

Stefan Kuhlins • Martin Schader

Die C++-Standardbibliothek

Einführung und Nachschlagewerk

Dritte, überarbeitete Auflage

Mit 77 Abbildungen und 37 Tabellen



Springer

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	<u>Funktionsobjekte</u>	33[^]
3.1	Basisklassen für Funktionsobjekte.....	33
3.2	Arithmetische, logische und Vergleichsoperationen.....	35
3.3	Projektionen.....	38
3.3.1	binderist.....	38
3.3.2	bindist.....	39
3.3.3	binder2nd.....	39
3.3.4	bind2nd.....	40
3.3.5	Beispiel.....	40
3.4	Negativierer.....	41
3.4.1	unary_negate.....	41
3.4.2	notl.....	42
3.4.3	binary_negate.....	42
3.4.4	not2.....	42
3.5	Adapter für Funktionszeiger.....	43
3.5.1	pointerjo_unary_function.....	44

Inhaltsverzeichnis

3.5.2	ptr_fun für einstellige Funktionen.....	44
3.5.3	pointer_to_binary_function.....	44
3.5.4	ptr_fun für zweistellige Funktionen.....	45
3.6	Adapter für Zeiger auf Elementfunktionen.....	45
3.6.1	mem_fun_ref,t und const_mem_fun_ref_t.....	47
3.6.2	mem_fun_ref für Elementfunktionen ohne Argument.....	47
3.6.3	mem_funl_ref_t und const_mem_funl_ref_t.....	48
3.6.4	mem_njn_ref für Elementfunktionen mit Argument.....	49
3.6.5	mem_nin_t und const_mem_fun_t.....	49
3.6.6	mem_fun für Elementfunktionen ohne Argument.....	50
3.6.7	mem_funl_t und const_mem_funl_t.....	50
3.6.8	mem_fun für Elementfunktionen mit Argument.....	51
3.7	Funktionen zusammensetzen.....	51
3.8	Aufgaben.....	52
4	<u>Hilfsmittel</u>	55[^]
4.1	Vergleichsoperatoren.....	55
4.2	pair.....	56
4.3	Aufgaben.....	57
5	<u>Container</u>	59[^]
5.1	vector.....	60
5.2	Allgemeine Anforderungen an Container.....	62
5.3	Anforderungen an reversible Container.....	68
5.4	Anforderungen an sequenzielle Container.....	68
5.5	Optionale Anforderungen an sequenzielle Container.....	73
5.6	Weitere Funktionen.....	76
5.7	deque.....	79
5.8	list.....	84
5.9	Auswahl nach Aufwand.....	90
5.10	Aufgaben.....	91
6	<u>Containeradapter</u>	93
6.1	Stack.....	93
6.2	queue.....	95
6.3	priority^queue.....	96
6.4	Aufgaben.....	99
7	Assoziative Container	101
7.1	map.....	102
7.2	Anforderungen an assoziative Container.....	104
7.3	Der Indexoperator der Klasse map.....	109
7.4	multimap.....	110
7.5	set.....	113
7.6	multiset.....	116
7.7	Elemente in set und multiset modifizieren.....	118
7.8	Übersicht der Container.....	120
7.9	Aufgaben.....	120

8	<u>Iteratoren</u>	123
8.1	Iteratoranforderungen.....	123
8.2	Input-Iteratoren.....	125
8.3	Output-Iteratoren.....	126
8.4	Forward-Iteratoren.....	127
8.5	Bidirectional-Iteratoren.....	129
8.6	Random-Access-Iteratoren.....	129
8.7	Übersicht über die Iteratorkategorien.....	131
8.8	Hilfsklassen und -funktionen für Iteratoren.....	132
	8.8.1 iterator_traits.....	132
	8.8.2 Die Basisklasse iterator.....	134
	8.8.3 Iteratorfunktionen.....	135
	8.8.3.1 advance.....	135
	8.8.3.2 distance.....	136
8.9	Reverse-Iteratoren.....	137
8.10	Insert-Iteratoren.....	140
	8.10.1 back_inserter.....	141
	8.10.2 front_inserter.....	142
	8.10.3 inserter.....	143
8.11	Stream-Iteratoren.....	144
	8.11.1 istream_iterator.....	145
	8.11.2 ostream_iterator.....	147
8.12	Aufgaben.....	148
9	<u>Algorithmen</u>	15[^]
9.1	Übersicht.....	153
9.2	Nichtmodifizierende Algorithmen.....	157
	9.2.1 for_each.....	157
	9.2.2 find und find_if.....	158
	9.2.3 find_end.....	160
	9.2.4 find_first_of.....	161
	9.2.5 adjacent_find.....	163
	9.2.6 count und count_if.....	164
	9.2.7 mismatch.....	165
	9.2.8 equal.....	167
	9.2.9 search.....	168
	9.2.10 search_j.....	169
9.3	Modifizierende Algorithmen.....	170
	9.3.1 copy.....	170
	9.3.2 copy_backward.....	172
	9.3.3 swap.....	173
	9.3.4 iter_swap.....	173
	9.3.5 swap_ranges.....	174
	9.3.6 transform.....	175
	9.3.7 replace und replace_if.....	177
	9.3.8 replace_copy und replace_copy_if.....	178
	9.3.9 fill und fill_n.....	179

