

Indhold

Tak	7	Parameteriserede funktioner	62
		Parameteriserede medlemsfunktioner	62
DEL 1: INTRODUKTION OG GENNEMGANG	9	Hvornår skal man angive en parameter?	63
Kapitel 1: Hvorfor nok en bog om C++?	11	Templates med mere end en parameter	63
Det allerhelligste	12	Parameteriserede typer i flere niveauer - nej tak!	63
Tre superbe ideer i C++	14	Nedarvning	64
Læsevejledning	16	Kombinationer af parameteriserede og simple typer	64
Lige et par ord om kodestilen	17	Usikre typer som offentlige basisklasser - nej tak!	65
		Usikre typer som private basisklasser	65
		Usikre typer som datamedlemmer	66
Kapitel 2: C++-syntaks og trivialiteter	19	Kapitel 4: Exception-håndtering	69
Variable og konstanter	19	ANSI-standardbaseret exception-håndtering	69
Const	19	Syntaksen for at kaste (throw) exceptions	70
Stak- eller heap-objekter	23	Syntaksen for at fange (catch) exceptions	72
Virkefelter og funktioner	26	Constructorer og destructorer	74
Virkefelter	26	Exception-håndtering, som ikke følger standarden	76
Det globale navnerum	29	Konventioner i denne bog	77
Overdefinition (overloading)	30	DEL 2: VIDERESTILLING	79
Adgangsregler	31	Kapitel 5: Smart pointere	81
Typer og operatorer	35	Dumme pointere	81
Constructorer	35	Smart pointer-konceptet	83
Initialisering af globale objekter	41	Operator ->	83
Destructorer	43	Parameteriserede smart pointere	84
Tildelinger	44	Hierarkier af smart pointere	85
Overdefinition af operatorer	49	Pointer-aritmetik	86
Kapitel 3: Templates og typesikkerhed	59	Hvor meget koster en smart pointer?	87
Hvad er en template, og hvorfor er den vigtig?	59	Programmer	87
Problemet	59	Undgå NULL-pointere	87
Lappeløsninger	60	Debugging og tracing	89
Templates er skønne makroer	61	Caching	91
Template-syntaks	61		
Parameteriserede typer	61		

Kapitel 6: Master pointere og C++ handles	93	Kapitel 8: Samlinger, markører og gennemløbere	121
Master pointerens semantik	93	Arrays og []-operatoren	121
Constructorer	94	Grænsecheck og tildelinger	121
Destructorer	95	Argumenter til []-operatoren,	
Kopiering	96	som ikke er talværdier	122
Tildelinger	96	Simulering af flerdimensionale arrays	122
Prototypen på en master point template	97	Flere overdefinitioner af []-operatoren	123
C++ handles	98	Virtuelle []-operatører	123
Hvad skal vi bruge det til?	99	Markører	124
Objekttæller	99	En simpel halvtom array-klasse	124
Skrivebeskyttede pointere	100	Markører og halvtomme arrays	125
Læse-skrive pointere	101	Konverterings- og ->-operatører	127
		Har jeg ikke set det tidligere?	127
Kapitel 7: Facets og andre smarte pointere	103	Gennemløbere	127
Interface-pointere	103	Aktive gennemløbere	128
Kopiering af interfacet	103	Passive gennemløbere	129
Beskyttelse af det underliggende objekt	104	Hvad er det bedste?	129
Ændring af interfacet	106	Tvivlsomme, men almindelige variationer	130
Facets	106	Bedre variationer	131
Konvertering af en pointer til en facet	107	En abstrakt array-gennemløber	132
Gemstones (ædelsten)	108	Samlingsoperatører	133
Variationer over temaet facets	109	Avancerede markører og flertrådgennemløbere	135
Indkapsling af det underliggende objekt	112	Private kopier af samlinger	137
Validering af facetter	112	Simplificering af private samlinger	138
Sikring af konsistens	113	Interne og eksterne gennemløbere	139
Facets og smarte pointere	115	Tidsstempling	141
Omdan en gemstone til en master pointer	116	Et eksempel	143
Udskiftelige typer	116		
Polymorfe pointere	116	Kapitel 9: Transaktioner og brillante pointere	149
Udvælgelse af typen på det underliggende objekt under opstarten	117	Nogle knudrede designproblemer	149
Udvælgelse af den underliggende klasse på kørselstidspunktet	117	Transaktioner	149
Proxies	117	Fortryd	150
Functors	118	Er det nok motivation?	150
		Aftryk og pointere	151
		En simpel aftrykspointer	151
		Aftryksstakke	152
		Aftryk af automatiske objekter	153
		Aftryk af pointere (pointer-aftryk)	156

