

#	Beschreibung	Seite
1	ARC in Swift in Aktion: Code und Konsolenausgabe	11
2	ARC in Swift in Aktion: Code und Konsolenausgabe	11
3	Erklärung des gezeigten Bitmusters als Referenzzähler	15
4	Die Klasse Node<T>	23
5	SIL-Repräsentation der Klasse Node<T> (deinit)	24
6	Die Klasse Node<T> als Knoten des Binärbaums	30
7	Deep-Copy Methode der Klasse Node<T>	31
8	Die Struktur Tree<T>	31
9	Copy-On-Write-Funktionalität im Getter der Wurzel	32
10	Copy-On-Write in Aktion	32
11	Ausgabe des Programms: Copy-On-Write sichert Wertsemantik	34
12	Unterschiedliches Verhalten von Variablenarten bei Wert- und Referenztypen	37
13	Beispiele von Kombinationen von Variablenarten und Typen	38
14	Hilfsmittel zum Untersuchen von Closures	41
15	Capture-Semantik von Referenztypen: Test1 und STDOUT	42
16	Capture-Semantik von Werttypen: Test2 und STDOUT	42
17	Closures mit Pass-By-Value-Semantik: Manuelles Anlegen einer Kopie	43
18	Definieren von Kopien in einer Capture List	43
19	Capture-Semantik mit Capture List bei Referenztypen: Test3 und STDOUT	43
20	Capture-Semantik mit Capture List bei Werttypen: Test4 und STDOUT	44
21	Capture-Semantik mit Capture List: Kopien und Mutationen wie gewohnt in Test 5 und 6	44
22	Anlegen der Variable "wrapper" vom Typ HeapWrapper in SIL	45
23	Eine Klasse mit Speicherleck durch eine Closure	46
24	Auflösung des Speicherlecks durch weak Capture in der Closure	47
25	Eine Struktur mit Speicherleck	48
26	Auflösung des Speicherlecks durch Capture einer Kopie in der Closure	48

#	Beschreibung	Seite
1	ARC: Objekte werden freigegeben, sobald der Referenzzähler auf 0 fällt	9
2	Obwohl die Variable gelöscht wurde, können die Objekte nicht freigegeben werden	9
3	Parent-Child-Beziehung mit einem Weak Pointer: <i>Child</i> kennt zwar <i>Parent</i> , erhöht aber dessen Referenzzähler nicht. Wenn die Variable <i>ptr</i> gelöscht wird, kann <i>Parent</i> deallokiert werden, da der Zähler auf 0 sinkt.	10
4	Struktur von Heapobjekten in Swift	13
5	Der PointerController verwaltet unterschiedliche Zeiger auf ein Objekt	16
6	Ein Tree wird durch mehrere PointerController untersucht	21
7	Aktivitätsdiagramm Dekrementierung und Dealloktation	25
8	Struktur von Strings in Swift	27
9	Ein String und eine Kopie davon teilen sich den selben Buffer	28
10	Strukturdiagramm Binärbaum mit Copy-On-Write-Semantik	30
11	Copy-On-Write: Sharing vor Modifikation	33
12	Copy-On-Write: Kopie bei Modifikation	33
13	Closure Capture: Variablen werden auf den Heap verschoben	45
14	Eine Closure Erzeugt einen Referenzzyklus	46
15	Durch die schwache Referenz wird ein Zyklus vermieden	47
16	Speicherabbild der Struktur mit Speicherleck	48

#	Beschreibung	Seite
1	Deklaration der Referenzzähler	14
2	Bedeutung der Bits in den Referenzzählern	15
3	Atomare Operatoren zur Modifikation von Referenzzählern	19
4	Operator zum Dekrementieren des starken Referenzzählers	20
5	Reaktion auf Dekrementieren des Referenzzählers wenn dieser auf 0 fällt	23
6	Erklärung der SIL-Funktion <code>destroy_addr</code>	24
7	Optimierung von Copy-On-Write bei Strings in der Swift-Standardbibliothek	34
8	Beispiel einer Funktion mit <code>@noescape</code> : Die Funktion <code>map</code> des <code>Optional-Enums</code>	50

#	Beschreibung	Seite
1	Initialzustand: ein einzelner starker Pointer ist gesetzt	17
2	Einige Pointer sind gesetzt	17
3	Die Anzahl der strong Pointers fällt auf 0	18
4	Die Anzahl der weak Pointers fällt auf 0. Der Speicher wurde freigegeben	18
5	Initialzustand: Alle Knoten haben einen einzelnen strong Pointer	22
6	Keine starken Zeiger auf die Wurzel: Alle Kindknoten sind freigegeben	22

#	Beschreibung	Quelle	Seite
1	Python: Reference Counts & Reference Counting in Python	https://docs.python.org/2/extending/extending.html#reference-counts	10
2	Structs and Classes: Unowned References (Seite 136)	Chris Eidhof, Airspeed Velocity: Advanced Swift (2016)	11
3	UnsafePointer Structure Reference	https://developer.apple.com/library/ios/documentation/Swift/Reference/Swift_UnsafePointer_Structure/index.html	12
4	Friday Q&A 2015-12-11: Swift Weak References	https://www.mikeash.com/pyblog/friday-qa-2015-12-11-swift-weak-references.html	18
5	Structs and Classes (Seite 117)	Chris Eidhof, Airspeed Velocity: Advanced Swift (2016)	27
6	Swift Enhancement: Proposal 125: Remove NonObjectiveCBase and isUniquelyReferenced	https://github.com/apple/swift-evolution/blob/master/proposals/0125-remove-nonobjectivecbase.md	29
7	ArraySlice Structure Reference	https://developer.apple.com/library/ios/documentation/Swift/Reference/Swift_ArraySlice_Structure/index.html	35
8	Guaranteed Optimization and Diagnostic Passes: Memory promotion	http://apple-swift.readthedocs.io/en/latest/SIL.html#guaranteed-optimization-and-diagnostic-passes	37
9	SIL in the Swift Compiler: SILGen und canonical SIL	http://apple-swift.readthedocs.io/en/latest/SIL.html#sil-in-the-swift-compiler	38
10	Closure Capture Semantics, Part 1: "Captured variables are evaluated on execution"	http://alisoftware.github.io/swift/closures/2016/07/25/closure-capture-1/	42
11	Swift Intermediate Language: "Promotion eliminates byref capture"	http://llvm.org/devmtg/2015-10/slides/GroffLattner-SILHighLevelIR.pdf	45
12	Swift Enhancement: Proposal 35: Limiting inout capture to @noescape context	https://github.com/apple/swift-evolution/blob/master/proposals/0035-limit-inout-capture.md	49
13	NSHint: @noescape Attribute	http://nshint.io/blog/2015/10/23/noescape-attribute/	50
14	Swift Enhancement: Proposal 103: Make non-escaping closures the default	https://github.com/apple/swift-evolution/blob/master/proposals/0103-make-noescape-default.md	50