Inhaltsverzeichnis

9.3.10	generate und generate_n.....	180
9.3.11	remove_copy und remove_copy_if.....	182
9.3.12	remove und remove_if.....	183
9.3.13	unique_copy.....	185
9.3.14	unique.....	187
9.3.15	reverse.....	188
9.3.16	reverse_copy.....	189
9.3.17	rotate.....	190
9.3.18	rotate_copy.....	191
9.3.19	random_shuffle.....	192
9.3.20	partition und stable_partition.....	193
9.4	Sortieren und ähnliche Operationen.....	195
9.4.1	sort.....	195
9.4.2	stable_sort.....	196
9.4.3	partial_sort.....	198
9.4.4	partial_sort_copy.....	199
9.4.5	nth^element.....	200
9.5	Binäre Suchalgorithmen.....	201
9.5.1	Lower_bound.....	201
9.5.2	upper_bound.....	203
9.5.3	equal_range.....	204
9.5.4	binary_search.....	205
9.5.5	Schlüsselsuche mit Funktionsobjekt.....	206
9.6	Mischalgorithmen.....	207
9.6.1	merge.....	207
9.6.2	inplace_merge.....	209
9.7	Mengenalgorithmen für sortierte Bereiche.....	210
9.7.1	includes.....	210
9.7.2	set_union.....	211
9.7.3	set_intersection.....	213
9.7.4	set_difference.....	214
9.7.5	set_symmetric_difference.....	215
9.7.6	Mengenalgorithmen für multiset-Objekte.....	216
9.8	Heap-Algorithmen.....	217
9.8.1	make_heap.....	218
9.8.2	pop_heap.....	218
9.8.3	push_heap.....	218
9.8.4	sort_heap.....	219
9.8.5	Beispielprogramm.....	219
9.9	Minimum und Maximum.....	221
9.9.1	min.....	221
9.9.2	max.....	221
9.9.3	min_element.....	222
9.9.4	max_element.....	223
9.10	Permutationen.....	224
9.10.1	LexicographicalCompare.....	225
9.10.2	next_permutation.....	226

Inhaltsverzeichnis

9.10.3	prev_permutation.....	227
9.11	Numerische Algorithmen.....	228
9.11.1	accumulate.....	228
9.11.2	inner_product.....	229
9.11.3	adjacent_difference.....	230
9.11.4	partial_sum.....	231
9.12	Erweitern der Bibliothek mit eigenen Algorithmen.....	231
9.13	Präfix- versus Postfixoperatoren.....	233
9.14	Aufgaben.....	234
10	<u>Allokatoren</u>	237[^]
10.1	Der Standardallokator.....	237
10.2	allocator<void>.....	241
10.3	Aufgaben.....	241
11	<u>Strings</u>	243
11.1	Containereigenschaften.....	244
11.2	basic_string.....	246
11.3	Implementierungsdetails.....	260
11.4	Sortieren von Strings.....	261
11.5	Aufgaben.....	262
12	<u>Streams</u>	265
12.1	Überblick.....	265
12.2	ios_base.....	268
12.2.1	Formatierung.....	270
12.2.2	Streamstatus.....	275
12.2.3	Initialisierung.....	276
12.2.4	Nationale Einstellungen.....	276
12.2.5	Synchronisation.....	278
12.3	basic_ios.....	278
12.3.1	Statusfunktionen.....	280
12.3.2	Ausnahmen.....	281
12.4	basic_ostream.....	282
12.4.1	sentry.....	283
12.4.2	Formatierte Ausgaben.....	285
12.4.3	Unformatierte Ausgaben.....	286
12.5	basic_istream.....	286
12.5.1	sentry.....	288
12.5.2	Formatierte Eingaben.....	288
12.5.3	Unformatierte Eingaben.....	289
12.6	basic_ostringstream.....	292
12.7	Ein- und Ausgabe von Objekten benutzerdefinierter Klassen.....	293
12.8	Namensdeklarationen.....	294
12.9	Manipulatoren.....	295
12.9.1	Manipulatoren ohne Parameter.....	295
12.9.2	Manipulatoren mit einem Parameter.....	298
12.10	Positionieren von Streams.....	299

Inhaltsverzeichnis

12.11	Streams für Dateien.....	301
12.11.1	Modi zum Öffnen von Dateien.....	301
12.11.2	Die Header-Datei <fstream>.....	302
12.12	Streams für Strings.....	305
12.13	Aufgaben.....	307
13	Weitere Komponenten der C++-Standardbibliothek	309
13.1	auto_ptr.....	309
13.2	bitset.....	315
13.3	vector<bool>.....	321
13.4	complex.....	323
13.5	numeric_limits.....	329
13.6	valarray.....	335
13.6.1	slice und slice_array.....	341
13.6.2	gslice und gslice_array.....	344
13.6.3	mask_array.....	346
13.6.4	indirect_array.....	348
13.7	Aufgaben.....	349
14	Zeiger in Containern verwalten	351
14.1	Beispielklassen.....	351
14.2	Ein set-Objekt verwaltet Zeiger.....	354
14.3	Smart-Pointer.....	355
14.4	Ein set-Objekt verwaltet Smart-Pointer.....	356
14.5	Ein set-Objekt verwaltet Zeiger mittels Funktionsobjekten.....	357
14.6	Ein map-Objekt verwaltet Zeiger.....	359
14.7	Aufgaben.....	361
15	Lösungen	363
Anhang		397
A	Die Containerklasse sList.....	397
B	Die Klasse LogAllocator.....	411
C	Literatur.....	413
Index		414