

Inhaltsverzeichnis

1	Vorbemerkungen	1
1.1	namespace std.....	1
1.2	Header-Dateien	2
1.3	Eigenschaften	2
1.4	Kanonische Klassen	4
1.5	Komplexität und Aufwand.....	5
2	Konzeption und Entwicklung der STL	7
2.1	Eine Feldklasse	7
2.2	Eine Listenklasse	8
2.3	Folgerungen	9
2.4	Standardisierung der Suchfunktion	9
2.5	Die Schnittstelle eines Iterators	11
2.6	Ein Iterator für die Feldklasse	12
2.7	Ein Iterator für die Listenklasse	13
2.8	Zusammenfassung.....	13
2.9	const-Korrektheit	14
2.10	Flexible Suche mit Funktionsobjekten.....	16
2.11	Kleinste Bausteine	18
2.12	Programmübersetzungs- und -laufzeit.....	20
2.13	Der Rückgabetypp für eine Zählfunktion.....	22
2.14	Fehlerquellen.....	25
2.15	Ist die STL objektorientiert?.....	27
2.16	Einsatz der Standardbibliothek.....	29
2.17	Aufgaben	30
3	Funktionsobjekte	33
3.1	Basisklassen für Funktionsobjekte	33
3.2	Arithmetische, logische und Vergleichsoperationen.....	35
3.3	Projektionen.....	38
3.3.1	binder1st.....	38
3.3.2	bind1st.....	39
3.3.3	binder2nd.....	39
3.3.4	bind2nd.....	39
3.3.5	Beispiel	39
3.4	Negativierer	40
3.4.1	unary_negate.....	40
3.4.2	not1.....	41
3.4.3	binary_negate	41
3.4.4	not2.....	42
3.5	Adapter für Funktionszeiger	42
3.5.1	pointer_to_unary_function	43

3.5.2	ptr_fun für einstellige Funktionen.....	43
3.5.3	pointer_to_binary_function.....	43
3.5.4	ptr_fun für zweistellige Funktionen.....	44
3.6	Adapter für Zeiger auf Elementfunktionen.....	44
3.6.1	mem_fun_ref_t und const_mem_fun_ref_t.....	46
3.6.2	mem_fun_ref für Elementfunktionen ohne Argument.....	47
3.6.3	mem_fun1_ref_t und const_mem_fun1_ref_t.....	47
3.6.4	mem_fun_ref für Elementfunktionen mit Argument.....	48
3.6.5	mem_fun_t und const_mem_fun_t.....	48
3.6.6	mem_fun für Elementfunktionen ohne Argument.....	49
3.6.7	mem_fun1_t und const_mem_fun1_t.....	49
3.6.8	mem_fun für Elementfunktionen mit Argument.....	50
3.7	Funktionen zusammensetzen.....	50
3.8	Aufgaben.....	51
4	Hilfsmittel	53
4.1	Vergleichsoperatoren.....	53
4.2	pair.....	54
4.3	Aufgaben.....	55
5	Container	57
5.1	vector.....	58
5.2	Allgemeine Anforderungen an Container.....	60
5.3	Anforderungen an reversible Container.....	65
5.4	Anforderungen an sequenzielle Container.....	66
5.5	Optionale Anforderungen an sequenzielle Container.....	71
5.6	Weitere Funktionen.....	74
5.7	deque.....	77
5.8	list.....	81
5.9	Auswahl nach Aufwand.....	88
5.10	Aufgaben.....	88
6	Containeradapter	91
6.1	stack.....	91
6.2	queue.....	93
6.3	priority_queue.....	94
6.4	Aufgaben.....	97
7	Assoziative Container	99
7.1	map.....	100
7.2	Anforderungen an assoziative Container.....	102
7.3	Der Indexoperator der Klasse map.....	107
7.4	multimap.....	108
7.5	set.....	111
7.6	multiset.....	114
7.7	Übersicht der Container.....	116
7.8	Aufgaben.....	118

8	Iteratoren	119
8.1	Iteratoranforderungen	119
8.2	Input-Iteratoren	121
8.3	Output-Iteratoren.....	122
8.4	Forward-Iteratoren	123
8.5	Bidirectional-Iteratoren	125
8.6	Random-Access-Iteratoren	125
8.7	Übersicht über die Iteratorkategorien	127
8.8	Hilfsklassen und -funktionen für Iteratoren.....	128
8.8.1	iterator_traits	128
8.8.2	Die Basisklasse iterator	130
8.8.3	Iteratorfunktionen	131
8.8.3.1	advance.....	131
8.8.3.2	distance	132
8.9	Reverse-Iteratoren.....	133
8.10	Insert-Iteratoren.....	136
8.10.1	back_insert_iterator	136
8.10.2	front_insert_iterator	138
8.10.3	insert_iterator	139
8.11	Stream-Iteratoren	140
8.11.1	istream_iterator	140
8.11.2	ostream_iterator	142
8.12	Aufgaben.....	144
9	Algorithmen	147
9.1	Übersicht.....	149
9.2	Nichtmodifizierende Algorithmen	153
9.2.1	for_each.....	153
9.2.2	find und find_if	154
9.2.3	find_end.....	156
9.2.4	find_first_of	157
9.2.5	adjacent_find	159
9.2.6	count und count_if.....	160
9.2.7	mismatch.....	161
9.2.8	equal.....	163
9.2.9	search	164
9.2.10	search_n.....	165
9.3	Modifizierende Algorithmen	166
9.3.1	copy.....	166
9.3.2	copy_backward.....	167
9.3.3	swap	169
9.3.4	iter_swap	169
9.3.5	swap_ranges	170
9.3.6	transform	171
9.3.7	replace und replace_if	173
9.3.8	replace_copy und replace_copy_if	174
9.3.9	fill und fill_n	175

