rechnungen mit Variablen verschiedener Länge?
50. Ist der Zuweisungswert im Wertebereich der Zielvariablen?
51. Gibt es einen Über- oder Unterlauf im Zwischenergebnis?
52. Kann der Divisior in einem Ausdruck Null werden?
53. Ist die Priorität der Operation richtig verstanden worden?
54. Wird die Größe einer Datenstruktur über eine Standardfunktion berechnet (z.B. „sizeof()“ in C?

Vergleich

55. Wird ein Vergleich zwischen inkonsistenten Variablen durchgeführt (verschiedene Typen)?
56. Sind die Vergleichsoperatoren richtig angewendet (z.B. in C: <, > <=, >=)?
57. Sind die Grenzwerte bei Vergleichen richtig erfaßt (z.B. in C: < oder <=)?
58. Gibt es Vergleiche zwischen Daten verschiedenen Typs oder verschiedener Länge?
59. Sind Konvertierungsregeln zu beachten?

Steuerfluß

60. Werden bei Mehrfachentscheidungen alle Sprungmöglichkeiten berücksichtigt?
61. Wird jede Schleife beendet, stimmt das Abbruchkriterium?
62. Wird eine Schleife aufgrund der Eingangsbedingung niemals ausgeführt?
63. Werden für eine Schleife alle Laufvariablen (Zähler, Pointer) initialisiert?
64. Wird ein Schleifenzähler innerhalb einer Schleife verändert?
65. Gibt es „off by one"-Fehler (z.B. eine Interation zuviel oder zuwenig)?
66. Wird in einer switch-Abfrage jeder Zweig abgeschlossen (z.B. break oder return)? falls nein, ist der „fall-through" kommentiert?
67. Wird das Programm oder die Subroutine beendet?

Schnittstellen

68. Passen die Schnittstellenvereinbarungen (Anzahl der Parameter, Reihenfolge, Typ und Länge) im aufrufenden und aufgerufenen Programm zusammen?
69. Werden Argumente verändert, die nur als Input dienen sollten?
70. Ist die Definition globaler Variablen über alle Moduln konsistent?
71. Wird der Returncode korrekt versorgt bzw. ausgewertet?
72. Wird der Returncode in allen Funktionen/Moduln einheitlich versorgt?

Fehlerbehandlung

73. Wird bei einem Fehlerausgang richtig aufgeräumt (müssen z.B Aktionen rückgängig gemacht werden)?
74. Werden Fehler in der rufenden Prozedur richtig ausgewertet?
75. Werden im Fehlerausgang Sequenzen ordentlich beendet?