

2 Testtechniken anwenden und bewerten

1. context Dreieck::klassifiziereDreieck(seite1, seite2, seite3) : DreieckArt

pre: seite1 > 0 and seite2 > 0 and seite3 > 0

2.

Eingabe	Ausgabe	
(5, 5, 5)	Gleichseitig	Für gleiche Seitenlänge muss ja Gleichseitig ausgespuckt werden
(10, 10, 3)	Gleichschenkelig	Für zwei gleiche Seiten muss Gleichschenkelig als Antwort folgen
(3, 4, 5)	Rechtwinklig	Ist halt rechtwinklig ...
(1, 4, 6)	Normal	Trifft auf keines der drei Bedingungen zu also sollte normal kommen.
(3, 10, 10)	Gleichschenkelig	Man guckt, ob die Reihenfolge eine Rolle spielt
(5, 4, 3)	Rechtwinklig	Gucken, ob die Reihenfolge eine Rolle spielt
(1, 1, 1)	Gleichseitig	Kleine Werte testen.

3.

Eingabe	Ausgabe
(3, 3, 6)	Gleichschenkelig
(4, 4, 4)	Gleichseitig
(1, 2, 3)	Normal
(3, 4, 5)	Rechtwinklig

4. Ja, schon alleine mit den Tests aus Nr. 3 erreicht man eine Abdeckung. Diese würde ich auch als Auswahl lassen. Zusätzlich kann man noch die Parameter vertauschen, um die OR-Klauseln abzudecken. Das Überdecken ist schon recht sinnvoll, kann aber bei vielen Bedingungen oder z.B. switch-Anweisungen sehr aufwendig werden.
5. Der 2. deckt ein Versagen auf (es sollte Gleichseitig kommen, es kommt aber gleichschenkelig). Dann noch der vierte, da kommt normal raus, obwohl rechtwinklig ausgegeben werden müsste.
6. Soweit ich sehe nicht ...

3 Diskussion

Auf der einen Seite kann man da hingehend argumentieren, dass der Programmierer seinen Code kennt, meint das er richtig ist, und vermutlich auch nur solche Test schreiben wird, so wie er auch den Code konzipiert hat. Andere kommen da meistens wohl aber noch auf andere Ideen, was man für Eingaben testen kann, die eventuell zu Versagen führen könnten. Auf der anderen Seite können kleine Fehler, sowas wie der Copy-and-Paste in Zeile 13 des Programms von Aufgabe 2 aber auch sehr schnell vom Programmierer entdeckt werden. Hier geht sowas vermutlich schneller, da sich ein dritter erst in den Code reindenken müsste. Ich würde also sagen, dass nicht nur der Programmierer seinen Code testen sollte, da ja z.B. Anforderungen falsch verstanden worden sein können und dies somit nicht aufgedeckt würde, aber ich würde die Regel auch nicht absolut sehen. Aus Erfahrung findet man in seinen Programm nämlich ja auch Fehler.