Kombinationer og variationer	157	Revision af dobbelt viderestilling	184
Transaktioner og fortryd	158	Copy constructor og -=operatorer - nej tak!	185
Transaktioner og låse	159	Klasseobjekter	185
Const pointere	160	Klasseinformation	186
Låse pointere	161	Mere om kirkegårdsfunktioner	187
Skabelse og sletning af objekter	162	Fremskaffelse af et objekts klasse	188
Simplificering af objektskabelsen	164	Exemplars	189
Fortryd	164		
Variationer	164	Kapitel 12: Usynlige pointere	191
Flere niveauer af låse	165	Det basale koncept	191
Deadlocks og køteknik	166	Indkapsling af pointeren og det	
Køteknik og deadlocks	166	underliggende objekt	191
Flerniveaus tilbagerulning	166	Fabriksfunktioner	192
Pladsoptimering	167	Reference til pointere	193
Nogle afsluttende bemærkninger	167	Anvendelse af pointere, som ikke	
		er master pointere	193
DEL 3: TYPETIPS	169	Master pointere	194
Kapitel 10: Multipel viderestilling	171	Tildelinger	195
Sammenhængende klassehierarkier	171	Dobbelt viderestilling - igen	196
Udskiftning af nedarvede klasser	172	Dobbelt dobbelt viderestilling	196
Normaliseret nedarvning	172	Hvor længe lever resultatet?	198
Indkapslede nedarvede klasser	173	Selvmodifikation og udskiftelighed	198
Multipel viderestilling	174	Multipel viderestilling	200
Dobbelt viderestilling	175	Programmer, som bruger usynlige pointere	200
Sammenhængende dobbelt viderestilling	176	Caching	200
Højere ordens viderestilling	177	Distribuerede objekter og proxy'er	200
Viderestillinger og konverteringer	178	Avancerede distribuerede arkitekturer	201
Vier ikke helt færdige	179		
		DEL 4: LAGERSTYRING	203
Kapitel 11: Fabriksfunktioner og klasseobjekter	181	Kapitel 13: Overdefinering af new og delete	205
Fabriksfunktioner	181	Overdefinering af new og delete	205
Make-funktioner	182	En simpel fripladsliste	205
Symbolklasser og overdefinerede		Nedarvning af new- og delete-operatorerne	208
make-funktioner	182	Argumenter til new-operatoren	209
Optimering ved at benytte fabriksfunktioner	182	Faseopdelt objektkonstruktion	210
Separering af kode via fabriksfunktioner	183	Faseopdelt destruktion	211
Kirkegårdsfunktioner	184	Hvem kontrollerer allokeringerne?	212

Global allokering og deallokering	212	Gradvis kopiering	247
Klassespecifik allokering og deallokering	212	Eksterne objekter	248
Brugerstyret lagerhåndtering	213	Yderligere C++-forfinelse af Bakers algoritme	249
Klasseobjekter og fabriksfunktioner	213	Et-rums komprimering	251
Master pointer-baseret lagerstyring	213	VoidPtr-basisklassen	252
Perspektivet	217	Master pointer-puljen	252
Kapitel 14: Grundlæggende lagerstyring	219	Master pointer-gennemløberen	253
Byggeklodser	219	Komprimeringsalgoritmen	253
Blokallokering	219	Optimering	254
Skjult information	221	Afsluttende bemærkninger	254
Fripladslisten	222	Kapitel 16: Oprydning	255
Referencetællere	224	Adgangsveje	255
En referencetæller-basisklasse	224	Ydergrænsen	255
Referencetællende pointer	224	Interiøret	257
Referencetællende master pointer	225	Analyse af instanserne	257
Referencetællende handles	226	Gennemløb af objektstrukturen	258
Problemer med referencetællere	226	Oprydning med Bakers algoritme	259
Referenceoptælling og master pointer	227	Den svage handle template	260
Lagerrum	227	Den stærke handle template	260
Opdeling efter klasse	228	Master pointer-gennemløbere	261
Opdeling efter størrelse	229	Gennemløb af pointerne	263
Opdeling efter anvendelse	229	Optimering	266
Opdeling efter adgangstype	229	Eksterne objekter	266
Stak- eller heap-lagerrum	229	Multiple lagerrum	267
Kapitel 15: Lagerkomprimering	231	Et-rums oprydning og komprimering	267
Pointer-identifikation	231	Vil du virkelig kalde destructorerne?	267
Hvor kan pointerne komme fra?	231	Kun adgang for professionelle vovehalse	268
Hvordan finder man alle pointerne?	236	Moder-til-alle-objekter løsningen	268
Handles, handles overalt	238	Organisering af lageret	269
Hovedtrækkene i arkitekturen	238	Bestemmelse af ydergrænsen	270
Master pointer	239	Gennemløb af interiøret	270
Variationer	243	Oprydning	271
Optimering	244	Gradvis oprydning	271
Bakers algoritme	244	Afsluttende bemærkninger	271
Objektlagerrummet	245	Stikord	273