9.3.10	generate und generate_n.....	176
9.3.11	remove_copy und remove_copy_if.....	178
9.3.12	remove und remove_if.....	179
9.3.13	unique_copy.....	180
9.3.14	unique.....	182
9.3.15	reverse.....	183
9.3.16	reverse_copy.....	184
9.3.17	rotate.....	185
9.3.18	rotate_copy.....	186
9.3.19	random_shuffle.....	187
9.3.20	partition und stable_partition.....	188
9.4	Sortieren und ähnliche Operationen.....	190
9.4.1	sort.....	190
9.4.2	stable_sort.....	191
9.4.3	partial_sort.....	193
9.4.4	partial_sort_copy.....	194
9.4.5	nth_element.....	195
9.5	Binäre Suchalgorithmen.....	196
9.5.1	lower_bound.....	196
9.5.2	upper_bound.....	198
9.5.3	equal_range.....	199
9.5.4	binary_search.....	200
9.6	Mischalgorithmen.....	201
9.6.1	merge.....	201
9.6.2	inplace_merge.....	203
9.7	Mengenalgorithmen für sortierte Bereiche.....	204
9.7.1	includes.....	204
9.7.2	set_union.....	205
9.7.3	set_intersection.....	206
9.7.4	set_difference.....	208
9.7.5	set_symmetric_difference.....	209
9.7.6	Mengenalgorithmen für multiset-Objekte.....	210
9.8	Heap-Algorithmen.....	211
9.8.1	make_heap.....	212
9.8.2	pop_heap.....	212
9.8.3	push_heap.....	212
9.8.4	sort_heap.....	213
9.8.5	Beispielprogramm.....	213
9.9	Minimum und Maximum.....	214
9.9.1	min.....	214
9.9.2	max.....	215
9.9.3	min_element.....	216
9.9.4	max_element.....	217
9.10	Permutationen.....	218
9.10.1	lexicographical_compare.....	218
9.10.2	next_permutation.....	219
9.10.3	prev_permutation.....	220

9.11	Numerische Algorithmen	222
9.11.1	accumulate	222
9.11.2	inner_product	223
9.11.3	adjacent_difference	223
9.11.4	partial_sum	224
9.12	Erweitern der Bibliothek mit eigenen Algorithmen	225
9.13	Präfix- versus Postfixoperatoren	227
9.14	Aufgaben	228
10	Allokatoren	231
10.1	Der Standardallokator	231
10.2	allocator<void>	234
10.3	Aufgaben	235
11	Strings	237
11.1	Containereigenschaften	238
11.2	basic_string	240
11.3	Implementierungsdetails	254
11.4	Aufgaben	255
12	Streams	257
12.1	Überblick	257
12.2	ios_base	259
12.2.1	Formatierung	261
12.2.2	Streamstatus	266
12.2.3	Initialisierung	267
12.2.4	Nationale Einstellungen	268
12.2.5	Synchronisation	269
12.3	basic_ios	270
12.3.1	Statusfunktionen	272
12.3.2	Ausnahmen	273
12.4	basic_ostream	274
12.4.1	sentry	275
12.4.2	Formatierte Ausgaben	276
12.4.3	Unformatierte Ausgaben	278
12.5	basic_istream	278
12.5.1	sentry	279
12.5.2	Formatierte Eingaben	280
12.5.3	Unformatierte Eingaben	281
12.6	basic_iostream	284
12.7	Ein- und Ausgabe von Objekten benutzerdefinierter Klassen	284
12.8	Namensdeklarationen	286
12.9	Manipulatoren	287
12.9.1	Manipulatoren ohne Parameter	287
12.9.2	Manipulatoren mit einem Parameter	289
12.10	Positionieren von Streams	291
12.11	Streams für Dateien	292
12.11.1	Modi zum Öffnen von Dateien	292

12.11.2	Die Header-Datei <fstream>	293
12.12	Streams für Strings	296
12.13	Aufgaben	299
13	Weitere Komponenten der C++-Standardbibliothek	301
13.1	auto_ptr	301
13.2	bitset	306
13.3	vector<bool>	312
13.4	complex	314
13.5	numeric_limits	320
13.6	valarray	327
13.6.1	slice und slice_array	333
13.6.2	gslice und gslice_array	336
13.6.3	mask_array	338
13.6.4	indirect_array	340
13.7	Aufgaben	341
14	Zeiger in Containern verwalten	343
14.1	Beispielklassen	343
14.2	Ein set-Objekt verwaltet Zeiger	346
14.3	Smart-Pointer	347
14.4	Ein set-Objekt verwaltet Smart-Pointer	348
14.5	Ein set-Objekt verwaltet Zeiger mittels Funktionsobjekten	349
14.6	Ein map-Objekt verwaltet Zeiger	351
14.7	Aufgaben	353
15	Lösungen	355
Anhang		385
A	Die Containerklasse list	385
B	Literatur	398
Index		399